Mattia Epifani, Pasquale Stirparo

Learning iOS Forensics

Second Edition

A practical guide to analyzing iOS devices with the latest forensics tools and techniques

Packt

Learning iOS Forensics

Second Edition

A practical guide to analyzing iOS devices with the latest forensics tools and techniques

Mattia Epifani
Pasquale Stirparo

Packt

BIRMINGHAM - MUMBAI

Learning iOS Forensics

Second Edition

First published: March 2015

Second edition: September 2016

Production reference: 1280916

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham
B3 2PB, UK.

ISBN 978-1-78588-208-1

www.packtpub.com

Credits

Authors

Mattia Epifani
Pasquale Stirparo

Reviewer

Michael Yasumoto

Commissioning Editor

Kunal Parikh

Acquisition Editor

Sonali Vernekar

Content Development Editor

Sanjeet Rao

Technical Editor

Devesh Chugh

Copy Editor

Tom Jacob

Project Coordinator

Judie Jose

Proofreader

Safis Editing

Indexer

Pratik Shirodkar

Graphics

Kirk D'Penha

Production Coordinator

Shantanu Zagade

About the Authors

Mattia Epifani (`@mattiaep`) is the CEO at Reality Net-System Solutions, an Italian consulting company involved in InfoSec and digital forensics.

He works as a digital forensics analyst for judges, prosecutors, lawyers, and private companies. He is a court witness and digital forensics expert.

He obtained a university degree in computer science in Genoa, Italy, and a postgraduate specialization course in computer forensics and digital investigations in Milan, Italy. Over the last few years, he obtained several certifications in digital forensics and ethical hacking (GCFA, GREM, GNFA, GCWN, GMOB, CIFI, CEH, CHFI, ACE, AME, ECCE, CCE, and MPSC) and attended several SANS classes (computer forensics and incident response, Windows memory forensics, mobile device security and ethical hacking, reverse engineering malware, smartphone forensics, Mac forensics, securing Windows, and network forensics analysis).

He speaks regularly on digital forensics at different Italian and European universities (Genoa, Milano, Roma, Bolzano, Pescara, Salerno, Campobasso, Camerino, Pavia, Savona, Catania, Lugano, Como, and Modena e Reggio Emilia) and events (DFRWS, SANS European Digital Forensics Summit, Security Summit, IISFA Forum, DEFT Conference, and DFA Open Day). He is a member of CLUSIT, DFA, IISFA, ONIF, and Tech and Law Center, and the author of various articles on scientific publications about digital forensics. More information is available on his LinkedIn profile (`http://www.linkedin.com/in/mattiaepifani`).

Pasquale Stirparo (`@pstirparo`) is currently working as a cyber threat intelligence and incident response engineer at a Fortune 500 company. Prior to this, among other positions, Pasquale has also worked at the Joint Research Centre (JRC) of the European Commission as a digital forensics and mobile security researcher, with particular interest in the security and privacy issues related to mobile device communication protocols, mobile applications, mobile malware, and cybercrime. Since 2016, he has been appointed to the Advisory Group on Internet Security at the European Cyber Crime Center (EC3) of Europol and is an incident handler with the SANS Internet Storm Center (ISC). Pasquale has also been involved in the standardization of Digital Forensics as a contributor (the first in Italy) to the development of the standard "ISO/IEC 27037: Guidelines for identification, collection and/or acquisition and preservation of digital evidence", for which he led the WG ISO27037 for the Italian National Body in 2010.

He is the author of many scientific publications and has also been invited as a speaker at several national and international conferences and seminars on Digital Forensics and as a lecturer on the same subject for the Polytechnic of Milano (CEFRIEL) and the United Nations (UNICRI). Pasquale holds a Ph.D. in Computer Security from the Royal Institute of Technology (KTH) of Stockholm and a M.Sc. in Computer Engineering from the Polytechnic of Torino, and is certified with GCFA, GREM, OPST, OWSE, and ECCE. More information is available on his LinkedIn personal profile (`https://www.linkedin.com/in/pasqualestirparo`).

We would like to thank our families, colleagues, and friends, who supported and encouraged us all the way since the first edition, bearing up with our writing schedules and deadlines. A special thank you goes to Francesco Picasso and Marco Scarito, for continuously engaging and challenging us, and for being a great source of knowledge and improvement. Last but not least, we would like to thank Cellebrite, Oxygen Forensic, and Elcomsoft for providing us demo licenses for the practical demos for this book.

About the Reviewer

Michael Yasumoto is the CEO at Deadbolt Forensics, a digital forensics consulting company located in Portland, Oregon. He is a digital forensics examiner for attorneys and private companies and has testified as an expert witness in court. He is also an instructor for AccessData, teaching mobile forensics to law enforcement, both domestically and abroad.

Michael holds a bachelor's degree in chemistry from the University of Washington and a master's degree in computer science from the George Washington University. Some of his forensic credentials include Certified Computer Examiner (CCE), X-Ways Professional in Evidence Recovery Techniques (X-PERT), EnCase Certified Examiner (EnCE), AccessData Certified Examiner (ACE), Cellebrite Certified Mobile Examiner (CCME), and AccessData Mobile Examiner (AME).

Table of Contents

Preface

This book is a complete discussion of state-of-the-art technology used in identification, acquisition, and forensic analysis of mobile devices with the iOS operating system, and it has been extended and updated to iOS 8 and 9. It is a practical guide that will help investigators understand how to manage scenarios efficiently during their daily work on this type of mobile device. The need for a practical guide in this area arises from the growing popularity of iOS devices and the different scenarios that an investigator may face, according to the type of device, the version of the operating system, and the presence or absence of security systems (code lock, backup password, and so on). The book is divided (conceptually) into four parts. The first part deals with the basic concepts related to methods and guidelines to be followed in the treatment of digital evidence and information specific to an iOS device. The second part covers the basic techniques and tools for acquisition of an iOS device, also through a backup and iCloud. The third part goes deep into the methodology and techniques for analyzing the data. Finally, the fourth part provides an overview of issues related to the analysis of iOS applications and malware. For those who are new to this field, we recommend a sequential reading of the book, since the arguments are processed in the order of the main phases of a forensic investigation (identification, acquisition, and analysis). For the more experienced readers, and for those who routinely deal with this type of device, the book can be considered as a useful tool to evaluate different techniques, depending on the type of case that you have to handle.

What this book covers

`Chapter 1`, *Digital and Mobile Forensics*, is an introduction to the most important concepts and definitions in the field of digital and mobile forensics, and the life cycle of digital evidence, which includes identification, acquisition, analysis, and reporting.

`Chapter 2`, *Introduction to iOS Devices*, contains useful information and references that will help you learn how to identify the various types of device (such as iPhone, iPad, and iPod Touch) with respect to their model and iOS version. It also contains basic information about the filesystem used on a specific kind of device.

`Chapter 3`, *Evidence Acquisition from iDevices*, explains how to acquire data from iOS devices with respect to their model and iOS version, which was introduced in the previous chapter. Physical, logical, and advanced logical acquisitions are discussed, along with the most useful techniques on how to crack or bypass the passcode set by the user. This chapter presents examples of acquisitions realized with various tools, with latest techniques for iOS 8 and iOS 9.

`Chapter 4`, *Evidence Acquisition and Analysis from iTunes Backup*, gives an overview of how to deal with the analysis of an iTunes backup taken from a PC or a Mac, focusing on how to read its content and how to try to attack a protected password set by the user. This chapter also explains how to recover passwords stored in the device when the backup is not protected by a password of its own or when the analyst is able to crack it.

`Chapter 5`, *Evidence Acquisition and Analysis from iCloud*, deals with the case in which the owner is using iCloud to store the device's backup. You will learn how to recover the credentials or the authorization token used to retrieve the information stored in Apple servers.

`Chapter 6`, *Analyzing iOS Devices*, provides a complete set of information regarding how to analyze data stored in the acquired device. Both preinstalled (such as address book, call history, SMS, MMS, and Safari) and third-party applications (such as chat, social network, and cloud storage) are explained, with particular attention to the core artifacts and how to search and recover them.

`Chapter 7`, *Applications and Malware Analysis*, is an introduction to the core concepts and tools used to perform an application assessment from a security point of view. You will also learn how to deal with mobile malware that may be present on jailbroken devices.

`Appendix A`, *References*, is a complete set of references that will help you understand some core concepts explained in the book so that you can go deeper into specific topics.

`Appendix B`, *Tools for iOS Forensics*, is a comprehensive collection of open source, freeware, and commercial tools used to acquire and analyze the content of iOS devices.

`Appendix C`, *Self-test Answers*, contains the answers to the questions asked in the chapters of the book.

What you need for this book

This book is designed to allow you to use different operating platforms (Windows, Mac, and Linux) through freeware, open source software, and commercial software. Many of the examples shown can be replicated using either the software tested by the authors or equivalent solutions that have been mentioned in `Appendix B`, *Tools for iOS Forensics*. Some specific cases require the use of commercial platforms, and among those, we preferred the platforms that we use in our daily work as forensic analysts (such as Cellebrite UFED, Oxygen Forensics, Elcomsoft iOS Forensic Toolkit, and Elcomsoft Phone Breaker). In any case, we were inspired by the principles of ease of use, completeness of information extracted, and the correctness of the presentation of the results by the software. This book is not meant to be a form of advertisement for the aforementioned software in any way, and we encourage you to repeat the tests that are carried out on one operating platform even on other platforms and software applications.

Who this book is for

This book is intended mainly for a technical audience, and more specifically for forensic analysts (or digital investigators) who need to acquire and analyze information from mobile devices running iOS. This book is also useful for computer security experts and penetration testers because it addresses some issues that must be definitely taken into consideration before the deployment of this type of mobile devices in business environments or situations where data security is a necessary condition. Finally, this book can be also of interest for developers of mobile applications, and they can learn what data is stored in these devices where the application is used. Thus, they will be able to improve security.

Conventions

In this book, you will find a number of styles of text that distinguish among different kinds of information. Here are some examples of these styles, and explanations of their meanings. Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows:

"Compile the source file by simply typing the `make` command."

A URL is written as follows: `http://theiphonewiki.com/wiki/UDID`

A pathname is written as follows:
`/private/var/root/Library/Lockdown/data_ark.plist`

Any command-line input or output is written as follows:

```
$ iproxy 2222 22
 $ ssh usb
```

New terms and important words are shown in bold. Words that you see on the screen, for example, in menus or dialog boxes, appear in the text like this: "Clicking the **Next** button moves you to the next screen."

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or disliked. Reader feedback is important for us as it helps us develop titles that you will really get the most out of.

To send us general feedback, simply e-mail feedback@packtpub.com, and mention the book's title in the subject of your message. If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide at www.packtpub.com/authors.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the color images of this book

We also provide you with a PDF file that has color images of the screenshots/diagrams used in this book. The color images will help you better understand the changes in the output. You can download this file from http://www.packtpub.com/sites/default/files/downloads/LearningiOSForensicsSecondEdition_ColorImages.pdf.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you could report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the Errata Submission Form link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website or added to any list of existing errata under the Errata section of that title. To view the previously submitted errata, go to `https://www.packtpub.com/books/content/support` and enter the name of the book in the search field. The required information will appear under the Errata section.

Piracy

Questions

If you have a problem with any aspect of this book, you can contact us at `questions@packtpub.com`, and we will do our best to address the problem.

1 Digital and Mobile Forensics

Not long from now, we would be talking mainly, if not solely, about computer forensics and computer crimes, such as an attacker breaking into a computer network system and stealing data. This would involve two types of offense: unlawful/unauthorized access and data theft. As mobile phones became more popular, the new field of mobile forensics developed.

Nowadays, things have changed radically and are still changing at quite a fast pace as technology evolves. Digital forensics, which includes all disciplines dealing with electronic evidence, is also being applied to common crimes, to those that, at least by definition, are not strictly IT crimes. Today, more than ever, we live in a society that is fully digitalized and people are equipped with all kinds of devices, which have different types of capabilities, but all of them process, store, and transmit information (mainly over the Internet). This means that forensic investigators have to be able to deal with all these devices.

As defined at the first **Digital Forensics Research Workshop** (**DFRWS**) in 2001, digital forensics is:

> *"The use of scientifically derived and proven methods toward the preservation, collection, validation, identification, analysis, interpretation, documentation, and presentation of digital evidence derived from digital sources for the purpose of facilitating or furthering the reconstruction of events found to be criminal, or helping to anticipate unauthorized actions shown to be disruptive to planned operations."*

As Casey asserted (**Casey, 2011**):

> *"In this modern age, it is hard to imagine a crime that does not have a digital dimension."*

Criminals of all kinds use technology to facilitate their offenses, communicate with their peers, recruit other criminals, launder money, commit credit card fraud, gather information on their victims, and so on. This obviously creates new challenges for all the different actors involved, such as attorneys, judges, law enforcement agents, forensic examiners.

Among the cases solved in recent years, there were kidnappings where the kidnapper was caught thanks to a request for ransom sent by e-mail from his mobile phone. There have been many cases of industrial espionage in which unfaithful employees were hiding projects in the memory cards of their smartphones, cases of drug dealing solved thanks to the evidence found in the backup of mobile phones that was on computer, and many other such cases. Even the largest robberies of our time are now being conducted via computer networks.

In this chapter, you will learn the following:

- Definition and principles of mobile forensics
- How to properly handle digital evidence
- The methodology for the identification and preservation of mobile evidence

Mobile forensics

Mobile forensics is a field of study in digital forensics that focuses on mobile devices. Among the different digital forensics fields, mobile forensics is without doubt the fastest growing and evolving area of study, having an impact on many different situations from corporate to criminal investigations and intelligence gathering, which are on the rise. Moreover, the importance of mobile forensics is increasing exponentially due to the continuous fast growth of the mobile market. One of the most interesting peculiarities of mobile forensics is that mobile devices, particularly mobile phones, usually belong to a single individual, while this is not always the case with a computer that may be shared among employees of a company or members of a family. For this reason, the analysis of mobile phones gives access to plenty of personal information.

Another important and interesting aspect that comes with mobile forensics, which is both challenging and frustrating at the same time for the analyst, is the multitude of different device models and the customized flavors of their operating systems available on the market. This makes it very difficult to have a single solution (either a tool or process) to address them all.

Just think of all the applications people have installed on their smartphones: IM clients, web browsers, social network clients, password managers, navigation systems, and much more, other than the classic default ones, such as an address book, which can provide a lot more information other than just the phone number for each contact that has been saved. Moreover, syncing such devices with a computer has become a very easy and smooth process, and all user activities, schedules, to-do lists, and everything else is stored inside a smartphone. Aren't these enough to profile a person and reconstruct all their recent activities, other than building the network of contacts?

Finally, in addition to a variety of smartphones and operating systems, such as Apple iOS, Google Android, Microsoft Windows Phone, and Blackberry OS, there is a massive number of so-called feature phones that use older mobile OS systems.

Therefore, it's pretty clear that when talking about mobile/smartphone forensics, there is so much more than just printouts of phone calls. In fact, with a complete examination, we can retrieve SMSes/MMSes, pictures, videos, installed applications, e-mails, geolocation data, and so on-both present and deleted information.

Digital evidence

As mentioned earlier, on one hand the increasing involvement of mobile devices in digital forensics cases has brought a whole new series of challenges and complexities. However, on the other hand, this has also resulted in a much greater amount of evidence from criminals that it is now being used to reconstruct their activities with a more comprehensive level of detail. Moreover, while classical physical evidence may be destroyed, digital evidence, most of the time, leaves traces.

Over the years, there have been several definitions of what digital evidence actually is, some of them focusing particularly on the evidentiary aspects of proof to be used in court, such as the one proposed by the **Standard Working Group on Digital Evidence** (**SWGDE**), stating that:

> *"Digital evidence is any information of probative value that is either stored or transmitted in a digital form."*

The definition proposed by the **International Organization of Computer Evidence** (**IOCE**) states:

> *"Digital evidence is information stored or transmitted in binary form that may be relied on in court."*

The definition given by E. Casey (**Casey, 2000**), refers to digital evidence as:

> *"Physical objects that can establish that a crime has been committed, can provide a link between a crime and its victim, or can provide a link between a crime and its perpetrator."*

While all of these are correct, as previously said, all of these definitions focus mostly on proofs and tend to disregard data that is extremely useful for an investigation.

For this reason, and for the purpose of this book, we will refer to the definition given by Carrier (**Carrier, 2006**), where digital evidence is defined as:

> *"Digital data that supports or refutes a hypothesis about digital events or the state of digital data."*

This definition is a more general one, but better matches the current state of digital evidence and its value within the entire investigation process.

Also from a standardization point of view, there have been, and still are, many attempts to define guidelines and best practices for digital forensics on how to handle digital evidence. Other than the several guidelines and special publications from NIST, there is a standard from ISO/IEC that was released in 2012, the ISO 27037 guidelines for identification, collection and/or acquisition, and preservation of digital evidence, which is not specific to mobile forensics but is related to digital forensics in general, aiming to build a standard procedure for collecting and handling digital evidence, which will be legally recognized and accepted in court in different countries. This is a really important goal if you consider the lack of borders in the Internet era, particularly when it comes to digital crimes, where illicit actions can be perpetrated by attackers from anywhere in the world. Refer to `Appendix A`, *References* for a more detailed list of standards and best practices.

Handling of mobile evidence

In order to be useful not only in court but also during the entire investigation phase, digital evidence must be collected, preserved, and analyzed in a forensically sound manner. This means that each step, from the identification to the reporting, has to be carefully and strictly followed. Historically, we are used to referring to a methodology as forensically sound if, and only if, it would imply the original source of evidence remains unmodified and unaltered. This was mostly true when talking about classical computer forensics, in scenarios where the forensic practitioner found the computer switched off or had to deal with external hard drives, although not completely true even in these situations. However, since the rise of live forensics, this concept has become more and more untrue. In fact, methods and tools for acquiring memory from live systems inevitably alter, even if just a little bit, the target system they are run on. The advent of mobile forensics stresses this concept even more, because mobile devices, and smartphones in particular, are networked devices that continuously exchange data through several communication protocols, such as GSM/CDMA, Wi-Fi, Bluetooth, and so on. Moreover, in order to acquire a mobile device, forensic practitioners need to have some degree of interaction with the device. Based on the type, a smartphone can need more or less interaction, altering in this way the original state of the device.

All of this does not mean that preservation of the source evidence is useless, but that it is nearly impossible in the field of mobile devices. Therefore, it becomes a matter of extreme importance to thoroughly document every step taken during the collection, preservation, and acquisition phases. Using this approach, forensic practitioners will be able to demonstrate that they have been as unintrusive as possible. As Casey states (**Casey, 2011**):

> *"One of the keys to forensic soundness is documentation. A solid case is built on supporting documentation that reports on where the evidence originated and how it was handled. From a forensic standpoint, the acquisition process should change the original evidence as little as possible and any changes should be documented and assessed in the context of the final analytical results."*

When in the presence of mobile devices to be collected, it is good practice for the forensic practitioner to consider the following points:

- Take note of the current location where the device has been found.
- Report the device status (switched on or off, broken screen, and so on).
- Report date, time, and other information visible on the screen if the device is switched on, for example by taking a picture of the screen.
- Look very carefully for the presence of memory cards. Although it is not the case with iOS devices, generally many mobile phones have a slot for an external memory card, where pictures, chat databases, and many other types of user data are usually stored.
- Look very carefully for the presence of cables related to the mobile phone that is being collected, especially if you don't have a full set of cables in your lab. Many mobile phones have their own cables to connect to the computer and to recharge the battery.
- Search for the original**Subscriber Identity Module** (**SIM**) package, because that is where the PIN and **PIN unblocking key** (**PUK**) codes are written.
- Take pictures of every item before collection.

Modifications to mobile devices can happen not only because of interaction with the forensic practitioner, but also due to interaction with the network, voluntarily or not. In fact, digital evidence in mobile devices can be lost completely as they are susceptible to being overwritten by new data, for example with the smartphone receiving an SMS while it is being collected, thus overwriting possible evidence previously stored in the same area of memory as the newly arrived SMS, or upon receiving a remote wiping command over a wireless network. Most of today's smartphones and iOS devices can be configured to be completely wiped remotely.

From a real case:
While searching inside the house of a person under investigation, law enforcement agents found and seized, among other things, computers and a smartphone. After cataloguing and documenting everything, they put all the material into boxes to bring them back to the laboratory. Once back in their laboratory, when acquiring the smart phone in order to proceed with the forensics analysis, they noticed the smartphone was *empty* and appeared to be *brand new*. The owner had wiped it remotely.

Therefore, isolating the mobile device from all radio networks is a fundamental step in the process of preservation of evidence. There are several ways to achieve this, all with their own pros and cons, as follows:

- **Airplane mode**: Enabling Airplane mode on a device requires some sort of interaction, which may pose some risks of modification by the forensic practitioner. This is one of the best possible options since it implies that all wireless communication chips are switched off. In this case, it is always good to document the action taken with pictures and/or videos. Normally, this is possible only if the phone is not password-protected or the password is known. However, for devices with iOS 7 or higher, it is also possible to enable airplane mode by lifting the dock from the bottom, where there will be a button with the shape of a plane. This is possible only if the **Access on Lock Screen** option is enabled from **Settings** | **Control Center**.
- **Faraday's bag**: This item is a sort of envelope made of conducting material, which blocks out static electric fields and electromagnetic radiation completely isolating the device from communicating with external networks. It is based, as the name suggests, on Faraday's law. This is the most common solution, particularly useful when the device is being carried from the crime scene to the lab after seizure. However, the use of Faraday's bag will make the phone continuously search for a network, which will cause the battery to quickly drain. Unfortunately, it is also risky to plug the phone to a power cable outside that will go inside the bag, because this may act as antenna. Moreover, it is important to keep in mind that when you remove the phone from the bag (once arrived in the lab) it will again be exposed to the network. So, you would need either a shielded lab environment or a Faraday solution that would allow you to access the phone while it is still inside the shielded container, without the need for external power cables.
- **Jamming**: A jammer is used to prevent a wireless device from communicating by sending out radio waves along the same frequencies as that device. In our case, it would jam the GSM/UMTS/LTE frequencies that mobile phones use to connect with cellular base stations to send/receive data. Be aware that this practice may be considered illegal in some countries, since it will also interfere with any other mobile device in the range of the jammer, disrupting their communications too.
- **Switching off the device**: This is a very risky practice because it may activate authentication mechanisms, such as PIN codes or passcodes, that are not available to the forensic practitioner, or other encryption mechanisms that carry with the risk of delaying or even blocking the acquisition of the mobile device.

- **Removing the SIM card:** In most mobile devices, this operation implies removing the battery and therefore all the risks and consequences we just mentioned regarding switching off the device; however, in iOS devices this task is quite straightforward and easy, and it does not imply removing the battery (in iOS devices this is not possible). Moreover, SIM cards can have PIN protection enabled; removing it from the phone may lock the SIM card, preventing its content from being displayed. However, bear in mind that removing the SIM card will isolate the device only from the cellular network, while other networks, such as Wi-Fi or Bluetooth, may still be active and therefore need to be addressed.

- The following image shows a SIM card extracted from an iPhone with just a clip; image taken from `http://www.maclife.com/`:

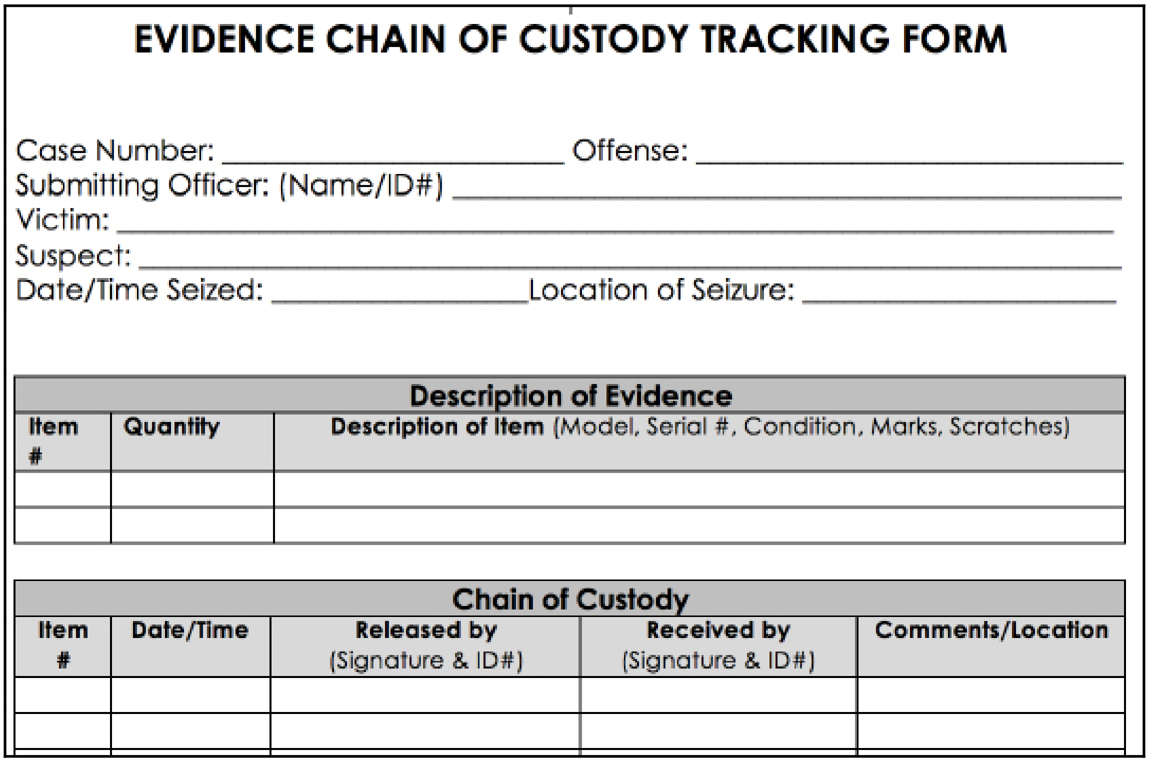
Preservation of evidence

Talking about the documentation and preservation of digital evidence, one of the most important steps is the correct and comprehensive compilation of the **chain of custody**. The purpose of this document is twofold: on one hand, to keep the record of each person who handled the evidence, enabling the identification of access and movement of potential digital evidence at any given point in time; and on the other hand, to maintain documentation demonstrating that the digital evidence has not been altered since it was collected while passing through the hands of the several analysts listed in the document.

Therefore, some of the information that the chain of custody should contain is as follows:

- A unique evidence identifier
- Who accessed the evidence and the time and location it took place
- Who checked the evidence in and out of the evidence preservation facility and when
- Reasons why the evidence was checked out
- The hash values of the evidence in order to prove that it has not been tampered with since it was last assigned to the previous person listed in the chain of custody
- Although the forensics investigation must never be performed directly on the original device/file, the chain of custody should mention if any unavoidable changes to the potential digital evidence had to be performed and the justification for the introduction of such changes, as well as the name of the individual responsible

- The following image shows a sample of chain of custody proposed by NIST:



EVIDENCE CHAIN OF CUSTODY TRACKING FORM

Case Number: ______________________ Offense: ____________________________
Submitting Officer: (Name/ID#) ____
Victim: ____
Suspect: ____
Date/Time Seized: _________________Location of Seizure: ____________________

Description of Evidence		
Item #	**Quantity**	**Description of Item** (Model, Serial #, Condition, Marks, Scratches)

Chain of Custody				
Item #	**Date/Time**	**Released by** (Signature & ID#)	**Received by** (Signature & ID#)	**Comments/Location**

Acquisition of evidence

Especially in mobile forensics, where visible information may be more volatile, and also in classical computer forensics, sometimes there may be an urgency to acquire the data available. Information may vanish before it is possible to isolate or properly handle the device. In such cases, effective on-scene triage processes and tools may preserve evidence that would otherwise be lost. Such processes may include taking immediate pictures or videos recording the screen of the device before proceeding with any other type of operation.

Having said that, once the mobile device has been handled correctly, forensic practitioners may proceed with the acquisition of the evidence from the device. In mobile forensics, and particularly for iOS devices, there are the following three different types of possible acquisition:

- **Physical**: This is the optimal and most desired option. It involves a physical acquisition consisting of an exact *bit-to-bit* copy of the device. This is the most comprehensive option since it also allows you to recover potentially deleted files.
- **FileSystem**: This is the second-best option when physical acquisition is not possible for whatever reason. This type of acquisition lets the forensic practitioner extract all the files visible at the filesystem level. In this way, it will be possible to analyze all active files that would be visible by browsing the filesystem, but it will not be possible to recover potentially deleted files.
- **Logical**: With this type of acquisition, it is possible to extract part of the filesystem. It consists recovering the available data by performing the backup of the device, via iTunes in the case of iOS devices. Unfortunately, with iOS, a logical/backup acquisition does not extract important files such as e-mails, geolocation databases, the app cache folder, and so on. Although it is the least comprehensive of the three, sometimes this may be the only option available.

The preceding three acquisition methods are the main methods for acquiring an iOS device; we will see more about this in detail later. In the following chapters, we will dive deep into each of the different methodologies, explaining how to behave in every possible situation and we will see most of the different tools available for performing the acquisition and further analysis of a physical filesystem and logical acquisition.

Mobile forensics, however, may also include the need to adopt some offensive security techniques. Depending on the device model and iOS version, in order to make a physical acquisition we may need to jailbreak the device, hopefully with a tethered technique so that modifications will not be persistent on the device and it will be restored once restarted. Even in cases when we can only perform an untethered jailbreak, such modifications will affect only the iOS device system partition, leaving the user partition unchanged and therefore the evidence preserved.

Another offensive technique we may need to use is password cracking. As we will see later, often we may find ourselves faced with a password-protected device. Dependent on the model and iOS version, it may be possible to perform brute force attacks on the passcode set by the user.

These more invasive techniques will need to be fully documented in the final report, detailing methodology, techniques, and tools used. It is very important, especially because of their invasiveness, to know well the tools and techniques used in order to be able to explain what and where the modifications have happened, and why they did not alter the evidence to the point of compromising it. Good reporting is key.

Evidence integrity

It has been mentioned already multiple times that when handling mobile devices, it is basically impossible not to interact with the device and therefore alter, to some extent, its current status. However, this does not mean that in mobile forensics there is no need or reason to put in place mechanisms of evidence integrity. In fact, once the acquisition has been completed, there must be some integrity verification mechanism in place for the data that has been extracted from the mobile device, be it an iTunes backup, a full physical acquisition, or simply a single file. In digital forensics, such a process of verifying the integrity of digital evidence is completed by comparing the digital fingerprint of the evidence taken at the time of acquisition with the digital fingerprint of the evidence in the current state. Such a fingerprint is also known as a **hash value** or **message digest**. Hashing functions are specific one-way mathematical functions that, given any input of arbitrary length, will produce as a result an output of a given fixed length. The same input will always produce the same output. This means that even if a single bit is changed, the new hash value will be completely different. The following table shows how simply by modifying only the case of two characters in the same sentence, the resulting hash value is completely different:

Input value	**MD5 output**
ios Forensics book	`9effa61083b07a164c5471d020 fa4306`
iOS Forensics book	`e6196e1b4f0d1535244eaab534 428542`

The two most common algorithms used to calculate hash values are MD5 and SHA-1. The MD5 algorithm produces an output of 128 bits, while the SHA-1 algorithm produces an output of 160 bits. The other important characteristic of these type of algorithms is that, theoretically, it should be computationally infeasible to be able to generate two different inputs that will produce the same hash value, as well as to generate a specific message that will match a given hash value. This problem is known as **collision**. It has been demonstrated in several studies that it is possible to create two different files with the same hash values in both MD5 and SHA-1 cases. However, it has to be specified that this is true under specific controlled conditions that are unlikely to happen in reality. In any case, such collisions do not invalidate the use of these hash algorithms for integrity verification. Since it is basically impossible to produce two files that have the same MD5 and SHA-1 hash value (or in general two hash values generated by two different independent algorithms), it is a good practice to generate both MD5 and SHA-1 hash values for each piece of digital evidence produced or collected.

SIM cards

When conducting forensic examinations of mobile devices, it is also important to acquire and analyze the content of the associated SIM cards. The SIM card is a type of smart card that allows a mobile device to connect to a cellular network through the cryptographic keys embedded in the SIM card itself. The SIM card is mainly characterized by the following two different codes that can be retrieved:

- **Integrated Circuit Card Identification** (**ICCID**): This code is a 20-digit code that internationally and univocally identifies each SIM card
- **International Mobile Subscriber Identity** (**IMSI**): This is a unique 15-digit number (in some places, as in South Africa, it's 14), which univocally identifies a user inside the mobile network

Although it is not the case with iOS devices, there might be multiple SIM cards that an individual uses with the same device for different purposes, since some mobile devices support functioning with dual SIM cards.

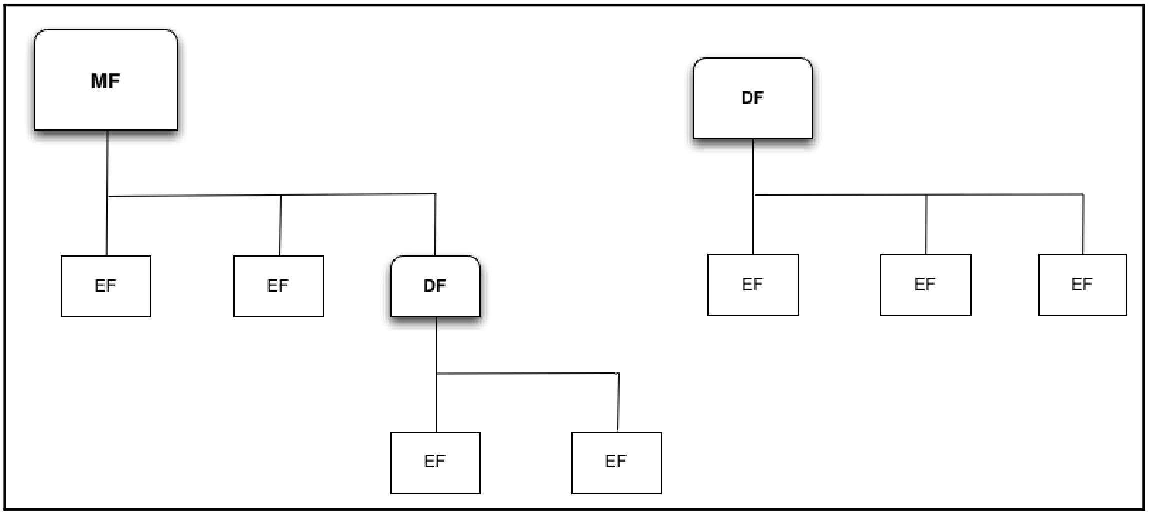
In addition, the storage capacity and utilization of SIM cards has increased a lot and may contain a large amount of relevant information. Just to give you an idea of the amount of data that it could be possible to store (or hide) inside a SIM card, consider that inside a 128 KB standard SIM card, it is possible to write up to 17 KB of data. The entire **United States Declaration of Independence** takes just 11 KB.

Some of the useful information to recover from a SIM card may be the list of incoming/outgoing phone calls, contacts information, the SMS content, for which it is possible to recover even those that have been deleted, and the location of the last cell to which the device was connected.

Looking into the details of the SIM card (Gubian, 2007), it is possible to see the hierarchical n-ary structure of the filesystem that has three different kinds of files, with the content of each file defined in the following GSM technical specification (GSM 11.11):

- **3F = Master File (MF)**: Its structure is composed of just a header and it is the root of the filesystem in the SIM card. Its address, which is the offset for every other file, is `3F00`.
- **7F/5F = 7F is a Dedicated File (DF)**: As for the MF, its structure is composed just of a header plus EFs. A DF can be compared to a normal folder in our PC.
- **2F = Elementary file (EF) under the master file and 6F/4F = 4F is an Elementary file under a dedicated file 6F**: The structure of the elementary file is composed of a header plus a body, which represents itself (for example, the SMS).

The following diagram gives an example of this hierarchical structure (the filesystem structure of a SIM):



The GSM technical specification already provides some files with common names. Some of the most interesting folders among the standard ones may be `3F00:7F10` (`DF_TELECOM`), which contains service-related information and some user data such as SMS, and `3F00:7F20` (`DF_GSM`), where information related to the GSM network is stored. The ICCID and IMSI mentioned previously can be found at `3F00:2FE2`, named `EF_ICCID`, and `3F00:7F20:6F07`, named `EF_IMSI`. The following table presents some of the well-known sources of information that can be found inside a SIM card and their respective locations:

Description	**Location**
SMS	`7F10:6F3C`
MSISDN	`7F10:6F40`
Last Number Dialed (**LND**)	`7F10:6F44`
Abbreviated Dialling Numbers (**ADN**)	`7F10:6F3A`
IMSI	`7F10:6F07`

In a SIM card, the access to **each file** (**EF**) is ruled by a certain number of privilege levels, which allow or deny certain actions according to the role the user has (which is given by the privilege level). Some of the useful privileges are `ALWays`, `CHV1`, and `CHV2`. Those are the privileges that allow the owner of the SIM card (or anyway the user who knows the codes) to access and modify the content of such files. For instance, any file that has one of these privileges related to the `UPDATE` command, allows those that know such codes (`CHV1`/`CHV2`) to modify the information inside that file.

The following table summarizes the access conditions for SIM cards:

Level	**Access conditions**
0	ALWays
1	CHV1
2	CHV2
3	Reserved for GSM future use
4 to 14	ADM
15	NEVer

SIM security

Other than ICCID and IMSI, which are mainly related to the SIM card itself, the other two important codes useful to know (actually, almost indispensable) when conducting an analysis are the PIN code and the PUK code. The PIN code is used to authenticate the user to the system, while the PUK code is used to unlock the SIM card after three incorrect attempts to insert the PIN code. Therefore, brute forcing the PIN is generally ineffective, because three failed PIN attempts will result in the SIM being locked.

Fortunately, SIM cards have a PUK and many **network service providers** (**NSP**) can provide it to law enforcements with a proper legal authorization signed by a judge (warrant), to get around the PIN or to access a locked SIM card.

If an incorrect PUK code is inserted 10 times, the SIM will block itself permanently, making its content completely inaccessible. This is something to keep in mind before starting a brute force attacks against those two codes.

Summary

In this chapter, we gave a general introduction to digital forensics for those relatively new to this area of study and a good recap to those already in the field, keeping the mobile forensics field specifically in mind. We have seen what digital evidence is and how it should be handled, presenting several techniques to isolate the mobile device from the network. You should always remember the importance of documenting any action taken (chain of custody, final report, and so on) and to put in place mechanisms to verify the integrity of the evidence (hash values). We also talked about the different acquisition techniques for iOS devices, anticipating some terms and technologies that will be covered in full detail in the following chapters of this book, from A to Z. Last but not least, we talked about the SIM card, how it is structured, and what type of useful information we can expect to find inside it.

In the next chapter, we will start focusing purely on the mobile forensics of Apple devices. In particular, you will have an introduction to the iOS devices, OS, and the filesystem.

Self-test questions

Q1. What is the best option to isolate a mobile device before acquisition?

1. Jammer
2. Faraday's bag
3. Airplane mode
4. Switch off the device

Q2. What is the most comprehensive acquisition method?

1. Logical
2. Advanced logical
3. Filesystem
4. Physical

Q3. What is the code that internationally and univocally identifies each SIM card called?

1. IMSI
2. ICCID
3. PUK
4. GSM

Q4. How many PUK attempts can we make before the SIM card becomes completely inaccessible?

1. 3
2. 5
3. 10
4. 15

2
Introduction to iOS Devices

Apple's history with mobile devices began in 1993, with the Newton project and the production of the first PDA, called Message Pad. These PDAs were equipped with an ARM processor, a touch screen, and handwriting recognition and were based on an operating system called Newton OS. The project was closed in 1997 when Steve Jobs returned to the helm of Apple.

The real revolution and the starting point for the current success of iPhone and iPad has been the creation of the iPod devices and the iTunes software. These devices have completely changed the way each of us enjoys listening to music or multimedia content. Released in 2001, they were produced in different shapes, size, and capacity, from the first model presented on October 23, 2001 based on an ARM processor and 5 GB of capacity (Steve Jobs described it as, *a product that put, 1000 songs in your pocket*). The traditional iPods (iPod Classic, iPod Mini, iPod Nano, and iPod Shuffle) have already been the subject of previous publications in relation to the forensics acquisition and analysis of their content. We limit ourselves here to recalling that, from the point of view of the acquisition, an iPod can be treated as an external hard drive and acquired with traditional techniques of disk imaging. Regarding the filesystem it uses, it is either HFS+ or FAT32, depending on if it has been connected for the first time to a Mac or a PC. The audio files are stored in a hidden folder (typically in `\iPod_Control\Music\`) and organized inside a database. Other information that can be found within this type of device are contacts, calendar, and any other file in digital format that the user has decided to copy on the device, in a totally analogous way to an external device (pen drive, hard disk, and so on.). Using traditional techniques, it is possible to perform file carving, to recover deleted files and to create a timeline of the device usage.

Apple entered mobile communication systems in 2005, through a joint venture with Motorola and the production of the phone Rokr, the world's first integrated cellphone with iTunes. The device was not a great success, to the point that the project was shut down after only one year of life. However, this device has laid the foundations for the emergence of the iPhone.

The purpose of this chapter is to introduce the basic aspects of the forensic analysis of an iOS device. In the first part, the different types and models of the Apple devices are discussed, with an indication of the methodologies and techniques to accurately identify the model that you have to acquire. The second part analyzes the fundamental principles of the operating system (types, versions, and so on) and the type and structure of the filesystem used on these devices. In particular, this chapter will focus on the following:

- Types of iOS devices
- iOS device connectors
- iOS operating system
- iDevice identification
- iOS filesystem

Types of iOS device

According to the commonly used definition, an iOS device is a device that uses the iOS operating system. Currently, we have four types of devices: iPhone, iPad, iPad mini, and iPod touch. Furthermore, Apple TV use Apple TV Software and tvOS and Apple Watch uses watchOS: both operating systems are based on iOS, therefore they are a sort of hybrid iOS device.

iPhone versions

The most famous iDevice is certainly the iPhone, which has caused a complete revolution in the concept of cellphones, being based on a multi-touch screen, a virtual keyboard, and few physical buttons (the Home, Volume, Power on/off, and Ringer/Vibration buttons).

iPhone (first model)

The first model of the iPhone, known simply as **iPhone**, was presented by Steve Jobs on January 9, 2007 and sold from June 29 of the same year.

It is equipped with a S5L8900 ARM processor at 620 MHz (underclocked to 412 MHz), 128 MB of RAM, and it uses a cellular connection type quad band GSM/GPRS/EDGE (850/900/1800/1900 MHz), as well as supporting Wi-Fi connectivity 802.11 b/g and Bluetooth 2.0 + EDR (information on how Bluetooth is implemented is available at `http://support.apple.com/kb/HT3647`).

It is equipped with a 30-pin dock connector (refer to `https://en.wikipedia.org/wiki/Dock_connector`).

The phone is identified by the model number A1203 and the hardware string iPhone1,1.

It was originally equipped with a 4 GB internal memory, which was subsequently increased in two distinct patterns in 8 GB and 16 GB.

Regarding the software, it originally used an ancestor of the iOS operating system, known as iPhone OS 1.0. The latest supported version is iPhone OS 3.1.3. Besides the traditional means of communication (internal phone book, calls, SMS, MMS, and so on), it integrates web browsing, e-mail, camera, iTunes, and YouTube. At the time of distribution of the iPhone 2G, the App Store did not exist and this is why Apple made Web Apps available, divided into categories (for example, Calculate, Entertainment, Games, and so on). These applications did not save information in the device and only created a shortcut icon for the application that was executed directly from the Internet. The production of these devices was discontinued in September 2007 (4 GB model) and in June 2008 (8 GB and 16 GB models).

iPhone 3G

The second model produced by Apple, marketed from July 11, 2008, was known as **iPhone 3G**, since it added support for the 3G cellular network. Similar to its predecessor, iPhone 3G was equipped with a S5L8900 ARM processor and 128 MB of RAM. In addition to support for the 3G network (UMTS/HSDPA up to 3.6 Mbit/s at 850, 1900, and 2100 MHz), the main innovation in the hardware was the presence of a GPS chip, which is used for geolocation services. The phone was identified by the model number A1241 (or A1324 for devices sold in China) and the string iPhone1,2. The device has been manufactured with 8 or 16 GB memory. Regarding its software, it originally used iPhone OS 2.0. The latest supported version is iOS 4.2.1. With the basic version of the iPhone 3G, the App Store appeared for the first time: it allowed users to download and install on the device applications from a centralized repository. The production of these devices was discontinued in June 2010.

iPhone 3GS

The third model produced by Apple, marketed from June 19, 2009, was known as **iPhone 3GS**. While it was aesthetically very similar to its predecessors, iPhone 3GS was equipped with a S5L8920 833 MHz ARM processor (underclocked to 600 MHz) and 256 MB of RAM and provided better performance compared to the previous versions. The main changes were the use of a more accurate GPS chip, support for additional data networks (HSDPA 7.2 Mbit/s), a 3 MP camera with the ability to shoot videos in VGA format (480p), a digital compass, and the voice control feature. From the point of view of the forensic analysis, it is interesting to highlight that starting from this model, it is possible to geotag images, making it possible for an investigator to identify the place where a picture was taken. The phone is identified by the model number A1303 (or A1325 for devices sold in China) and the string iPhone2,1. The device has been manufactured with 8, 16, or 32 GB memory. Regarding its software, it originally used iPhone OS 3.0. The latest supported version was iOS 6.1.6. The production of these devices was discontinued in September 2012.

iPhone 4

The fourth model produced by Apple, marketed from June 21, 2010, was known as **iPhone 4**. It was a completely renewed device compared to the previous iPhone models, both in appearance and functionality. The device was more squared in its aesthetic form and presented several hardware improvements, such as an Apple A4 S5L8930 1 GHz processor (underclocked to 800 MHz), 512 MB of RAM, a 5 MP camera with the ability to shoot videos in HD (720p), and a 3-axis gyroscope. The phone is identified by two model numbers: A1332 (GSM model) and A1349 (CDMA model) and by three strings: iPhone3,1; iPhone3,2; and iPhone3,3. The device has been manufactured with 8, 16, or 32 GB memory. Regarding its software, it originally used iOS 4.0, which is the first version with the new name. The latest supported version was iOS 7.1.2. The production of these devices was discontinued in September 2013.

iPhone 4s

The fifth model produced by Apple, marketed from October 4, 2011, was known as **iPhone 4s**. It was aesthetically very similar to iPhone 4, except for the presence of two cuts on the upper part of both sides. The new hardware consisted of an Apple A5 S5L8940 1 GHz processor (underclocked to 800 MHz), 512 MB of RAM, support for HSPA+ up to 14.4 Mbit/s, and an 8 MP rear camera with the ability to shoot videos in HD (1080p). The phone is identified by the model number A1387 (or A1431 for devices distributed in China) and the string iPhone4,1. The device has been manufactured with 8, 16, 32, or 64 GB memory. Regarding its software, it originally used iOS 5.0. Currently, iPhone 4s is supported by the latest available version (iOS 9.3.4), but it will not be supported by iOS 10. The production of these devices was discontinued in September 2014.

iPhone 5

The sixth model produced by Apple, marketed from September 21, 2012, was known as **iPhone 5**. It used an Apple A6 S5L8950 processor 1.3 GHz, 1 GB of RAM, and it supported HSPA+ and LTE cellular networks. It was also equipped with a 1.2 MP front camera for pictures and video up to 720p HD quality. It is the first device in the series with a 4 inch screen. It is the first model equipped with the 8-pin Lightning connector (refer to `https://en.wikipedia.org/wiki/Lightning_%28connector%29`). The phone is identified by three model numbers: A1428 (GSM model), A1429 (GSM and CDMA model), and A1442 (CDMA model for China) and by two strings: iPhone5,1 (USA version with LTE support) and iPhone5,2 (other countries). The device has been manufactured with 16, 32, or 64 GB memory. Regarding its software, it originally used iOS 6.0. Currently, iPhone 5 is supported by the latest available version (iOS 9.3.4) and it will be supported by iOS 10. The production of these devices was discontinued in September 2013.

iPhone 5c

The seventh model produced by Apple, marketed from September 20, 2013, was known as **iPhone 5c**. It used the same processor and the same amount of RAM as the iPhone 5 model, from which it differed in an LTE network support extended to the whole world and a more powerful battery. The phone is identified by six model numbers: A1516 (China), A1526 (China), A1532 (North American model), A1456 (the U.S. and Japanese model), A1507 (Europe), and A1529 (Asia and Oceania) and by two strings: iPhone5,3 and iPhone5,4. The device has been manufactured with 8, 16, or 32 GB memory. Regarding its software, it originally used iOS 7.0. Currently, iPhone 5c is supported by the latest available version (iOS 9.3.4) and it will be supported by iOS 10. The production of these devices was discontinued in September 2015.

iPhone 5s

The eighth model produced by Apple, marketed from September 20, 2013, was known as **iPhone 5s**. It used an Apple A7 S5L8960 processor 1.3 GHz, 1 GB of RAM, an 8 MP camera improved in comparison with the previous versions and the biometric authentication system based on fingerprints, called Touch ID. It also had a motion coprocessor Apple M7. The phone is identified by six model numbers: A1518 (China), A1528 (China), A1533 (North American model), A1453 (the U.S. and Japanese model), A1457 (Europe), and A1530 (Asia and Oceania) and by two strings: iPhone6,1 and iPhone6,2. The device was manufactured with 16, 32, or 64 GB memory. Regarding its software, it originally used iOS 7.0. Currently, iPhone 5s is supported by the latest available version (iOS 9. 3.4) and it will be supported by iOS 10. The production of these devices was discontinued in March 2016.

iPhone 6

The ninth model produced by Apple, marketed from September 19, 2014, is known as **iPhone 6**. It uses an Apple A8 T7000 processor 1.38 GHz with 1 GB of RAM. It is the first model that supports NFC communications. It has also a motion coprocessor Apple M8. The phone is identified by three model numbers: A1549 (North America), A1586 (Global), and A1589 (China) and by the string iPhone7,2. The device has been manufactured with 16, 64, or 128 GB memory. Regarding its software, it originally used iOS 8.0. Currently, iPhone 6 is supported by the latest available version (iOS 9. 3.4) and it will be supported by iOS 10. The production of this device is still active at the date of the publication of this book.

iPhone 6 Plus

The tenth model produced by Apple, marketed from September 19, 2014, is known as **iPhone 6 Plus.** It uses an Apple A8 T7000 processor 1.38 GHz with 1 GB of RAM. It has also a motion coprocessor Apple M8. The phone is identified by three model numbers: A1522 (North America), A1524 (Global), and A193 (China) and by the string iPhone7,1. The device has been manufactured with 16, 64, or 128 GB memory. Regarding its software, it originally used iOS 8.0. Currently, iPhone 6 Plus is supported by the latest available version (iOS 9. 3.4) and it will be supported by iOS 10. The production of this device is still active at the date of the publication of this book.

iPhone 6s

The eleventh model produced by Apple, marketed from September 25, 2015, is known as **iPhone 6s**. It uses an Apple A9 S8000 (Samsung) or S8003 (TSMC) processor 1.85 GHz with 2 GB of RAM. It has also a motion coprocessor Apple M9. The phone is identified by three model numbers: A1633 (North America), A1688 (Global), and A1699 (China) and by the string iPhone8,1. The device has been manufactured with 16, 64, or 128 GB memory. Regarding its software, it originally used iOS 9.0. Currently, iPhone 6s is supported by the latest available version (iOS 9. 3.4) and it will be supported by iOS 10. The production of this device is still active at the date of the publication of this book.

iPhone 6s Plus

The twelfth model produced by Apple, marketed from September 25, 2015, is known as **iPhone 6s Plus**. It uses an Apple A9 S8000 (Samsung) or S8003 (TSMC) processor 1.85 GHz with 2 GB of RAM. It has also a motion coprocessor Apple M9. The phone is identified by three model numbers: A1634 (North America), A1687 (Global), and A1699 (China) and by the string iPhone8,2. The device has been manufactured with 16, 64, or 128 GB memory. Regarding its software, it originally used iOS 9.0. Currently, iPhone 6s Plus is supported by the latest available version (iOS 9.3.4) and it will be supported by iOS 10. The production of this device is still active at the date of the publication of this book.

iPhone SE

The thirteenth model produced by Apple, marketed from March 31, 2016, is known as **iPhone SE**. It uses an Apple A9 S8000 (Samsung) or S8003 (TSMC) processor 1.85 GHz with 2 GB of RAM. It has also a motion coprocessor Apple M9. The phone is identified by three model numbers: A1662 (North America), A1723 (Global), and A1724 (China) and by the string iPhone8,4. The device has been manufactured with 16 or 64 GB memory. Regarding its software, it originally used iOS 9.3. Currently, iPhone SE is supported by the latest available version (iOS 9.3.4) and it will be supported by iOS 10. The production of this device is still active at the date of the publication of this book.

iPad

After the success of the iPhone, Apple carried out the project of designing and producing a larger version, which for the first time gave substance to Steve Jobs' idea in 1983:

> *"Apple's strategy is really simple. What we want to do is we want to put an incredibly great computer in a book that you can carry around with you."*

After the launch of the first iPad, Jobs said that Apple had begun to develop the iPad tablet before the iPhone, but had subsequently decided to concentrate its efforts in the development of the iPhone.

iPad (first model)

The first model of iPad, known simply as **iPad** (or **iPad first generation**), was presented by Steve Jobs on January 27, 2010 and sold from April 3 of the same year. It was equipped with a 1 GHz S5L8930 ARM processor (known as the Apple A4) and 256 MB of RAM. As with all the iPad device family, there are two distinct versions: the first one was equipped only with Wi-Fi 802.11 a/b/g/n connection, while the second one was also equipped with 3G UMTS/HSDPA/EDGE and a GPS. The two models are identified by model number A1219 (Wi-Fi only) and A1337 (Wi-Fi and 3G), while both models are characterized by the string iPad1,1. From a software point of view, it originally used the iPhone OS 3.2. The latest supported version is iOS 5.1.1. The production of these devices was discontinued in March 2011.

iPad 2

The second model of the iPad, known as **iPad 2**, has been marketed since March 11, 2011. It was equipped with a 1 GHz S5L8940 ARM processor (known as Apple A5) and 512 MB of RAM. Compared to the previous version, Apple introduced a front and a rear camera of 0.75 MP. It was produced in three models: Wi-Fi only (model number A1395), Wi-Fi and GSM (model number A1396), and Wi-Fi and CDMA (model number A1397). There are four hardware strings: iPad2,1 (Wi-Fi only); iPad2,2 (Wi-Fi and GSM); iPad2,3 (CDMA and Wi-Fi); and iPad2,4 (Wi-Fi only with S5L8942 processor, known as A5 Rev. A). All versions were produced with 16, 32, or 64 GB memory. Regarding its software, it originally used iOS 4.3. Currently, iPad 2 is supported by the latest version available (iOS 9.3.4), but it will not be supported by iOS 10. The production of these devices was discontinued in March 2014.

iPad 3 (the new iPad)

The third model of iPad, known as **iPad 3** (or **iPad third generation**), has been marketed since March 16, 2012. It was equipped with a 1 GHz S5L8945 ARM processor (known as Apple A5X) and 1 GB of RAM memory. It was produced in three models: Wi-Fi only (model number A1416), Wi-Fi and cellular (VZ) (model number A1403), and cellular and Wi-Fi (model number A1430). There are three hardware strings of identification: iPad3,1 (Wi-Fi only); iPad3,2 (Wi-Fi, GSM, and CDMA); and iPad3,3 (Wi-Fi and GSM). All versions were produced with 16, 32, or 64 memory. Regarding its software, it originally used iOS 5.1. Currently, iPad 3 is supported by the latest version available (iOS 9.3.4), but it will not be supported by iOS 10. The production of these devices was discontinued in October 2012.

iPad 4 (with Retina display)

The fourth model of iPad, known as **iPad 4** (or **iPad fourth generation**), has been marketed since November 2, 2012. It was equipped with a 1.4 GHz S5L8955 ARM processor (known as Apple A6X) and 1 GB of RAM. It was produced in three models: Wi-Fi only (model number A1458), Wi-Fi and cellular (MM) (model number A1460), and cellular and Wi-Fi (model number A1459). There were three hardware strings of identification: iPad3,4 (Wi-Fi only); iPad3,5 (Wi-Fi and GSM); and iPad3,6 (Wi-Fi, GSM, and CDMA). All versions were produced with 16, 32, 64, or 128 GB memory. Regarding its software, it originally used iOS 6.0.1. Currently, iPad 4 is supported by the latest version available (iOS 9.3.4) and it will be supported by iOS 10. The production of these devices was discontinued in October 2014.

iPad Air

The fifth model of iPad, known as **iPad Air**, has been marketed since November 2013. It was equipped with a 64 bit 1.4 GHz S5L8965 ARM processor (known as Apple A7) and 1 GB of RAM memory. It also has a motion coprocessor Apple M7. It was produced in three models: Wi-Fi only (model number A1474), Wi-Fi and cellular (model number A1475), and Wi-Fi and cellular China (model A1476). There are three hardware strings of identification: iPad4,1 (Wi-Fi only), iPad4,2 (Wi-Fi and cellular), and iPad4,3 (Wi-Fi and cellular China). All versions were produced with 16, 32, 64, or 128 GB memory. Regarding its software, it originally used iOS 7.0.3. Currently, iPad Air is supported by the latest version available (iOS 9.3.4) and it will be supported by iOS 10. The production of these devices was discontinued in March 2016.

iPad Air 2

The sixth model of iPad, known as **iPad Air 2**, has been marketed since October 2014. It is equipped with a 64 bit 1.5 GHz T7001 ARM processor (known as Apple A8X) and 2 GB of RAM memory. It also has a motion coprocessor Apple M8. It was produced in two models: Wi-Fi only (model number A1566) and cellular and Wi-Fi (model number A1567). There are two hardware strings of identification: iPad5,3 (Wi-Fi only) and iPad5,4 (Wi-Fi and cellular). All versions were produced with 16, 64, or 128 GB memory. Regarding its software, it originally used iOS 8.1. Currently, iPad Air 2 is supported by the latest version available (iOS 9.3.4) and it will be supported by iOS 10. The production of this device is still active at the date of the publication of this book.

iPad Pro (12.9 inch)

The first model of iPad Pro, a bigger version of the iPad, is known as **iPad Pro 12.9 inch** and has been marketed since November 2015. It is equipped with a 64 bit 2.26 GHz S8001 ARM processor (known as Apple A9X) and 4 GB of RAM. It also has a motion coprocessor Apple M9. It was produced in two models: Wi-Fi only (model number A1584) and Wi-Fi and cellular (model number A1652). There are two hardware strings of identification: iPad6,7 (Wi-Fi only) and iPad6,8 (Wi-Fi and cellular). All versions were produced with 32, 128, or 256 GB memory. Regarding its software, it originally used iOS 9.1. It is currently still supported by the latest version available (iOS 9.3.4) and it will be supported by iOS 10. The production of this device is still active at the date of the publication of this book.

iPad Pro (9.7 inch)

The second model of iPad Pro is known as **iPad Pro 9.7 inch** and has been marketed since March 2016. It is equipped with a 64 bit 2.16 GHz S8001 ARM processor (known as Apple A9X) and 2 GB of RAM. It also has a motion coprocessor Apple M9. It was produced in two models: Wi-Fi only (model number A1673) and Wi-Fi and cellular (model number A1674). There are two hardware strings of identification: iPad6,3 (Wi-Fi only) and iPad6,4 (Wi-Fi and cellular). All versions were produced with 32, 128, or 256 GB memory. Regarding its software, it originally used iOS 9.3. It is currently still supported by the latest version available (iOS 9.3.4) and it will be supported by iOS 10. The production of this device is still active at the date of the publication of this book.

iPad mini

The first model of iPad mini, a smaller version of the iPad, is known simply as **iPad mini** and has been marketed since October 2012. It is equipped with a 1 GHz S5L8942 ARM processor (known as the Apple A5 Rev. A) and 512 MB of RAM. It was produced in three models: Wi-Fi only (model number A1432); Wi-Fi and GSM (model number A1454); and Wi-Fi, GSM, and CDMA (model number A1455). There are three hardware strings of identification: iPad2,5 (Wi-Fi only); iPad2,6 (Wi-Fi and GSM); and iPad2,7 (Wi-Fi, GSM, and CDMA). All versions were produced with 16, 32, or 64 GB memory. Regarding its software, it originally used iOS 6.0.1. It is currently still supported by the latest version available at the time of writing the book (iOS 9.3.4) but it will not be supported by iOS 10. The production of these devices was discontinued in June 2015.

iPad mini second generation

The second model of iPad mini, known as **iPad mini second generation** (or **iPad mini with Retina display**), has been marketed since November 2013. It is equipped with a 64 bit 1.3 GHz S5L8960 ARM processor (known as Apple A7) and 1 GB of RAM. Compared to its predecessor, it uses a Retina screen and an Apple M7 motion coprocessor. It was produced in three models: Wi-Fi only (model number A1489), Wi-Fi and cellular (model number A1490), and Wi-Fi and cellular China (model number A1491). There are three hardware strings of identification: iPad4,4 (Wi-Fi only); iPad4,5 (Wi-Fi and cellular); and iPad4,6 (Wi-Fi and cellular China). All versions were produced with 16, 32, 64, or 128 GB memory. Regarding its software, it originally used iOS 7.0.3. It is currently still supported by the latest version available (iOS 9.3.4) and it will be supported by iOS 10. The production of this device is still active at the date of the publication of this book.

iPad mini third generation

The third model of iPad mini, known as **iPad mini third generation**, has been marketed since October 2014. It is equipped with a 64 bit 1.3 GHz S5L8960 ARM processor (known as Apple A7) and 1 GB of RAM. It also has a motion coprocessor Apple M7. It was produced in three models: Wi-Fi only (model number A1599), Wi-Fi and cellular (model number A1600), and Wi-Fi and cellular China (model number A1601). There are three hardware strings of identification: iPad4,7 (Wi-Fi only); iPad4,8 (Wi-Fi and cellular); and iPad4,9 (Wi-Fi and cellular China). All versions were produced with 16, 64, or 128 GB memory. Regarding its software, it originally used iOS 8.0. It is currently still supported by the latest version available (iOS 9.3.4) and it will be supported by iOS 10. The production of these devices was discontinued in September 2015.

iPad mini fourth generation

The fourth model of iPad mini, known as **iPad mini fourth generation**, has been marketed since September 2015. It is equipped with a 64 bit 1.5 GHz T7000 ARM processor (known as Apple A8) and 2 GB of RAM. It also has a motion coprocessor Apple M8. It was produced in two models: Wi-Fi only (model number A1538) and Wi-Fi and cellular (model number A1550). There are two hardware strings of identification: iPad5,1 (Wi-Fi only) and iPad5,2 (Wi-Fi and cellular). All versions were produced with 16, 64, or 128 GB memory. Regarding its software, it originally used iOS 9.0. It is currently still supported by the latest version available (iOS 9.3.4) and it will be supported by iOS 10. The production of this device is still active at the date of the publication of this book.

iPod touch

The iPod touch device is a media player that looks like the iPhone and uses the iOS operating system. It can play media and video games. It includes a Wi-Fi connection so that it can access the Internet with the mobile version of Safari, purchase songs online from the iTunes Store, and download apps from the App Store.

iPod touch (first generation)

The first model of iPod touch, known simply as **iPod touch**, has been marketed since September 2007. It was equipped with a 620 MHz S5L8900 ARM processor and 128 MB of RAM memory. It is identified by the model number A1213 and by the hardware string iPod1,1. It was produced with 8, 16, or 32 GB memory. Regarding its software, it originally used iPhone OS 1.1. The latest supported version is iPhone OS 3.1.3. The production of these devices was discontinued in September 2008.

iPod touch (second generation)

The second model of iPod touch, known as **iPod touch** (**second generation**) has been marketed since September 2008. It was equipped with a 620 MHz S5L8720 ARM processor and 128 MB of RAM memory. It is identified by the model number A1288 and by the hardware string iPod2,1. It was produced with 8, 16, or 32 GB memory. Regarding its software, it originally used iPhone OS 2.1.1. The latest supported version is iOS 4.2.1. The production of these devices was discontinued in September 2010.

iPod touch (third generation)

The third model of iPod touch, known as **iPod touch** (**third generation**) has been marketed since September 2009. It was equipped with an 833 MHz S5L8922 ARM processor and 256 MB of RAM memory. It is identified by the model number A1318 and by the hardware string iPod3,1. It was produced with 32 or 64 GB memory. Regarding its software, it originally used iPhone OS 3.1. The latest supported version is iOS 5.1.1. The production of these devices was discontinued in September 2010.

iPod touch (fourth generation)

The fourth model of iPod touch, known as **iPod touch (fourth generation)** has been marketed since September 2010. It was equipped with a 1 GHz S5L8930 ARM processor (known as Apple A4) and 256 MB of RAM memory. It is identified by the model number A1367 and by the hardware string iPod4,1. It was produced with 8, 16, 32, or 64 GB memory. With regards to the software, it originally used iOS 4.1. The latest supported version is iOS 6.1.6. The production of these devices was discontinued in May 2013.

iPod touch (fifth generation)

The fifth model of iPod touch, known as **iPod touch (fifth generation)** has been marketed since October 2012. It was equipped with a 1 GHz S5L8942 ARM processor (known as Apple A5) and 512 MB of RAM memory. It is identified by the model number A1421 or A1509 and by the hardware string iPod5,1. It was produced with 16, 32, or 64 GB memory. Regarding its software, it originally used iOS 6.0. It is currently still supported by the latest version available (iOS 9.3.4) but it will not be supported by iOS 10. The production of these devices was discontinued in July 2015.

iPod touch (sixth generation)

The sixth model of iPod touch, known as **iPod touch** (**sixth generation**) has been marketed since July 2015. It is equipped with a 64 bit 1.4 GHz T7000 ARM processor (known as Apple A8) and 1 GB of RAM memory. It also has a motion coprocessor Apple M8. It is identified by the model number A1574 and by the hardware string iPod7,1. It was produced with 16, 32, 64, or 128 GB memory. Regarding its software, it originally used iOS 8.4. It is currently still supported by the latest version available (iOS 9.3.4) and it will be supported by iOS 10. The production of this device is still active at the date of the publication of this book.

Apple TV

The Apple TV device is a digital media player that can receive digital data from various sources and play them on a TV screen. It uses a variant of the iOS operating system, known as **Apple TV Software**. It includes a Wi-Fi connection so that it can access the Internet and purchase songs and films online from the iTunes Store.

Apple TV (first generation)

The first model of Apple TV, known simply as **Apple TV**, has been marketed since January 2007. It was equipped with a 1 GHz Intel Pentium M processor and 256 MB of RAM memory. It is identified by the model number A1218 and by the hardware string AppleTV1,1. It was produced with 40 or 160 GB memory. Regarding its software, it originally used Apple TV Software 1.0. The latest supported version is Apple TV Software 3.0.2. The production of these devices was discontinued in September 2010.

Apple TV (second generation)

The second model of Apple TV, known as **Apple TV 2G** or **second generation**, has been marketed since September 2010. It was equipped with an Apple A4 S5L8930 processor and 256 MB of RAM memory. It is identified by the model number A1378 and by the hardware string AppleTV2,1. It has a 8 GB memory used for cache. Regarding its software, it originally used Apple TV Software 4.0. The latest supported version is Apple TV Software 6.2.1. The production of these devices was discontinued in March 2012.

Apple TV (third generation)

The third model of Apple TV, known as **Apple TV 3G** or **third generation**, has been marketed since March 2012. It was equipped with an Apple A5 Rev. A S5L8942 processor and 512 MB of RAM memory. It is identified by the model number A1427 and by the hardware string AppleTV3,1. It has a 8 GB memory used for cache. Regarding its software, it originally used Apple TV Software 5.0. The latest supported version is Apple TV Software 7.2.1. The production of these devices was discontinued in March 2013.

Apple TV (third generation Rev. A)

A revised version of Apple TV 3G, known as **Apple TV 3G Rev. A** or **third generation Rev. A**, has been marketed since January 2013. It is equipped with an Apple A5 Rev. 8 S5L8947 processor and 512 MB of RAM memory. It is identified by the model number A1469 and by the hardware string AppleTV3,2. It has a 8 GB memory used for cache. Regarding its software, it originally used Apple TV Software 5.2. The latest supported version is Apple TV Software 7.2.1. The production of this device is still active at the date of the publication of this book.

Apple TV (fourth generation)

The fourth model of Apple TV touch, known as **Apple TV 4G** or **fourth generation**, has been marketed since October 2015. It is equipped with an Apple A8 T7000 processor and 2 GB of RAM memory. It is identified by the model number A1625 and by the hardware string AppleTV5,3. It was produced with 32 or 64 GB memory. Regarding its software, it originally used tvOS 9.0 (a completely redesigned version of Apple TV Software). It is currently still supported by the latest version available (tvOS 9.2.2). The production of this device is still active at the date of the publication of this book.

Apple Watch

The Apple Watch is a smart watch that has been developed and produced by Apple since 2015. It uses the watchOS operating system, based on iOS. Apple Watch needs an iPhone, connected via Wi-Fi or Bluetooth, to perform most of its functions like making a call or sending a text message. At the date of the publication of this book, two versions are available: 38 mm (A1553 model, Watch1,1 string) and 42 mm (A1554 model, Watch1,2 string). Both versions are available with four distinct variants (Regular, Sport, Gold, and Hermes) and use an Apple S1 S7002 processor with 512 MB of RAM Memory and 8 GB device memory.

iOS devices connectors

iOS devices (iPhone, iPad, and iPod Touch) use two different types of connectors:

- **30-pin dock connector** (`https://en.wikipedia.org/wiki/Dock_connector`): This is Apple's proprietary 30-pin connector that was designed in 2003 and produced up to 2014. It was used by iPhone, iPhone 3G, iPhone 3GS, iPhone 4, iPhone 4s, iPad, iPad 2, iPad 3, iPod touch first generation, iPod touch second generation, iPod touch third generation, and iPod touch fourth generation.
- **Lightning connector** (`https://en.wikipedia.org/wiki/Lightning_(connector)`): This is Apple's proprietary 8-pin connector that was designed in 2012 and is still in use. It has been used by iPhone 5, iPhone 5c, iPhone 5s, iPhone 6, iPhone 6 Plus, iPhone 6s, iPhone 6s Plus, iPad 4, iPad Air, iPad Air 2, iPad Mini first generation, iPad Mini second generation, iPad Mini third generation, iPad Mini fourth generation, iPad Mini fifth generation, and iPod touch fifth generation.

iOS devices matrix

In the following table, we propose a useful matrix to easily identify iOS friend devices:

Device name	Model number	Internal	Identifier
iPhone	A1203	M68AP	iPhone1,1
iPhone 3G	A1241	N82AP	iPhone1,2
	A1324 (China/No Wi-Fi)		
iPhone 3GS	A1303	N88AP	iPhone2,1
	A1325 (China/No Wi-Fi)		
iPhone 4	A1332 (GSM)	N90AP	iPhone3,1
iPhone 4	A1349 (CDMA/Verizon/Sprint)	N92AP	iPhone3,3
iPhone 4s	A1387	N94AP	iPhone4,1
	A1431 (GSM China/WAPI)		
iPhone 5	A1428 (GSM/LTE/North America)	N41AP	iPhone5,1
	A1429 (GSM/LTE 1, 3, 5/International)		
iPhone 5 rev.2	A1429 (CDMA/LTE, Sprint/Verizon)	N42AP	iPhone5,2
	A1442 (CDMA China/UIM/WAPI)		
iPhone 5C	A1456 (CDMA/US/Japan)	N48AP	iPhone5,3
	A1532 (CDMA/China Telecom)		
	A1532 (GSM/North America)		
	A1532 (CDMA/Verizon)		
iPhone 5C	A1507 (UK/Europe/Middle East)	N49AP	iPhone5,4
	A1516 (China Mobile)		
	A1526 (China Unicom)		
	A1529 (Asia Pacific)		
iPhone 5s	A1453 (CDMA/US/Japan)	N51AP	iPhone6,1
	A1533 (CDMA/Verizon)		
	A1533 (CDMA/China Telecom)		
	A1533 (GSM/North America)		
iPhone 5s	A1457 (UK/Europe/Middle East)	N53AP	iPhone6,2
	A1518 (China Mobile)		
	A1528 (China Unicom)		
	A1530 (Asia Pacific)		
iPhone 6 Plus	A1522 (CDMA/Verizon)	N56AP	iPhone7,1
	A1522 (GSM/North America)		
	A1524 (Global/Sprint)		
	A1593 (China Mobile)		
iPhone 6	A1549 (CDMA/Verizon)	N61AP	iPhone7,2
	A1549 (GSM/North America)		
	A1586 (Global/Sprint)		
	A1589 (China Mobile)		

Device name	**Model number**	**Internal**	**Identifier**
iPhone 6s	A1633 (AT&T/SIM Free)	N71AP	iPhone8,1
	A1688 (Global)		
	A1700 (Mainland China)		
iPhone 6s Plus	A1634 (AT&T/SIM Free)	N66AP	iPhone8,2
	A1687 (Global)		
	A1699 (Mainland China)		
iPhone SE	A1662 (North America)	N69AP	iPhone8,4
	A1723 (Global)		
	A1724 (China)		
iPad	A1219 (Wi-Fi)	K48AP	iPad1,1
	A1337 (Wi-Fi + 3G)		
iPad 2	A1395 (Wi-Fi)	K93AP	iPad2,1
	A1396 (Wi-Fi/GSM/GPS)	K94AP	iPad2,2
	A1397 (Wi-Fi/CDMA/GPS)	K95AP	iPad2,3
	A1395 (Wi-Fi – With A5 Rev.A)	K93AAP	iPad2,4
iPad 3	A1416 (Wi-Fi)	J1AP	iPad3,1
	A1403 (Wi-Fi/Cellular Verizon)	J2AP	iPad3,2
	A1430 (Wi-Fi/Cellular)	J2AAP	iPad3,3
iPad 4	A1458 (Wi-Fi)	P101AP	iPad3,4
	A1459 (Wi-Fi/Cellular)	P101AP	iPad3,5
	A1460 (Wi-Fi/Cellular Verizon)	P103AP	iPad3,6
iPad Air	A1474 (Wi-Fi)	J71AP	iPad4,1
	A1475 (Wi-Fi/Cellular)	J72AP	iPad4,2
	A1476 (Wi-Fi/Cellular China)	J73AP	iPad4,3
iPad Air 2	A1566 (Wi-Fi)	J81AP	iPad5,3
	A1567 (Wi-Fi/Cellular)	J82AP	iPad5,4
iPad Mini	A1432 (Wi-Fi)	P105AP	iPad2,5
	A1454 (Wi-Fi/Cellular)	P106AP	iPad2,6
	A1455 (Wi-Fi/Cellular Verizon)	P107AP	iPad2,7
iPad Mini 2	A1489 (Wi-Fi)	J85AP	iPad4,4
	A1490 (Wi-Fi/Cellular)	J86AP	iPad4.5
	A1491 (Wi-Fi/Cellular China)	J87AP	iPad4,6
iPad Mini 3	A1599 (Wi-Fi)	J85mAP	iPad4,7
	A1600 (Wi-Fi/Cellular)	J86mAP	iPad4,8
	A1601 (Wi-Fi/Cellular China)	J87mAP	iPad4,9
iPad Mini 4	A1538 (Wi-Fi)	J96AP	iPad5,1
	A1550 (Wi-Fi/Cellular)	J97AP	iPad5,2

Device name	Model number	Internal	Identifier
iPad Pro 12.9	A1584 (Wi-Fi)	J98aAP	iPad6,7
	A1652 (Wi-Fi/Cellular)	J99aAP	iPad6,8
iPad Pro 9.7	A1673 (Wi-Fi)	J127AP	iPad6,3
	A1674 (Wi-Fi/Cellular)	J128AP	iPad6,4
iPod Touch 1st generation	A1213	N45AP	iPod1,1
iPod Touch 2nd generation	A1288	N72AP	iPod2,1
	A1319 (China)		
iPod Touch 3rd generation	A1318	N18AP	iPod3,1
iPod Touch 4th generation	A1367	N81AP	iPod4,1
iPod Touch 5th generation	A1421	N78AP	iPod5,1
	A1509 (No iSight)	N78aAP	
iPod Touch 6th generation	A1574	N102AP	iPod7,1
Apple TV 1st generation	A1218		AppleTV1,1
Apple TV 2nd generation	A1378	K66AP	AppleTV2,1
Apple TV 3rd generation	A1427	J33AP	AppleTV3,1
Apple TV 3rd generation Rev. A	A1469	J33IAP	AppleTV3,2
Apple TV 4th generation	A1625	J42dAP	AppleTV5,3
Apple Watch 38 mm	A1553	N27aAP	Watch1,1
Apple Watch 42 mm	A1554	N28aAP	Watch1,2

- **iOS models** (`http://theiphonewiki.com/wiki/Models`): This page contains detailed tables with device name, device model, FCC-ID, internal name, and hardware identifier. Some useful information about the iOS devices can be found at the following links:
    - **Application Processor** (`http://theiphonewiki.com/wiki/Application_Processor`): This page contains a detailed list of the processors installed on the iOS devices
    - **Identify your iPhone Model** (`https://support.apple.com/en-us/HT21296`): This is the official Apple page containing information to identify an iPhone model

- **iPhone** (`http://theiphonewiki.com/wiki/IPhone`): This page contains a detailed table with all the features and characteristics for every iPhone model
- **Identify your iPad Model** (`https://support.apple.com/en-us/HT21471`): This is the official Apple page containing information to identify an iPad model
- **iPad** (`http://theiphonewiki.com/wiki/IPad`): This page contains a detailed table with all the features and characteristics for every iPad model
- **Identify your iPod Model** (`https://support.apple.com/en-us/HT24217`): This is the official Apple page containing information to identify an iPod model
- **iPod touch** (`http://theiphonewiki.com/wiki/IPod_touch`): This page contains a detailed table with all the features and characteristics for every iPod touch model
- **Identify your Apple TV Model** (`https://support.apple.com/en-us/HT28`): This is the official Apple page containing information to identify an Apple TV model
- **Apple TV** (`https://www.theiphonewiki.com/wiki/Apple_TV`): This page contains a detailed table with all the features and characteristics for every Apple TV model
- **Identify your Apple Watch** (`https://support.apple.com/en-us/HT2457`): This is the official Apple page containing information to identify an Apple Watch model
- **iOS Support Matrix** (`http://iossupportmatrix.com/`): This page contains a visual representation of all the iDevice models with their hardware and software features and support
- **iPhone IMEI** (`http://iphoneimei.info/`): This page contains a search engine to find the specific iPhone model from the IMEI number
- **IMEI.info** (`http://www.imei.info/`): This link is similar to the preceding link
- **iPhoneox** (`http://www.iphoneox.com/`): This link is similar to the preceding link

iOS operating system

All the devices described in this chapter have in common the use of the iOS operating system, originally known as iPhone OS up to version 3. It was developed by Apple specifically for iPhone, iPad, and iPod touch. It was unveiled for the first time in January 2007 and was introduced with the first model of iPhone in June of the same year.

iOS is an operating system based on the older forefather Mac OS X, a derivative of BSD Unix with a Mach kernel XNU based on Darwin OS. It uses the following four levels of abstraction:

- **Core OS**: This level consists of filesystem, memory management, security, power management, TCP/IP, sockets, and encryption
- **Core services**: This level consists of networking, SQLite, geolocation, and threads
- **Media**: This level consists of OpenAL, audio, image, video, and OpenGL
- **Cocoa touch**: This level consists of core animation, multitasking, and gesture recognizer

The main screen, known as SpringBoard, is divided into the following three parts:

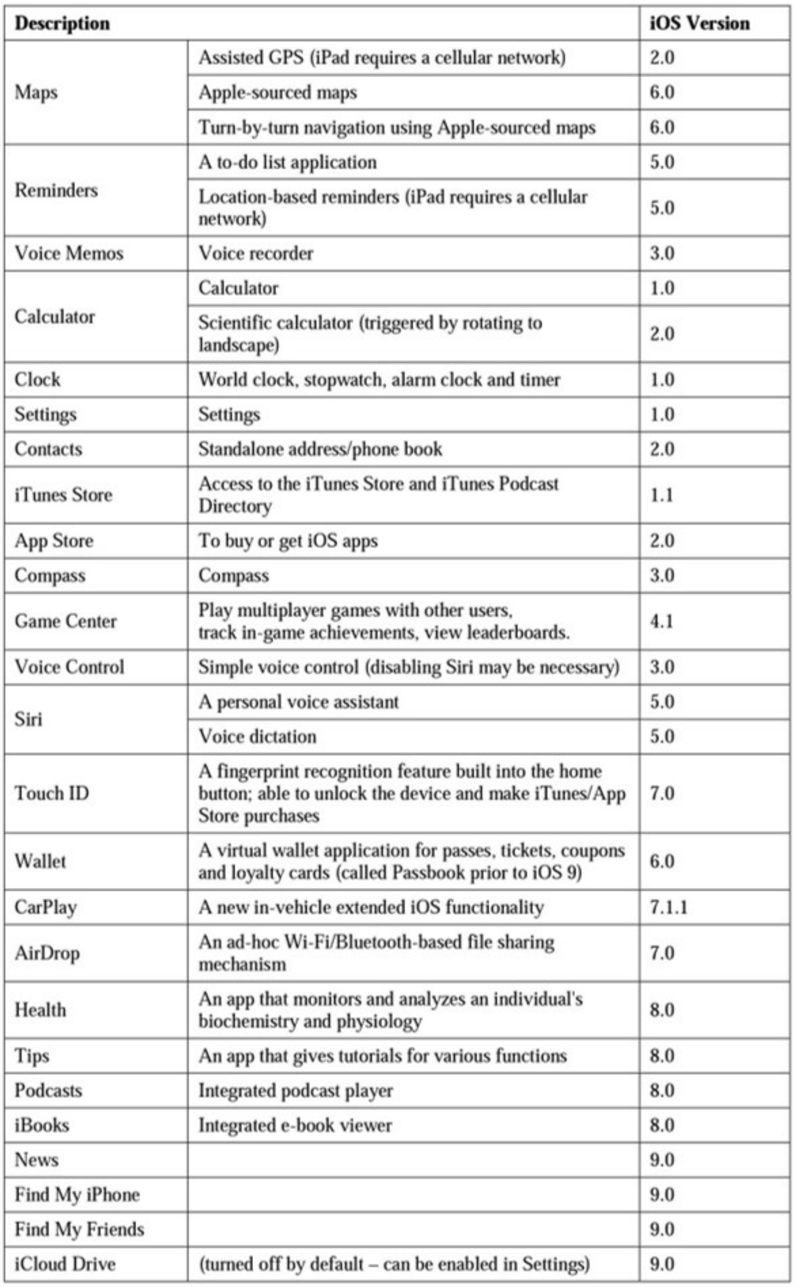
- The top bar that displays the telephone signal, any 3G/Wi-Fi/Bluetooth active connections, and the battery status
- The central part containing the icons of the applications in your device
- The bar at the bottom containing the most frequently-used applications, such as the following:
    - iPhone: Phone, Mail, Safari, and Music
    - iPad/iPod touch: Messages, Mail, Safari, and Music

The home screen appears whenever the user unlocks the device or presses the Home button while in another app.

The complete list of all the operating system versions produced by Apple is published and frequently updated at `http://theiphonewiki.com/wiki/Firmware` and a detailed explanation of all specific versions and subversions is available at `https://en.wikipedia.org/wiki/IOS_version_history`. Moreover, at `http://www.ipswdownloader.com/`, it is possible to download all firmware for all models.

The following table provides a detailed list of included applications with the specific version where the application was originally introduced. This table is extracted and redacted from the URL `https://en.wikipedia.org/wiki/IOS`.

Description		**iOS Version**
Phone	Telephone	1.0
Mail	Email client	1.0
Safari	Web browser	1.0
Music	Portable media player	5.0
Videos	Video player	5.0
SpringBoard	Home screen	1.0
	Spotlight search	3.0
	Folders	4.0
	Home screen backgrounds	4.0
Messages	Text messaging	1.0
	MMS	3.0
	iMessage instant messaging	5.0
Calendar	Calendar	1.0
Photos	Photo viewer	1.0
	Video viewer	2.0
	Crop, red eye fix, auto enhance and photo rotate	5.0
Camera	Camera	1.0
	Camcorder	3.0
	Auto-focus	3.0
	HDR	4.1
	Crop, red eye fix, auto enhance and photo rotate	5.0
	Panorama	6.0
	Take still photos while recording video	6.0
	Photo filters	7.0
	Burst mode	7.0
FaceTime	Video calling over Wi-Fi	4.0
	Video calling over 3G/LTE (iPad requires a cellular network)	6.0
	FaceTime Audio	7.0
Photo Booth	A camera application with added special effects	4.2.1
Stocks	Stocks provided by Yahoo! Finance	1.0
	Stocks Widget for Notification Center	5.0
Weather	Weather provided by The Weather Channel	8.0
	Weather Widget for Notification Center	5.0
Notes	A simple note-taking program	1.0



Description		**iOS Version**
Maps	Assisted GPS (iPad requires a cellular network)	2.0
	Apple-sourced maps	6.0
	Turn-by-turn navigation using Apple-sourced maps	6.0
Reminders	A to-do list application	5.0
	Location-based reminders (iPad requires a cellular network)	5.0
Voice Memos	Voice recorder	3.0
Calculator	Calculator	1.0
	Scientific calculator (triggered by rotating to landscape)	2.0
Clock	World clock, stopwatch, alarm clock and timer	1.0
Settings	Settings	1.0
Contacts	Standalone address/phone book	2.0
iTunes Store	Access to the iTunes Store and iTunes Podcast Directory	1.1
App Store	To buy or get iOS apps	2.0
Compass	Compass	3.0
Game Center	Play multiplayer games with other users, track in-game achievements, view leaderboards.	4.1
Voice Control	Simple voice control (disabling Siri may be necessary)	3.0
Siri	A personal voice assistant	5.0
	Voice dictation	5.0
Touch ID	A fingerprint recognition feature built into the home button; able to unlock the device and make iTunes/App Store purchases	7.0
Wallet	A virtual wallet application for passes, tickets, coupons and loyalty cards (called Passbook prior to iOS 9)	6.0
CarPlay	A new in-vehicle extended iOS functionality	7.1.1
AirDrop	An ad-hoc Wi-Fi/Bluetooth-based file sharing mechanism	7.0
Health	An app that monitors and analyzes an individual's biochemistry and physiology	8.0
Tips	An app that gives tutorials for various functions	8.0
Podcasts	Integrated podcast player	8.0
iBooks	Integrated e-book viewer	8.0
News		9.0
Find My iPhone		9.0
Find My Friends		9.0
iCloud Drive	(turned off by default – can be enabled in Settings)	9.0

iDevice identification

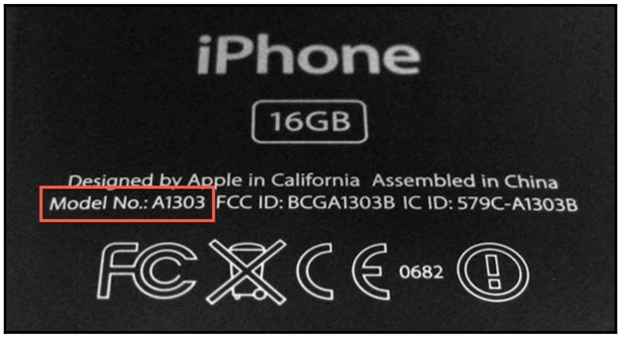
It is very useful for a forensic investigator to be able to recognize the specific model of an iOS device while conducting a search and seizure or prior to an acquisition activity.

The recognition phase can be performed in the following four ways:

- Identifying the shape of the device and the connector used
- Checking the model number printed on the back of the device
- Connecting the device to a laptop and directly communicating with it
- Directly through the OS by navigating to **Settings** | **General** | **About**

The first method can be used by practicing the identification of the unique characteristics of each model. In some cases, it may be a complex assessment and it is therefore advisable to confirm the first evaluation with one of the other three methods.

The second method requires you to identify, from the back of the device, the model number. As reported in the previous tables, from the model number it is easy to identify the type of device. In the example shown in the following screenshot, it is possible to identify the model as an A1303 or an iPhone 3GS with 16 GB memory:


iPhone
16GB
Designed by Apple in California Assembled in China
Model No.: A1303 FCC ID: BCGA1303B IC ID: 579C-A1303B
0682

The third method is to retrieve the information directly, interacting with the device connected to a computer. As we will explore later on, once you turn on an iDevice, it can be password-protected and present a view to insert the lock code. Regardless of the knowledge of the code or the ability to overcome it or violate it, the device can communicate some information when connected to a computer.

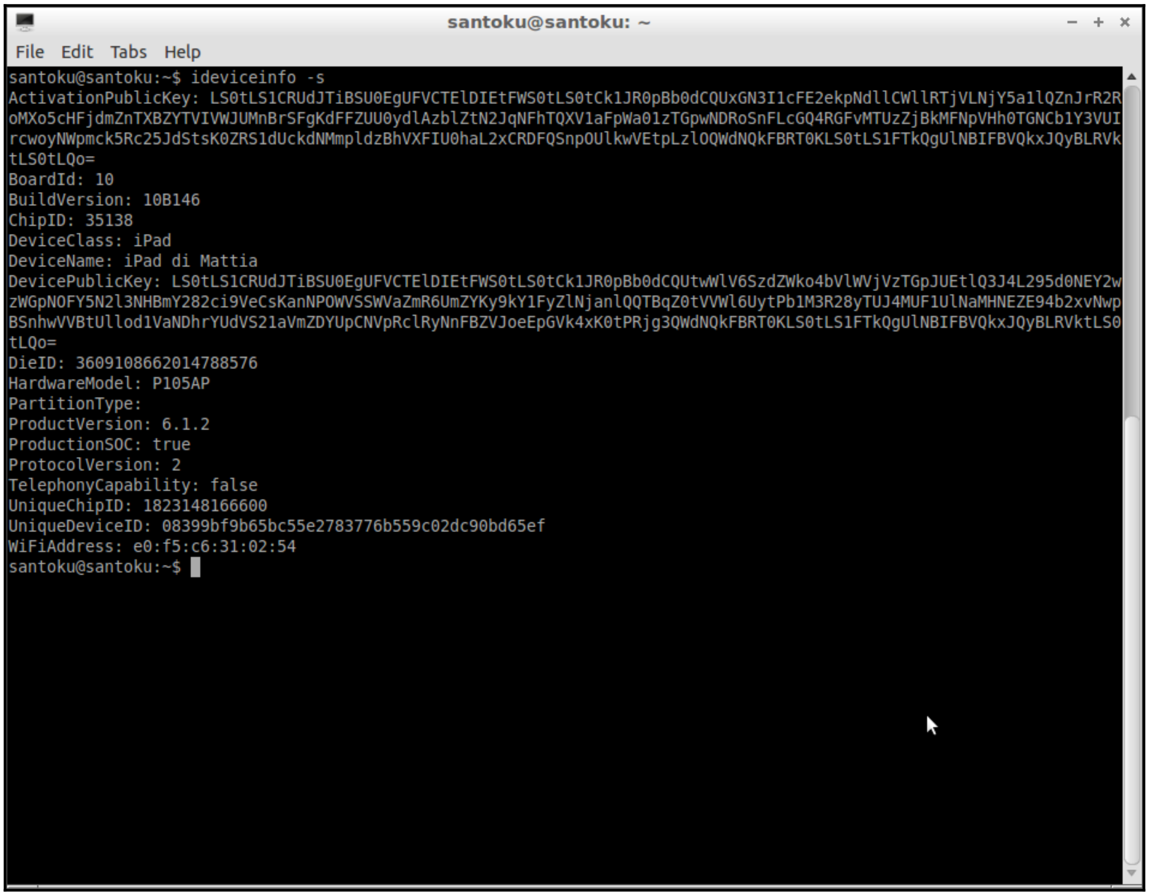
Very useful in this context is the collection of tools and libraries available at `http://www.libimobiledevice.org/` and preinstalled in the Linux distributions Santoku (`https://santoku-linux.com/`) and DEFT 8.1 (`http://www.deftlinux.net`).

Using the `ideviceinfo` command, it is possible to extract some information from the device, with no need of unlocking it.

The information that can be extracted is as follows:

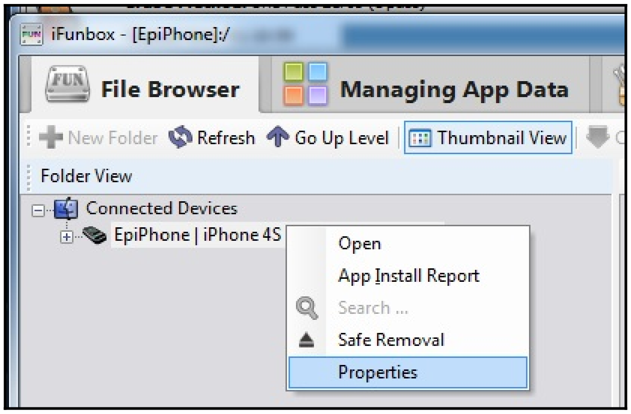
- Device name
- Device class
- Hardware model
- iOS version
- Telephony capability
- Unique device ID
- Bluetooth MAC address
- Wi-Fi MAC address

In the example shown in the following screenshot, it is possible to identify that the connected device is a Wi-Fi only iPad mini 1 (hardware model P105AP) with iOS 6.1.2 (build 10B146) called `iPad di Mattia`:

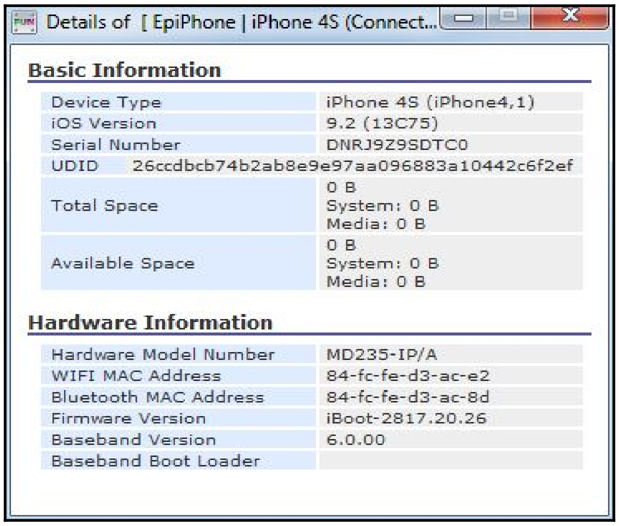


```
santoku@santoku:~$ ideviceinfo -s
ActivationPublicKey: LS0tLS1CRUdJTiBSU0EgUFVCTElDIEtFWS0tLS0tCk1JR0pBb0dCQUxGN3I1cFE2ekpNdllCWllRTjVLNjY5a1lQZnJrR2R
oMXo5cHFjdmZnTXBZYTVIVWJUMnBrSFgKdFFZUU0ydlAzblZtN2JqNFhTQXV1aFpWa01zTGpwNDRoSnFLcGQ4RGFvMTUzZjBkMFNpVHh0TGNCb1Y3VUI
rcwoyNWpmck5Rc25JdStsK0ZRS1dUckdNMmpldzBhVXFIU0haL2xCRDFQSnpOUlkwVEtpLzlOQWdNQkFBRT0KLS0tLS1FTkQgUlNBIFBVQkxJQyBLRVk
tLS0tLQo=
BoardId: 10
BuildVersion: 10B146
ChipID: 35138
DeviceClass: iPad
DeviceName: iPad di Mattia
DevicePublicKey: LS0tLS1CRUdJTiBSU0EgUFVCTElDIEtFWS0tLS0tCk1JR0pBb0dCQUtwWlV6SzdZWko4bVlWVjVzTGpJUEtlQ3J4L295d0NEY2w
zWGpNOFY5N2l3NHBmY282ci9VeCsKanNPOWVSSWVaZmR6UmZYKy9kY1FyZlNjanlQQTBqZ0tVVWl6UytPb1M3R28yTUJ4MUF1UlNaMHNEZE94b2xvNwp
BSnhwVVBtUllod1VaNDhrYUdVS21aVmZDYUpCNVpRclRyNnFBZVJoeEpGVk4xK0tPRjg3QWdNQkFBRT0KLS0tLS1FTkQgUlNBIFBVQkxJQyBLRVktLS0
tLQo=
DieID: 3609108662014788576
HardwareModel: P105AP
PartitionType:
ProductVersion: 6.1.2
ProductionSOC: true
ProtocolVersion: 2
TelephonyCapability: false
UniqueChipID: 1823148166600
UniqueDeviceID: 08399bf9b65bc55e2783776b559c02dc90bd65ef
WiFiAddress: e0:f5:c6:31:02:54
santoku@santoku:~$ 
```

Another useful tool in order to extract information about the device is iFunBox (`http://www.i-funbox.com/`). The device information is available by right clicking on the device name in the **File Browser** window and selecting **Properties**:



In this example, it is possible to identify that the connected device is a **iPhone 4s** (hardware model **MD235-IP/A**) with iOS **9.2 (13C75)** called EpiPhone:


Details of [EpiPhone | iPhone 4S (Connect...

Basic Information

Device Type	iPhone 4S (iPhone4,1)
iOS Version	9.2 (13C75)
Serial Number	DNRJ9Z9SDTC0
UDID	26ccdbcb74b2ab8e9e97aa096883a10442c6f2ef
Total Space	0 B System: 0 B Media: 0 B
Available Space	0 B System: 0 B Media: 0 B

Hardware Information

Hardware Model Number	MD235-IP/A
WIFI MAC Address	84-fc-fe-d3-ac-e2
Bluetooth MAC Address	84-fc-fe-d3-ac-8d
Firmware Version	iBoot-2817.20.26
Baseband Version	6.0.00
Baseband Boot Loader	

iOS filesystem

All the iDevices use HFSX as their filesystem, a variant case of HFS+. It is possible to store two or more files with the same name within the same folder, but with a different case of each individual character (for example, `iOS.jpg` and `ios.jpg`).

The HFS+ filesystem

HFS Plus (or HFS+) is the filesystem developed by Apple to replace, from Mac OS 8.1, HFS as the default filesystem for Mac computers. In Apple's official documentation, it is called**Mac OS Extended**.

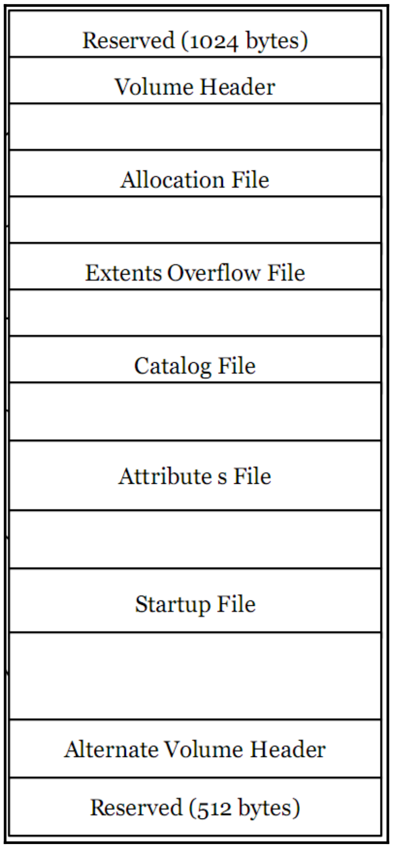
HFS+ is an improved version of HFS, which allows the user to support larger files (thanks to block addresses of 32 bits instead of 16 bits) and uses Unicode for the names of filesystem objects (files and folders), thus allowing up to 255 characters for each. Until Mac OS X Tiger, HFS+ only supported Unix filesystem privileges to access the file. The Tiger version introduced support for security checks based on **Access Control List** (**ACL**), typical of Microsoft environments.

The HFS+ volumes are allocation blocks that may contain one or more sectors (typically 512 bytes in a hard drive). The number of allocation blocks depends on the total size of the volume. The HFS+ filesystem uses 32 bits to address the allocation blocks, thus allowing access to 232 blocks (4,294,967,296).

A typical HFS+ volume is defined by the following six major data structures that contain the information needed to manage the data volume:

- The `Volume Header` file: This file defines the basic structure of the volume, as the size of each allocation block, the number of used and free blocks, and the size and position of the other special files
- The `Allocation` file: This file includes a bitmap with the used and unused blocks within a volume
- The `Catalog` file: This file defines the structure of the folders in the filesystem and it is used to identify the location of a specific file or folder
- The `Extents Overflow` file: This file contains pointers to additional extents for files that require more than eight contiguous allocation blocks
- The `Attributes` file: This file contains the customizable attributes of a file
- The `Startup` file: This file contains the information required at system boot

The data structure can be represented by the following figure:



Both the special and user files are stored in **forks** or in a set of allocation blocks. The space is usually allocated in **clumps**, where the size of a clump is a multiple of the size of a block. The contiguous allocation blocks for a given file are grouped into **extents**. Each extent is characterized by a starting allocation block and by the number of blocks, which indicates how many blocks contain data from that specific file.

In the `boot` blocks and `startup` files, the first 1,024 bytes of a volume are reserved as boot blocks and may contain information requested during the startup of the system. Alternatively, boot information can be found within the `startup` file, which allows you to store a greater amount of information.

A `volume header` file, a 512 byte data structure, contains the volume information including the location of other data structures. It is always located at the beginning of the block 2 or 1,024 bytes after the beginning of the volume. A copy of the `volume header` file, called the **alternate volume header**, is 1,024 bytes before the end of the volume. The first 1,024 bytes and the last 512 bytes of the volume are reserved.

The information contained in a `volume header` file is as follows:

Field name	**Size**	**Description**
`signature`	2 bytes	This field implies the volume signature, which must be `'H+'`, if the volume is HSF Plus, and `'HX'`, if the volume is HFSX
`version`	2 bytes	This field implies the format version, which is `'4'` for HFS Plus and `'5'` for HFSX
`attributes`	4 bytes	This field implies the volume attributes (for example, journaling active)
`lastMountedVersion`	4 bytes	This field describes the operating system installed
`journalInfoBlock`	4 bytes	This field is the allocation block that manages the journaling
`createDate`	4 bytes	This field implies the volume creation date
`modifyDate`	4 bytes	This field implies the volume last modified date
`backupDate`	4 bytes	This field implies the volume last backup date
`checkedDate`	4 bytes	This field implies the volume last consistency check date
`fileCount`	4 bytes	This field implies the number of files in the volume, without the special files
`folderCount`	4 bytes	This field implies the number of folders in the volume, without the root folder
`blockSize`	4 bytes	This field implies the allocation block size (bytes)
`totalBlocks`	4 bytes	This field implies the total number of allocation blocks
`freeBlocks`	4 bytes	This field implies the number of available allocation blocks
`nextAllocation`	4 bytes	This field implies the address of the next available allocation block
`rsrcClumpSize`	4 bytes	This field implies the default clump size for a resource fork

`dataClumpSize`	4 bytes	This field implies the default clump size for a data fork
`nextCatalogID`	4 bytes	This field implies the first available CatalogID
`writeCount`	4 bytes	This field implies the number of times the volume has been mounted
`encondingsBitmap`	8 bytes	This bitmap describes the encoding used for file and folder name
`finderInfo`	32 bytes	This field implies the information used by the Mac OS Finder and the system software boot process
`allocationFile`	80 bytes	This field implies the location and the size of File Allocation
`extentsFile`	80 bytes	This field implies the location and the size of the extents file
`catalogFile`	80 bytes	This field implies the location and the size of the catalog file
`attributesFile`	80 bytes	This field implies the location and the size of the attributes file
`startupFile`	80 bytes	This field implies the location and the size of the startup file

The `allocation` (bitmap) file is used to keep track of which allocation blocks on a volume are currently allocated to a structure (file or folder). It is a bitmap that contains one bit for each allocation block in the volume. If a bit is 1, the corresponding allocation block is in use. If the bit is 0, the corresponding allocation block is not currently in use and is therefore available to be assigned to a file or folder.

The catalog file is used to keep the information on the hierarchy of files and folders on HFS+. A catalog file is organized as a binary tree (type B-tree) and therefore consists of head node, index nodes, and leaf nodes. The position of the first block of the catalog file (and thus the head node of the file) is stored in the volume header file. The catalog file contains the metadata of all the files and folders on a volume, including creation, modification and access date, permissions, file identifier, and information about the user that created the file.

The data structure for each file in the catalog file is as follows:

```
struct HFSPlusCatalogFile {
    SInt16              recordType;
    UInt16              flags;
    UInt32              reserved1;
    HFSCatalogNodeID    fileID;
    UInt32              createDate;
    UInt32              contentModDate;
    UInt32              attributeModDate;
    UInt32              accessDate;
    UInt32              backupDate;
    HFSPlusBSDInfo      permissions;
    FileInfo            userInfo;
    ExtendedFileInfo    finderInfo;
    UInt32              textEncoding;
    UInt32              reserved2;
    HFSPlusForkData     dataFork;
    HFSPlusForkData     resourceFork;
};
```

The two areas of most interest to identify the location of the files are dataFork and resourceFork (both of the HFSPlusForkData type).

The dataFork field contains information about the location and size of a file or the current contents of the file, while the resourceFork field contains the application metadata of the file.

The HFSPlusForkData data structure is defined by four fields as follows:

```
struct HFSPlusForkData {
    UInt64                  logicalSize;
    UInt32                  clumpSize;
    UInt32                  totalBlocks;
    HFSPlusExtentRecord     extents;
};
```

The `logicalSize` field defines the size in bytes of the data, the `totalBlocks` field defines the number of blocks allocated, the `extents` field stores the first eight extents of a file descriptor (an extent is a contiguous segment of a file). If a file requires a greater number of descriptor extents, these are stored in the `extents overflow` file. Each extent that composes a file is described in the `HFSPlusExtentDescriptor` data structure and is defined by the two fields as follows:

```
struct HFSPlusExtentDescriptor {
    UInt32                  startBlock;
    UInt32                  blockCount;
};
```

The `startBlock` field identifies the first allocation block in an extent while the `blockCount` field identifies the length in the number of allocation blocks of an extent. The start offset of a file can then be determined by finding the first extent and multiplying the corresponding `startBlock` field to the size of the allocation block, which is defined in the `volume header` file. Since the files cannot always be completely stored in contiguous blocks on the disk and may be fragmented, `HFS+ dataFork` defines a structure that holds up to eight extents. When a file requires more than eight extents, it uses the `extents overflow` file, which combines the file with additional extents.

For the `extents overflow` file, if a file in an HFS+ volume is composed of more than eight extents (or is fragmented over more than eight contiguous positions of the volume), the extents in excess will be stored in the `extents overflow` file. The file structure is similar to the `content` file (binary tree, B-tree); however, it's greatly simplified by the presence of a single data structure (`HFSPlusExtentKey`).

The `attributes` file enables the direct management through the filesystem of additional attributes for a file. The attributes are defined as key/value pairs.

An interesting concept associated with HFS+ is the filesystem journaling used for a recovery process after a volume was not safely unmounted. This file stores file transactions (`create`, `delete`, `modify`, and so on) and might contain the same metadata stored in the attributes or in the `catalog` file. It is activated by default on the iOS devices and can be used to recover deleted content.

Device partitions

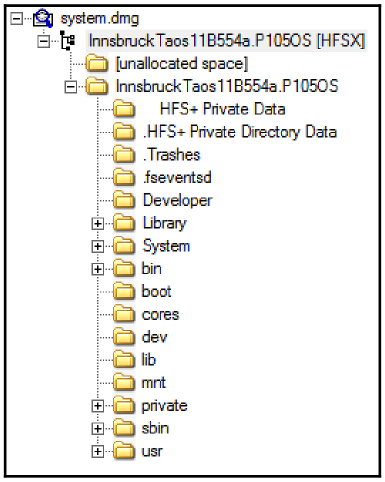
iDevices use a NAND type memory divided into two partitions: the system or firmware partition and the data partition.

The system partition contains the iOS operating system and all the preinstalled applications and it is identified as `/dev/disk0s1` or `/dev/disk0s1s1`. This partition is not generally accessible to the user in the write mode and may only be modified by an update of the operating system. Since it cannot contain user-installed applications and data, it is small and its size depends on the specific model.

The data partition occupies most of the space in the NAND memory and is identified as `/dev/disk0s2` or `/dev/disk0s2s2`. The partition contains user data and user-installed applications and is mounted at run time by the operating system inside `/private/var`.

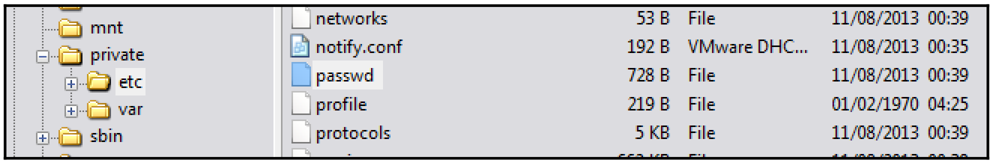
System partition

If the device is in a normal condition, all information relevant to an investigation is within the partition containing the user data. The system partition is therefore not usually of interest. A complete description of the folder content is available at `http://theiphonewiki.com/wiki/` and the partition will look like the following screenshot:



```
system.dmg
  InnsbruckTaos11B554a.P105OS [HFSX]
    [unallocated space]
    InnsbruckTaos11B554a.P105OS
      HFS+ Private Data
      .HFS+ Private Directory Data
      .Trashes
      .fseventsd
      Developer
      Library
      System
      bin
      boot
      cores
      dev
      lib
      mnt
      private
      sbin
      usr
```

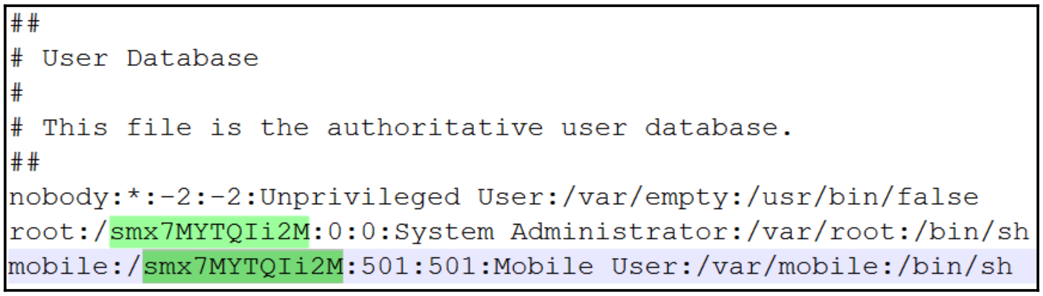
It should be noted, however, that `/private/etc/passwd` (shown in the following screenshot) contains the password of the users configured on the device (`mobile` and `root`):



```
mnt
private
  etc
  var
sbin
```

networks	53 B	File	11/08/2013 00:39
notify.conf	192 B	VMware DHC...	11/08/2013 00:35
passwd	728 B	File	11/08/2013 00:39
profile	219 B	File	01/02/1970 04:25
protocols	5 KB	File	11/08/2013 00:39

For all iDevices, the default password for the `mobile` and `root` users is **alpine**. This password cannot be modified by the user, unless they are performing the jailbreaking operations, as shown in the following screenshot:

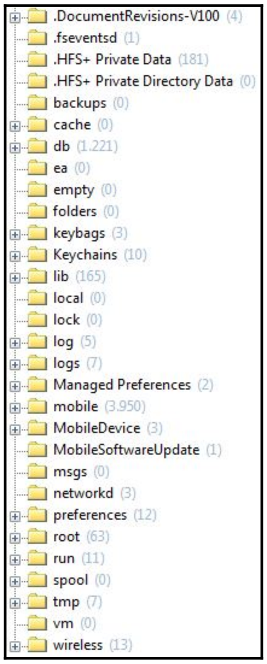


```
##
# User Database
#
# This file is the authoritative user database.
##
nobody:*:-2:-2:Unprivileged User:/var/empty:/usr/bin/false
root:/smx7MYTQIi2M:0:0:System Administrator:/var/root:/bin/sh
mobile:/smx7MYTQIi2M:501:501:Mobile User:/var/mobile:/bin/sh
```

Data partition

The structure of the data partition has changed over the different evolutions of the operating system.

The following screenshot shows an example of the folder structure extracted from a jailbroken iPad mini 1G running iOS 9.0.2:



```
.DocumentRevisions-V100 (4)
.fseventsd (1)
.HFS+ Private Data (181)
.HFS+ Private Directory Data (0)
backups (0)
cache (0)
db (1.221)
ea (0)
empty (0)
folders (0)
keybags (3)
Keychains (10)
lib (165)
local (0)
lock (0)
log (5)
logs (7)
Managed Preferences (2)
mobile (3.950)
MobileDevice (3)
MobileSoftwareUpdate (1)
msgs (0)
networkd (3)
preferences (12)
root (63)
run (11)
spool (0)
tmp (7)
vm (0)
wireless (13)
```

The useful elements for the analysis of an iDevice will be discussed in `Chapter 6`, *Analyzing iOS Devices*. However, here we want to recall the general content of the folders in the *Data partition* section, using *The iPhone Wiki* as the main reference:

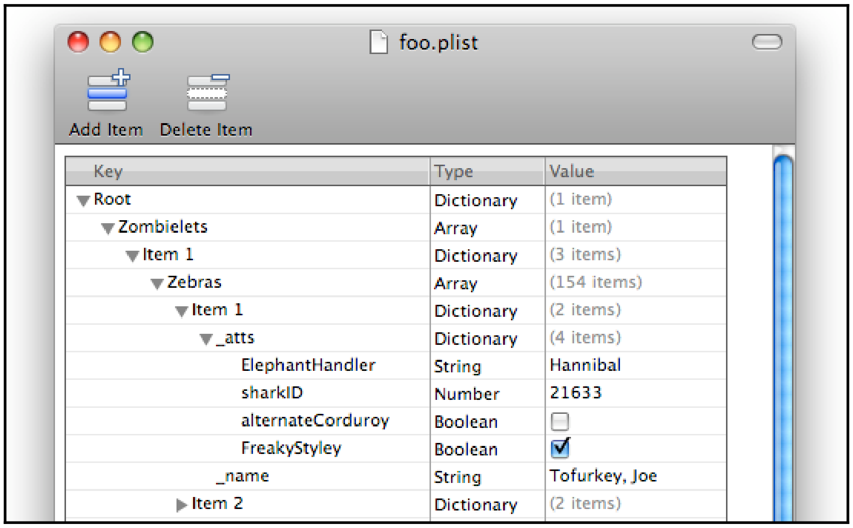
- `backups`: Typically empty and unused by iOS.
- `cache`: Typically empty and unused by iOS.
- `db`: This holds databases that are used by system components, such as time zone information and DHCP leases obtained by the device. If the `stash` subfolder is present, the device is jailbroken.
- `ea`: Typically empty and unused by iOS.
- `empty`: Typically empty and unused by iOS.
- `folder`: Typically empty and unused by iOS.
- `keychains`: This stores the password used within iOS.

- `keybags`: This contains keybags, that are similar to keychains with the difference being that keybags are cleared at reboot while keychains are retained through a reboot. It contains only one file (`systembag.kb`) which stores, for example, escrow keybags used by iTunes. It has been present since iOS 4.0.
- `lib`: This holds state information about an installed application. If the `cydia` subfolder is present, the device is jailbroken.
- `local`: Typically empty and unused by iOS.
- `lock`: Typically empty and unused by iOS.
- `log`: This stores some of the system logs (for example; `asl`, `revision`, `ecc`.)
- `logs`: This stores other system logs (for example, CrashReporter).
- `Managed Preferences`: This stores files related to Microsoft Exchange through ActiveSync.
- `mobile`: This contains the home folder for a mobile user and holds the most useful information from a forensics perspective.
- `MobileDevice`: This contains provisioning profiles, typically used for the development stage.
- `MobileSoftwareUpdate`: This is used for over-the-air updates and it has been present since iOS 5.0.
- `msgs`: Typically empty and unused by iOS.
- `networkd`: This contains files used by networkd, a daemon related to networking services.
- `preferences`: This holds some system configuration.
- `root`: This contains the `home` folder for a mobile user.
- `run`: This contains files used only at run time, typically by lockdownd.
- `spool`: Typically empty and unused by iOS.
- `tmp`: This contains temporary files and its content is often not preserved between device reboots.
- `vm`: Empty and unused by iOS and doesn't support swap files and hibernation.
- `wireless`: This contains data related to phone services provided by the device carrier.

Finally, it is useful to point out that the iDevices use the Property List and SQLite databases as data and configuration containers.

The property list file

The `property list` files (also known as `plist`) are used by Apple for the management of the configuration of the operating system and key applications. Typically, these are simple text files formatted in XML. In most cases, a `plist` file contains the text strings and Boolean values; in some cases, it can contain data encoded in the binary format, as shown in the following screenshot. Although they can be easily analyzed using a simple text editor, it is more convenient to browse the hierarchical structure through a dedicated reader. A complete description of the plist format is beyond the scope of this book, but a complete reference can be found online at `https://developer.apple.com/legacy/library/documentation/Darwin/Reference/ManPages/man5/plist.5.html`



Key	Type	Value
▼Root	Dictionary	(1 item)
▼Zombielets	Array	(1 item)
▼Item 1	Dictionary	(3 items)
▼Zebras	Array	(154 items)
▼Item 1	Dictionary	(2 items)
▼_atts	Dictionary	(4 items)
ElephantHandler	String	Hannibal
sharkID	Number	21633
alternateCorduroy	Boolean	☐
FreakyStyley	Boolean	☑
_name	String	Tofurkey, Joe
▶Item 2	Dictionary	(2 items)

In the Mac environment, it is possible to install the freeware tool, **Property List Editor**, developed by Apple. It can be downloaded from the website of the XCode development platform (`https://developer.apple.com/xcode/`).

In a Windows environment, we can use**plist Editor for Windows** (`http://www.icopybot.com/plist-editor.htm`).

SQLite databases

The iOS devices use SQLite databases to store information and user data. A complete description of the plist format is beyond the scope of this book, but a complete reference can be found online at `https://www.sqlite.org/`. The analysis of these files requires a minimum knowledge of the SQL commands for the selection of data; however, there are several free software options that can interpret and easily display the data in a database. An example of cross-platform software is **SQLite Database Browser** (`http://sqlitebrowser.org/`), which allows us to visualize the structure of the database and to navigate within the data, as shown in the following screenshot:

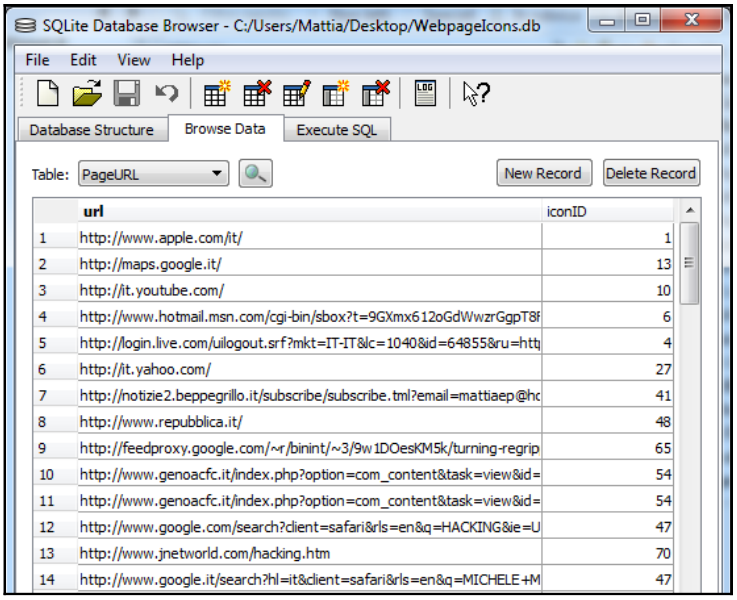


Table: PageURL

	url	iconID
1	http://www.apple.com/it/	1
2	http://maps.google.it/	13
3	http://it.youtube.com/	10
4	http://www.hotmail.msn.com/cgi-bin/sbox?t=9GXmx612oGdWwzrGgpT8f	6
5	http://login.live.com/uilogout.srf?mkt=IT-IT&lc=1040&id=64855&ru=htt	4
6	http://it.yahoo.com/	27
7	http://notizie2.beppegrillo.it/subscribe/subscribe.tml?email=mattiaep@hc	41
8	http://www.repubblica.it/	48
9	http://feedproxy.google.com/~r/binint/~3/9w1DOesKM5k/turning-regrip	65
10	http://www.genoacfc.it/index.php?option=com_content&task=view&id=	54
11	http://www.genoacfc.it/index.php?option=com_content&task=view&id=	54
12	http://www.google.com/search?client=safari&rls=en&q=HACKING&ie=U	47
13	http://www.jnetworld.com/hacking.htm	70
14	http://www.google.it/search?hl=it&client=safari&rls=en&q=MICHELE+M	47

In a Windows environment, it is also advisable to use the software**SQLite Expert** (available in both personal and commercial licenses at `http://www.sqliteexpert.com/`).

Summary

This chapter illustrated the features of interest for iOS devices during mobile forensic activities. In particular, it introduced the different models with guidance on recognition techniques based on the model number or hardware model number. It also contained an introduction to the iOS operating system with particular reference to the filesystem (HFSX), the partitions (system and data), and the main data structures (`property list` files and SQLite databases). These topics are the basics for forensic activity on an iDevice and will be useful inorder to understand the content of the next chapter, where you will learn various tools and techniques that you can use when dealing with acquisition and analysis.

Self-test questions

Q1. What is the latest supported version of iOS for iPhone 4?

1. iOS 5.1.1
2. iOS 6.1.2
3. iOS 7.1.2
4. iOS 8.1.2
5. iOS 9.0.2

Q2. Which are the model numbers associated with iPhone 6s?

1. A1522 and A1524
2. A1549 and A1586
3. A1633 and A1688
4. A1428 and A1429

Q3. What filesystem does iOS use?

1. NTFS
2. EXT3
3. HFS+
4. HFSX

Q4. What metafile is used to keep information on files and folders in iOS filesystem?

1. Volume Header
2. Allocation
3. Catalog
4. Extent

Q5. What is the default root user password?

1. apple
2. iphone
3. leopard
4. alpine

Q6. What kind of file is mostly used to keep iOS configuration?

1. Text
2. Json
3. Plist
4. HTML

3
Evidence Acquisition from iDevices

The purpose of this chapter is to introduce techniques and tools used for the acquisition of data from an iDevice. In the first part of the chapter, we will go through the boot process, the data security features, and the encryption used by Apple. The second part deals with the different acquisition methods **Apple File Conduit** (**AFC**) protocol, iTunes backup, Apple File Relay protocol, and physical), providing a description of state-of-the-art techniques for cracking or bypassing the lock code. Finally, in the last part, we will introduce the concept of jailbreaking, which is useful for the physical acquisition of the latest devices.

iOS boot process and operating modes

The boot process of an iOS device is composed of three steps: **Low Level Bootloader** (**LLB**), **iBoot**, and **iOS kernel**. To guarantee the integrity of the different components, all the steps involved in the boot process are signed. The signature for LLB is verified by the Apple Root CA public key, contained in the Boot ROM code. Then, LLB verifies and executes iBoot, which then verifies and executes the iOS kernel. Consequently, all the components are signed by Apple. There are a lot of studies, papers, and books related to the iOS boot process and on how to overcome the protection implemented by Apple (you can find all the details in `Appendix A`, *References*). We suggest that you read the latest version of the Apple paper *iOS Security*, *Apple*, *May 2016*.

From the point of view of a forensic analyst, it is important to know that iDevices can operate in the following three different modes:

- **Normal**: This mode is the traditional iOS user interface.

- **Recovery**: This mode is used to perform activation and upgrades on an iDevice. It can be activated by holding down the Home button on a powered off device and connecting it to a computer via the USB cable.
- **Device firmware upgrade** (**DFU**): This mode is used by an iDevice during the iOS upgrades and when one of the processes in verification boot chain fails. It can be activated by holding down the Home and the Power button together (with the device powered on or off) for 10 seconds, and then it is necessary to release the Power button and hold the Home button for 10 seconds more.

Both Recovery and DFU modes are really useful for the physical acquisition of iDevices, as we will show later in the dedicated section.

iOS data security

A complete description of iOS data security is out of the scope of this book, but we wish to give you just an overview (taken from the Apple paper *iOS Security* and from Christian D'Orazio's thesis; refer to `Appendix A`, *References*) of the hardware and software security features.

Hardware security features

Every iDevice, starting from iPhone 3GS, has a dedicated AES 256-bit crypto engine built in between the flash storage and the main system memory. The purposes of this processor are to accelerate the encryption and decryption operations and to protect user data so that it remains encrypted on the device's flash memory. A **unique ID** (**UID**) is associated with each device and allows data to be cryptographically tied to a particular device. The UID cannot be read directly and it is used as an AES 256-bit key to generate encryption keys that protect user data. These encryption keys, known as **EMF** and **Dkey** (**Class D Key**), are stored in a specific area of the flash memory called **PLOG block** (or **Effaceable Storage**). When the device deletes this area it makes the whole volume unreadable and the content is completely and definitely encrypted, with no way to recover it.

File data protection

As described by Apple in the paper *iOS Security* (refer to `Appendix A`, *References*):

"In addition to the hardware encryption features built into iOS devices, Apple uses a technology called Data Protection to further protect data stored in flash memory on the device."

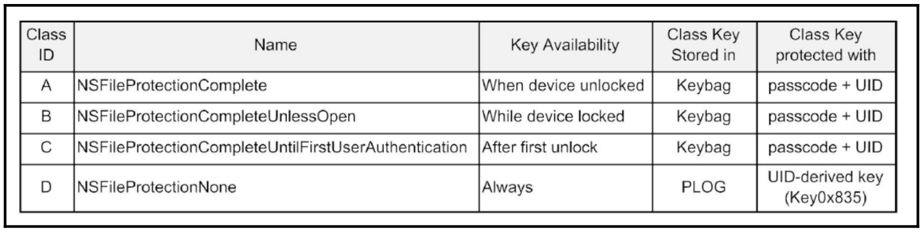
Apple implements an encrypted HFS+ volume, in which each file is assigned to a class, depending on the type of data and the security level required. The paper also states that:

"Every time a file on the data partition is created, Data Protection creates a new 256-bit key (the per-file key) and gives it to the hardware AES engine, which uses the key to encrypt the file as it is written to flash memory using AES CBC mode."

The per-file key is then wrapped with the key of the class to which the file belongs. The wrapped per-file key is stored in the `cprotect` attribute, which is part of the file's metadata contained in the `Attributes` file. The paper further states that:

"When a file is opened, its metadata is decrypted with the filesystem key, revealing the wrapped per-file key and a notation on which class protects it. The per-file key is unwrapped with the class key, then supplied to the hardware AES engine, which decrypts the file as it is read from flash memory."

It is important to note that the filesystem key can be erased, and in that case the content of every file becomes definitely unreadable. There are four basic classes that use different policies to determine when file content is accessible and where the class keys are stored. With the exception of the Dkey, all class keys are stored in the `Keybag` system, which is a file that contains master keys for each one of the classes available, as shown in the following screenshot:



Class ID	Name	Key Availability	Class Key Stored in	Class Key protected with
A	NSFileProtectionComplete	When device unlocked	Keybag	passcode + UID
B	NSFileProtectionCompleteUnlessOpen	While device locked	Keybag	passcode + UID
C	NSFileProtectionCompleteUntilFirstUserAuthentication	After first unlock	Keybag	passcode + UID
D	NSFileProtectionNone	Always	PLOG	UID-derived key (Key0x835)

Class D offers the lowest security level because the Dkey is not derived from the passcode but wrapped in the `PLOG` area with a value (`Key0x835`) that can be retrieved by communicating with the kernel. From a forensics point of view, it is important to note that all the files created by a native iOS application, except e-mail messages and related attachments, belong to Class D. It means that all the cryptographic keys required to decrypt a file can be retrieved without knowing or cracking the passcode.

Unique device identifier

Every single iDevice produced is identified by a **Unique Device ID** (**UDID**), as is explained in The iPhone Wiki (`http://theiphonewiki.com/wiki/UDID`). It can be calculated as the **SHA-1** hash of a particular 60- or 59-character long string that can be obtained as follows:

- An 11-character or 12-character long (on newer devices) serial number (exactly like it is written in the **Settings** app)
- A 15-character long IMEI number without spaces (on older devices), empty string for iPod touch and the Wi-Fi model iPad devices, or a 13-character ECID in decimal with no leading zeroes (on newer devices)

The ECID is the Electronic Chip ID. For more information, refer to `https://theiphonewiki.com/wiki/ECID`.

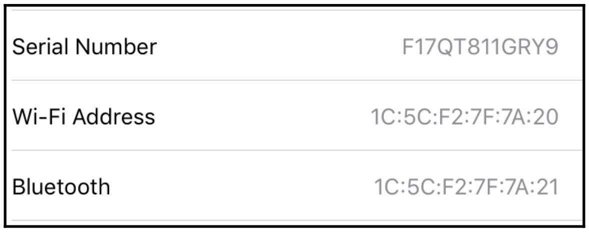
- A 17-character long Wi-Fi MAC address (letters in lowercase, including colons). For the iPod touch first generation, use `00:00:00:00:00:00`
- A 17-character long Bluetooth MAC address (letters in lowercase, including colons)

Case study – UDID calculation on iPhone 6s

On iPhone 6s, the UDID is calculated as follows:

SHA1 (serial number + ECID converted to decimal + Wi-Fi MAC address + Bluetooth MAC address)

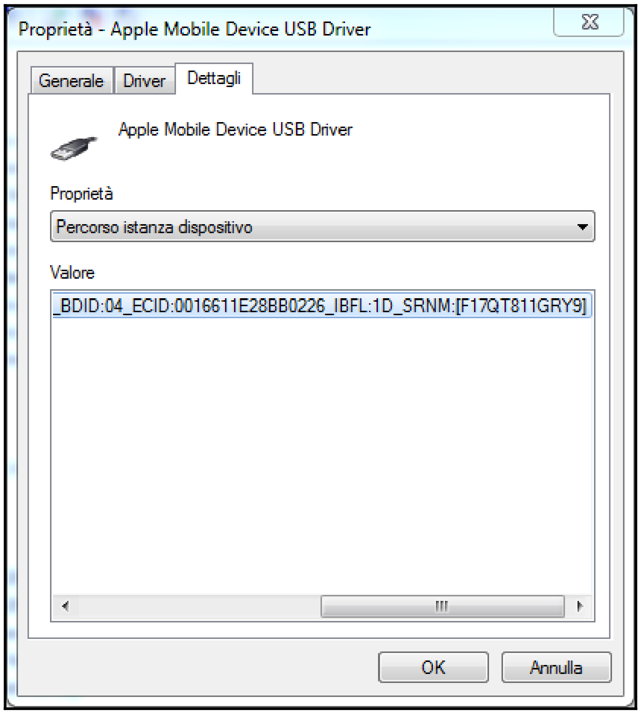
If the device is unlocked, the serial number, Wi-Fi MAC address, and Bluetooth can be obtained by tapping **About** from **Settings** | **General** on the device's main screen, as shown in the following screenshot:



Serial Number	F17QT811GRY9
Wi-Fi Address	1C:5C:F2:7F:7A:20
Bluetooth	1C:5C:F2:7F:7A:21

The ECID can be obtained as follows:

1. Put the device in Recovery mode.
2. Open Windows **Device Manager**, navigate to **Universal Serial Bus controllers** | **Apple Mobile Device USB Driver**, right-click on it, and select **Properties**.
3. Click on **Details**, search and select **Device Instance Path** in the drop-down menu, and copy the text to a text file.
4. On a Mac OS X, navigate to **System Information** | **System Report** and look in the USB entry under **Hardware**.



Apple Mobile Device USB Driver

Proprietà

Percorso istanza dispositivo

Valore

_BDID:04_ECID:0016611E28BB0226_IBFL:1D_SRNM:[F17QT811GRY9]

In this example we have the following entries:

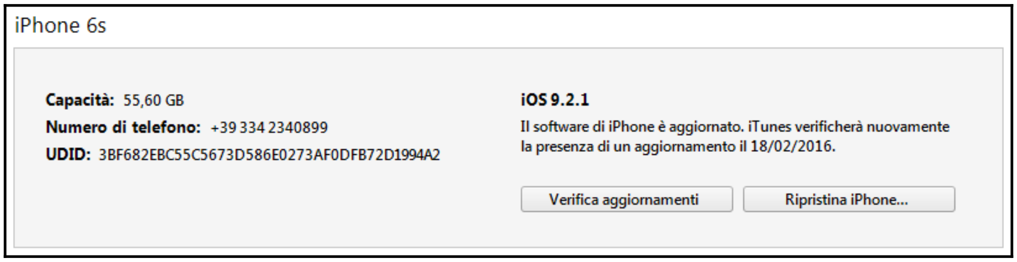
- **Serial number**: `F17QT811GRY9`
- **ECID**: `0016611E28BB0226`
- **Wi-Fi MAC address**: `1C:5C:F2:7F:7A:20`
- **Bluetooth MAC address**: `1C:5C:F2:7F:7A:21`

Before calculating the UDID, we need to convert the hex value for ECID to a decimal number, so `16611E28BB0226` corresponds to `6299231647892006`. Moreover the letters in the Wi-Fi MAC address and in the Bluetooth MAC address must be converted to lower case.

The UDID can be calculated as follows:

```
SHA1(F17QT811GRY96299231647892006 1c:5c:f2:7f:7a:201c:5c:f2:7f:7a:21) =
3bf682ebc55c5673d586e0273af0dfb72d1994a2
```

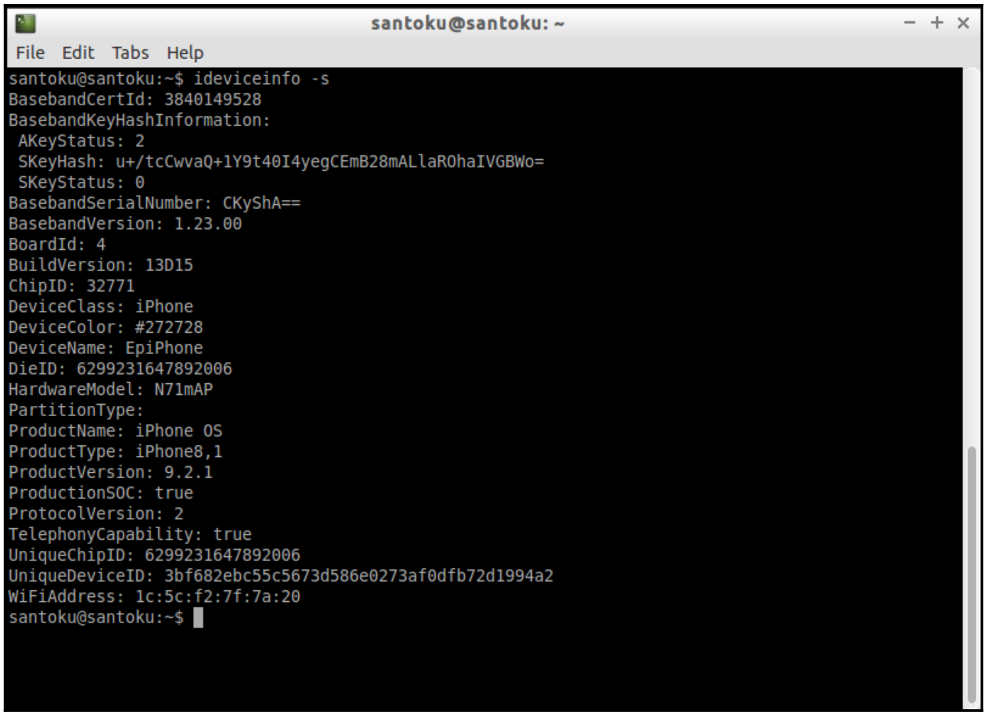
The calculated value can be verified using iTunes, after connecting the device to the computer, as shown in the following screenshot:


iPhone 6s

Capacità: 55,60 GB
Numero di telefono: +39 334 2340899
UDID: 3BF682EBC55C5673D586E0273AF0DFB72D1994A2

iOS 9.2.1
Il software di iPhone è aggiornato. iTunes verificherà nuovamente la presenza di un aggiornamento il 18/02/2016.

Otherwise, the UDID can also be verified using the `ideviceinfo` tool introduced in `Chapter 2`, *Introduction to iOS Devices*, as shown in the following screenshot:



```
santoku@santoku:~$ ideviceinfo -s
BasebandCertId: 3840149528
BasebandKeyHashInformation:
 AKeyStatus: 2
 SKeyHash: u+/tcCwvaQ+1Y9t40I4yegCEmB28mALlaROhaIVGBWo=
 SKeyStatus: 0
BasebandSerialNumber: CKyShA==
BasebandVersion: 1.23.00
BoardId: 4
BuildVersion: 13D15
ChipID: 32771
DeviceClass: iPhone
DeviceColor: #272728
DeviceName: EpiPhone
DieID: 6299231647892006
HardwareModel: N71mAP
PartitionType:
ProductName: iPhone OS
ProductType: iPhone8,1
ProductVersion: 9.2.1
ProductionSOC: true
ProtocolVersion: 2
TelephonyCapability: true
UniqueChipID: 6299231647892006
UniqueDeviceID: 3bf682ebc55c5673d586e0273af0dfb72d1994a2
WiFiAddress: 1c:5c:f2:7f:7a:20
santoku@santoku:~$ 
```

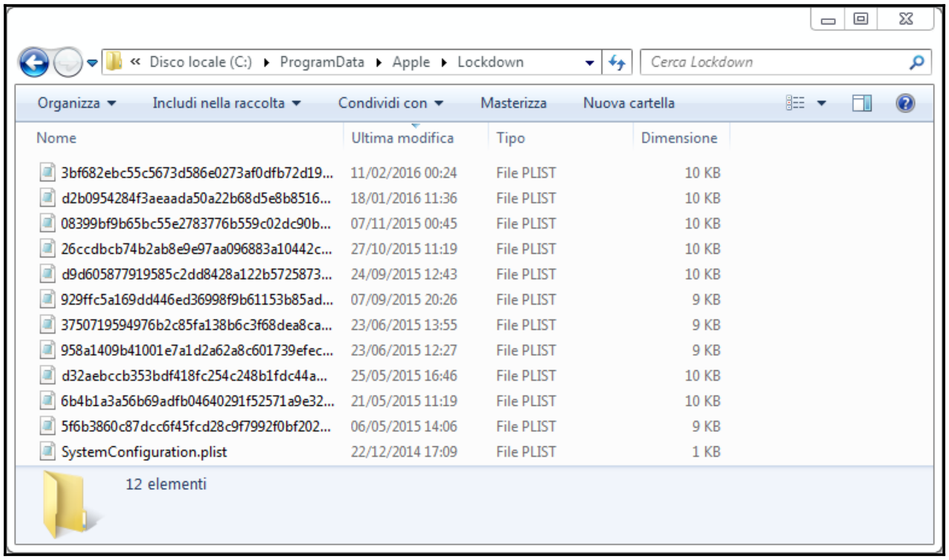
Lockdown certificate

The first time you connect an unlocked iDevice to a computer and run the iTunes software, a pairing/sync certificate, known as a **lockdown** certificate, is created on the computer's hard drive. Depending on the operating system in which iTunes is installed, lockdown certificates are stored in the following folders:

- **Windows 7/8/10**: `C:\Program Data\Apple\Lockdown`
- **Windows Vista**: `C:\Users\[username]\AppData\roaming\Apple Computer\Lockdown`
- **Windows XP**: `C:\Documents and Settings\[username]\Application Data\Apple Computer\Lockdown`
- **Mac OS X**: `/var/db/lockdown`

In older version of Mac OS X lockdown certificates are stored in `/Users/[username]/Library/Lockdown/`.

Within these paths, there is a lockdown certificate for each device that was ever connected to the computer. The certificate is a `plist` file called `<UDID>.plist`, where `UDID` corresponds with the unique identifier of the iDevice, as shown in the following screenshot:

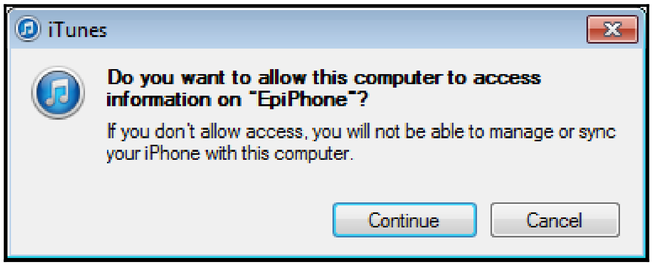


Nome	Ultima modifica	Tipo	Dimensione
3bf682ebc55c5673d586e0273af0dfb72d19...	11/02/2016 00:24	File PLIST	10 KB
d2b0954284f3aeaada50a22b68d5e8b8516...	18/01/2016 11:36	File PLIST	10 KB
08399bf9b65bc55e2783776b559c02dc90b...	07/11/2015 00:45	File PLIST	10 KB
26ccdbcb74b2ab8e9e97aa096883a10442c...	27/10/2015 11:19	File PLIST	10 KB
d9d605877919585c2dd8428a122b5725873...	24/09/2015 12:43	File PLIST	10 KB
929ffc5a169dd446ed36998f9b61153b85ad...	07/09/2015 20:26	File PLIST	9 KB
3750719594976b2c85fa138b6c3f68dea8ca...	23/06/2015 13:55	File PLIST	9 KB
958a1409b41001e7a1d2a62a8c601739efec...	23/06/2015 12:27	File PLIST	9 KB
d32aebccb353bdf418fc254c248b1fdc44a...	25/05/2015 16:46	File PLIST	10 KB
6b4b1a3a56b69adfb04640291f52571a9e32...	21/05/2015 11:19	File PLIST	10 KB
5f6b3860c87dcc6f45fcd28c9f7992f0bf202...	06/05/2015 14:06	File PLIST	9 KB
SystemConfiguration.plist	22/12/2014 17:09	File PLIST	1 KB

12 elementi

Once the certificate for a specific device has been generated, it remains valid until the user resets the device to factory settings. If the device is turned on and has been unlocked at least once in the last 48 hours (from iOS 8), the certificate can be used to bypass passcode to both obtain a backup and extract files through the AFC service. Of fundamental importance to the forensic acquisition of data is the fact that the certificate can be copied to another machine.

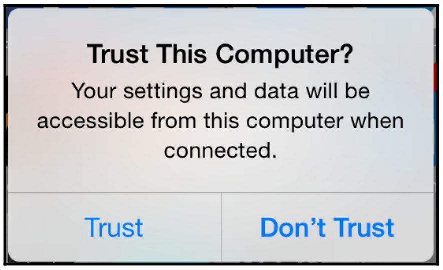
Starting from iOS 7.0, when you connect a device, two pop-up authorizations are displayed. The first popup appears on the computer in iTunes and it asks the user to click on **Continue**.



Do you want to allow this computer to access information on "EpiPhone"?

If you don't allow access, you will not be able to manage or sync your iPhone with this computer.

The second popup appears on the iDevice screen once unlocked, and requires the user to click on the **Trust** button to allow pairing with the computer.



Trust This Computer?

Your settings and data will be accessible from this computer when connected.

Search and seizure

If you have to deal with a search and seizure of an iDevice, it is really important to perform some steps. We have three scenarios: device turned off, device turned on and unlocked at the time of search and seizure, and device turned on and locked at the time of search and seizure:

1. If the device is turned on and locked, do the following:

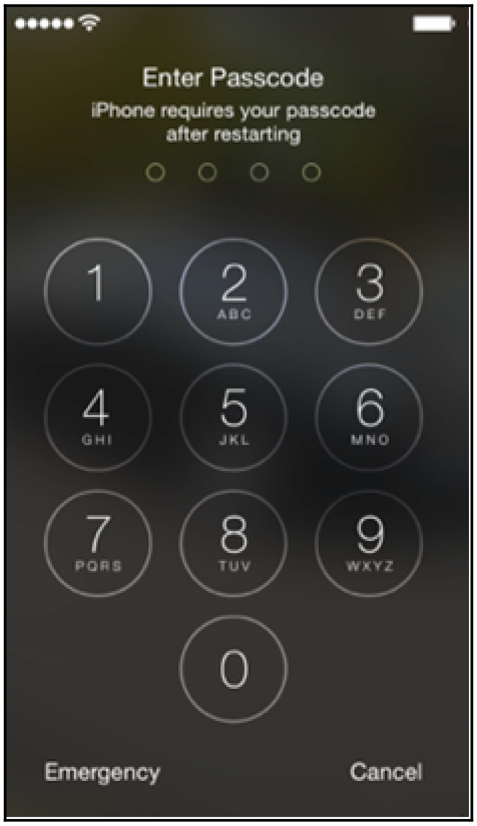
   If the passcode is set, acquire the content from the device as soon as possible or keep the device on and in charge.

   If the passcode is not set, turn it off.

2. If the device is off, leave it off.
3. Set **Auto-Lock** to **Never** from **Settings** | **General** | **Auto-Lock**.
4. Check whether the passcode is set or not from **Settings** | **Passcode**.

5. If the device is on and unlocked, turn on **Airplane Mode** in **Settings:**

   Check if the device displays the **iPhone requires your passcode after restarting** message, as shown in the following screenshot:


Enter Passcode

iPhone requires your passcode after restarting

6. Turn on **Airplane Mode** using Apple**Control Center**, which can be activated with a single swipe, even with locked device if such access has not been disabled by the user.
7. Keep the phone on and charging.
8. Seize any computer that could have been used to synchronize or simply authorize the iDevice, because there you can possibly find a lockdown certificate that will allow access to the data.
9. Acquire the data as soon as possible because the lockdown certificate is valid only if the device remains on and in any case no more than 48 hours.

iOS device acquisition

Once you have identified the specific model that you need to acquire, it becomes important to understand the best technique to use. The type of acquisition depends basically on the following seven parameters:

- Device model
- iOS version
- Passcode (not set, simple passcode, or complex passcode)
- Device status (on or off)
- Availability of a pairing certificate
- Presence of a backup password
- Device jailbroken or not

Nowadays, in the forensic community the following four techniques are used to access data stored on iDevices:

- **Apple File Conduit**: This technique consists of a direct interaction with a powered on device, typically through non-forensic software, and permits obtaining a reduced set of the filesystem
- **iTunes Backup**: This technique consists of a partial filesystem acquisition through the iTunes backup feature or using a forensic acquisition tool that uses the iTunes libraries
- **Advanced logical**: This technique is based on Apple File Relay and was introduced for the first time by the researcher Jonathan Zdziarski
- **Physical**: This technique generates a traditional forensic image for both the system and data partition

Apple File Conduit acquisition

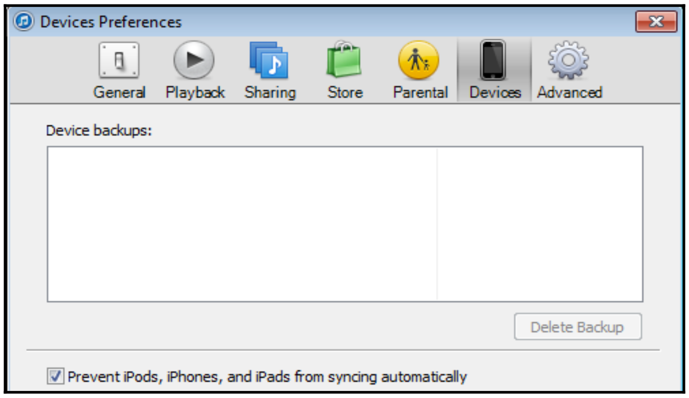
The acquisition through the AFC service can be carried out with all iDevices, regardless of the operating system version. It requires that the device is not protected with a passcode, the passcode is known, or the analyst has a valid lockdown certificate. The conditions and the kind of data that you can get with this kind of acquisition have changed a lot through the various versions of iOS.

Starting from iOS 9, if the device is unlocked or with a valid lockdown certificate, it is only possible to extract multimedia information (photos, videos, music, and so on) and the related configuration files (for example, the iTunes library, the database of the Photo application, and so on). Moreover, in previous versions up to iOS 8.3, it was also possible to access the data folders of third-party applications, with the phone unlocked or with a valid lockdown certificate. It is also interesting to note that this method allows us to extract the `Crash Reports` folder from the device, which contains the error logs generated by the operating system and installed applications and the live System Log. Neither of these will be present on an acquisition made through iTunes backup, as will be explained in `Chapter 6`, *Analyzing iOS Devices*, and in some cases, these files can contain very important information. It is therefore advisable to take this into account while choosing the acquisition method.

To make that kind of acquisition, you can use various types of software known as iDevice browsers.

This activity is performed with a non-forensic tool that also permits writing operations, so the analyst must operate very carefully to avoid accidental erasure.

The most used tools on Windows and Mac for this type of acquisition are **iFunBox**, **iTools**, **iBackup Bot**, **iMazing**, **iExplorer**, and **WonderShare Dr.Fone**. These tools require the installation of an updated version of iTunes because they use its libraries to communicate with the device. Before connecting the device to your computer, you should ensure that in **iTunes** | **Preferences** | **Devices**, the **Prevent iPods, iPhones, and iPads from syncing automatically** option is enabled, as shown in the following screenshot:

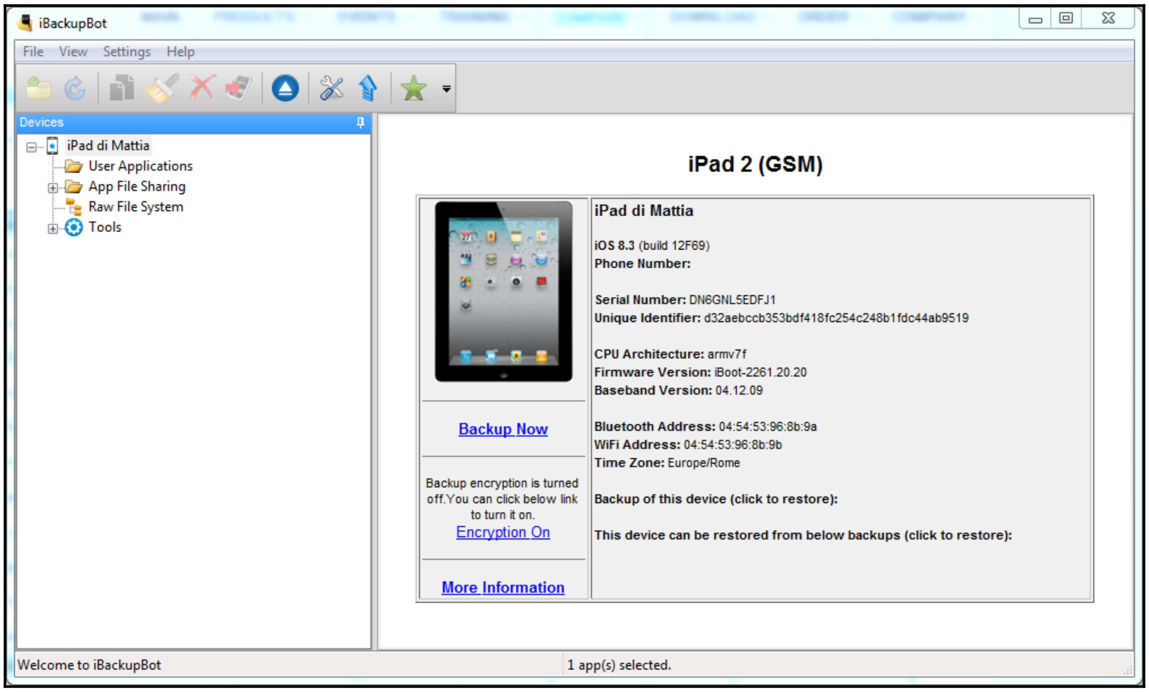


☑ Prevent iPods, iPhones, and iPads from syncing automatically

Case study – AFC acquisition with iBackupBot

iBackupBot is a commercial tool for Windows that allows access to connected iOS devices to perform both direct extraction via AFC protocol and the generation of backups in iTunes format. The trial version allows loading and extracting information from the device.

Once the software has been downloaded, installed, and run, simply connect the device to acquire and wait for the related information to load. In the following screenshot you can see an example related to an **iPad 2 (GSM)** running iOS 8.3, which shows the main characteristics of the device (**Serial Number**, **Unique Identifier**, **Bluetooth Address**, **Wi-Fi Address**, and so on) and the backup encryption configuration status (that is, if it's enabled or not):



iPad 2 (GSM)

iPad di Mattia

iOS 8.3 (build 12F69)
Phone Number:

Serial Number: DN6GNL5EDFJ1
Unique Identifier: d32aebccb353bdf418fc254c248b1fdc44ab9519

CPU Architecture: armv7f
Firmware Version: iBoot-2261.20.20
Baseband Version: 04.12.09

Bluetooth Address: 04:54:53:96:8b:9a
WiFi Address: 04:54:53:96:8b:9b
Time Zone: Europe/Rome

Backup of this device (click to restore):

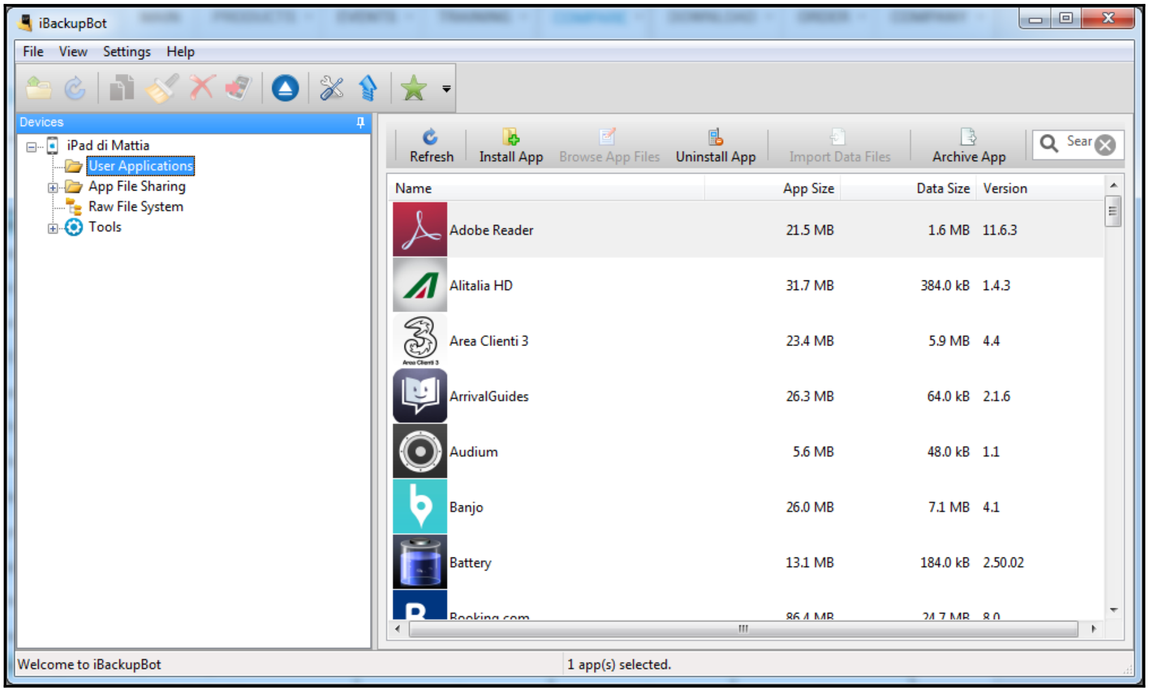
This device can be restored from below backups (click to restore):

Backup encryption is turned off.You can click below link to turn it on.
Encryption On

Welcome to iBackupBot

1 app(s) selected.

Clicking on **User Applications** shows the list of all applications installed by the user, reporting the size of the application and of its data, plus the version that is currently installed:



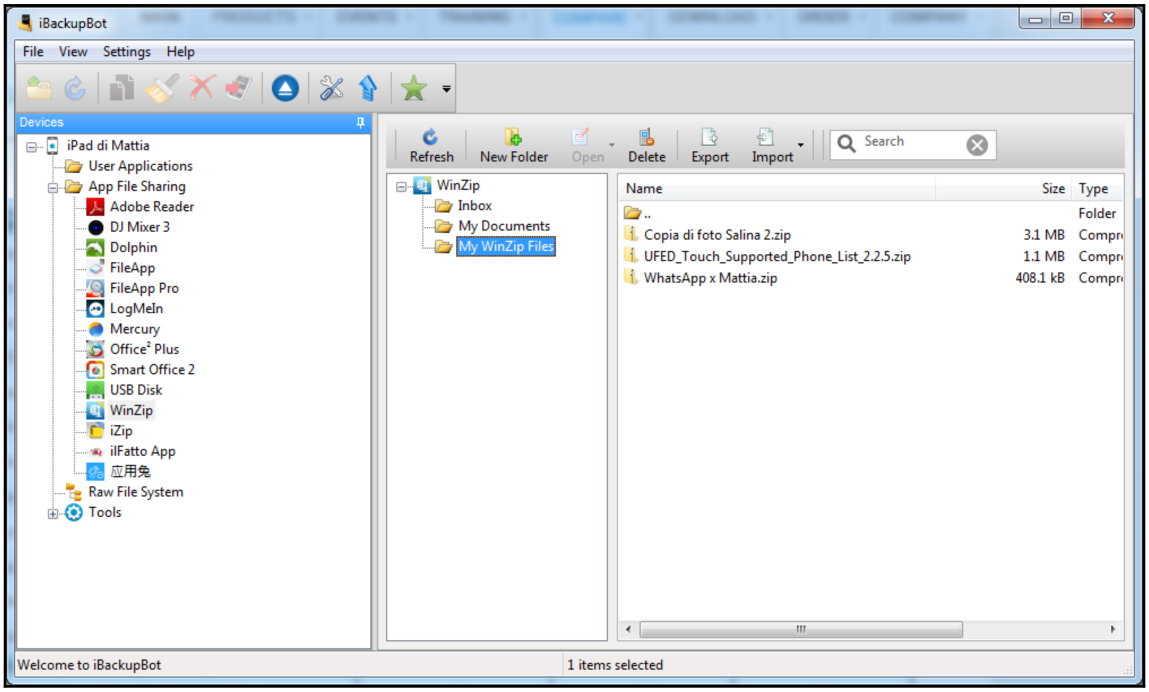
Name	App Size	Data Size	Version
Adobe Reader	21.5 MB	1.6 MB	11.6.3
Alitalia HD	31.7 MB	384.0 kB	1.4.3
Area Clienti 3	23.4 MB	5.9 MB	4.4
ArrivalGuides	26.3 MB	64.0 kB	2.1.6
Audium	5.6 MB	48.0 kB	1.1
Banjo	26.0 MB	7.1 MB	4.1
Battery	13.1 MB	184.0 kB	2.50.02
Booking.com	86.4 MB	24.7 MB	8.0

Welcome to iBackupBot

1 app(s) selected.

The **App File Sharing** option provides access to application data whose developers have decided to share data. This feature is used to allow, typically through iTunes, the exchange of files between the device and a computer. All shared files from the applications can be extracted directly from the device. The functionality is often used in applications where the user needs to create and/or modify the contents (for example, the Office Suites app) or manage files (for example, the Compression app).

The following screenshot shows an example on the WinZip application. By right-clicking on one or more files you can export them to a folder on your computer. Pay attention to the fact that this access is read-write, and the graphical interface provides also a **New Folder** button that allows the creation of content on the device.

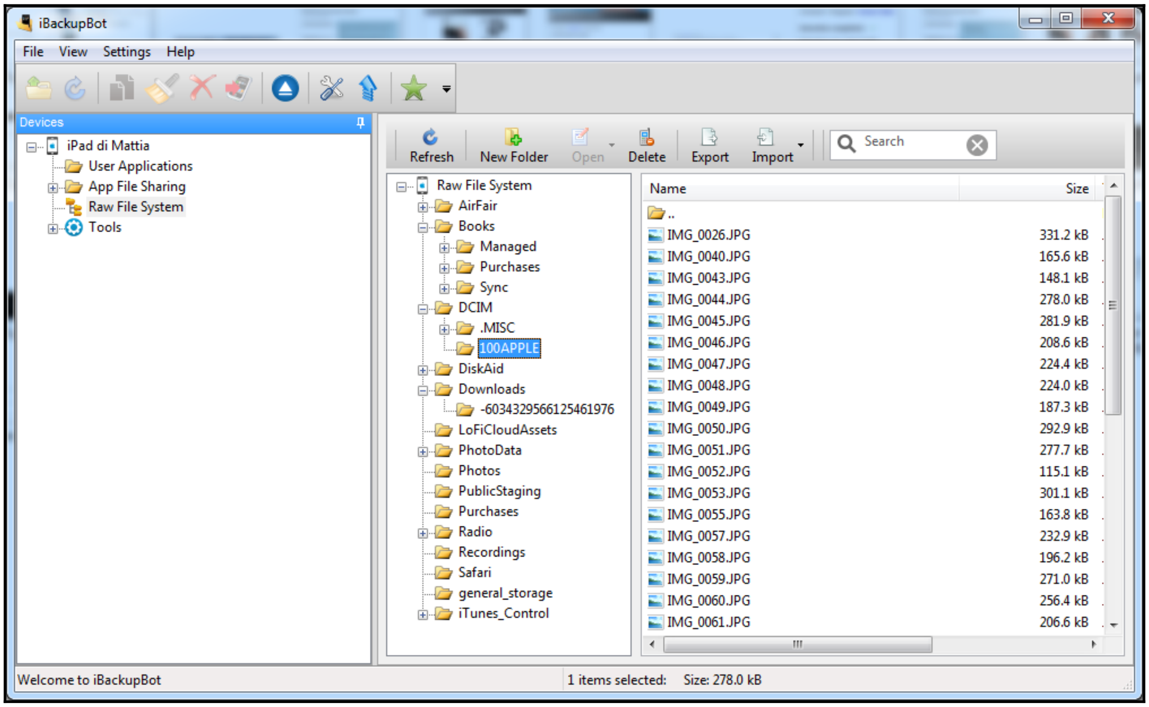


Name	Size	Type
..		Folder
Copia di foto Salina 2.zip	3.1 MB	Compr
UFED_Touch_Supported_Phone_List_2.2.5.zip	1.1 MB	Compr
WhatsApp x Mattia.zip	408.1 kB	Compr

Welcome to iBackupBot

1 items selected

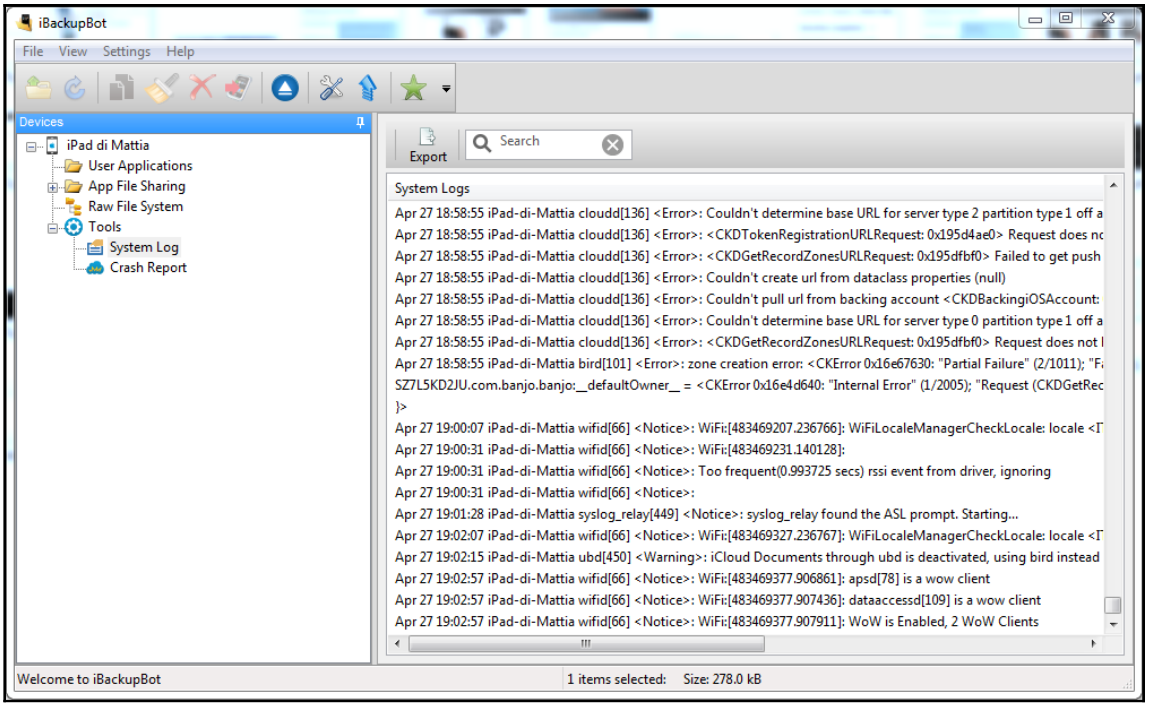
The **Raw File System** option allows you to navigate within the part of the device's filesystem that is made available through the AFC protocol. As already mentioned, it mainly includes media data (`DCIM` folder), books loaded into the iBooks app (`Books` folder), iTunes library (`iTunes_Control` folder), and files downloaded by the user or the applications (`Downloads` folder). Similar to the previous option, files and folders can be extracted from the device to the computer.



Name	Size
..	
IMG_0026.JPG	331.2 kB
IMG_0040.JPG	165.6 kB
IMG_0043.JPG	148.1 kB
IMG_0044.JPG	278.0 kB
IMG_0045.JPG	281.9 kB
IMG_0046.JPG	208.6 kB
IMG_0047.JPG	224.4 kB
IMG_0048.JPG	224.0 kB
IMG_0049.JPG	187.3 kB
IMG_0050.JPG	292.9 kB
IMG_0051.JPG	277.7 kB
IMG_0052.JPG	115.1 kB
IMG_0053.JPG	301.1 kB
IMG_0055.JPG	163.8 kB
IMG_0057.JPG	232.9 kB
IMG_0058.JPG	196.2 kB
IMG_0059.JPG	271.0 kB
IMG_0060.JPG	256.4 kB
IMG_0061.JPG	206.6 kB

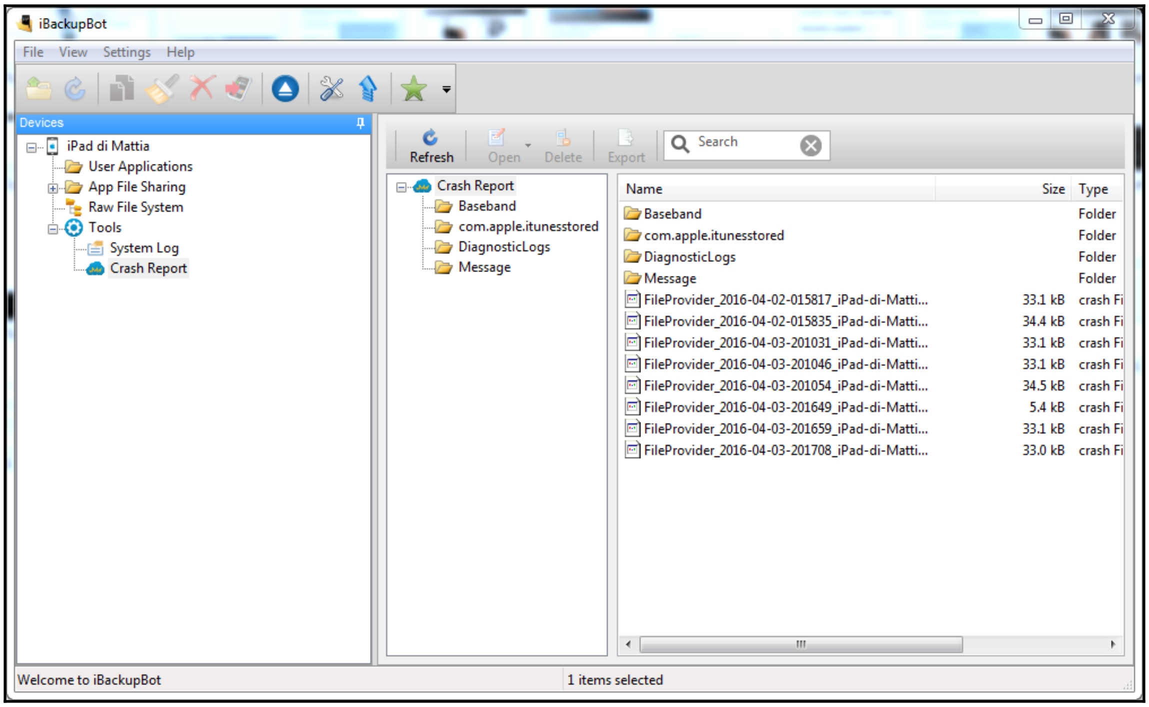
Lastly, the **Tools** option allows the file extraction of the `System Log` and the `Crash Report` folder.

The `System Log` can be exported in text format and contains the latest information on the activity of the device. It is useful if extracted in the immediacy or, in general, to support the information attached to the acquisition reports and demonstrate at least in part the inevitable changes made to the system during extraction.



```
System Logs
Apr 27 18:58:55 iPad-di-Mattia cloudd[136] <Error>: Couldn't determine base URL for server type 2 partition type 1 off a
Apr 27 18:58:55 iPad-di-Mattia cloudd[136] <Error>: <CKDTokenRegistrationURLRequest: 0x195d4ae0> Request does nc
Apr 27 18:58:55 iPad-di-Mattia cloudd[136] <Error>: <CKDGetRecordZonesURLRequest: 0x195dfbf0> Failed to get push
Apr 27 18:58:55 iPad-di-Mattia cloudd[136] <Error>: Couldn't create url from dataclass properties (null)
Apr 27 18:58:55 iPad-di-Mattia cloudd[136] <Error>: Couldn't pull url from backing account <CKDBackingiOSAccount:
Apr 27 18:58:55 iPad-di-Mattia cloudd[136] <Error>: Couldn't determine base URL for server type 0 partition type 1 off a
Apr 27 18:58:55 iPad-di-Mattia cloudd[136] <Error>: <CKDGetRecordZonesURLRequest: 0x195dfbf0> Request does not l
Apr 27 18:58:55 iPad-di-Mattia bird[101] <Error>: zone creation error: <CKError 0x16e67630: "Partial Failure" (2/1011); "F
SZ7L5KD2JU.com.banjo.banjo:__defaultOwner__ = <CKError 0x16e4d640: "Internal Error" (1/2005); "Request (CKDGetRec
}>
Apr 27 19:00:07 iPad-di-Mattia wifid[66] <Notice>: WiFi:[483469207.236766]: WiFiLocaleManagerCheckLocale: locale <I
Apr 27 19:00:31 iPad-di-Mattia wifid[66] <Notice>: WiFi:[483469231.140128]:
Apr 27 19:00:31 iPad-di-Mattia wifid[66] <Notice>: Too frequent(0.993725 secs) rssi event from driver, ignoring
Apr 27 19:00:31 iPad-di-Mattia wifid[66] <Notice>:
Apr 27 19:01:28 iPad-di-Mattia syslog_relay[449] <Notice>: syslog_relay found the ASL prompt. Starting...
Apr 27 19:02:07 iPad-di-Mattia wifid[66] <Notice>: WiFi:[483469327.236767]: WiFiLocaleManagerCheckLocale: locale <I
Apr 27 19:02:15 iPad-di-Mattia ubd[450] <Warning>: iCloud Documents through ubd is deactivated, using bird instead
Apr 27 19:02:57 iPad-di-Mattia wifid[66] <Notice>: WiFi:[483469377.906861]: apsd[78] is a wow client
Apr 27 19:02:57 iPad-di-Mattia wifid[66] <Notice>: WiFi:[483469377.907436]: dataaccessd[109] is a wow client
Apr 27 19:02:57 iPad-di-Mattia wifid[66] <Notice>: WiFi:[483469377.907911]: WoW is Enabled, 2 WoW Clients
```

The `Crash Report` folder can be extracted in a similar way as the previous options. Typically, it contains the information about the last 24-48 hours of device usage, but the amount of information present is not determinable a priori, because it is affected by several parameters. As will be detailed in `Chapter 6`, *Analyzing iOS Devices*, these logs may contain information that cannot be extracted by other methods.



Devices:
- iPad di Mattia
  - User Applications
  - App File Sharing
  - Raw File System
  - Tools
    - System Log
    - Crash Report

Crash Report:
- Baseband
- com.apple.itunesstored
- DiagnosticLogs
- Message

Name	Size	Type
Baseband		Folder
com.apple.itunesstored		Folder
DiagnosticLogs		Folder
Message		Folder
FileProvider_2016-04-02-015817_iPad-di-Matti...	33.1 kB	crash Fi
FileProvider_2016-04-02-015835_iPad-di-Matti...	34.4 kB	crash Fi
FileProvider_2016-04-03-201031_iPad-di-Matti...	33.1 kB	crash Fi
FileProvider_2016-04-03-201046_iPad-di-Matti...	33.1 kB	crash Fi
FileProvider_2016-04-03-201054_iPad-di-Matti...	34.5 kB	crash Fi
FileProvider_2016-04-03-201649_iPad-di-Matti...	5.4 kB	crash Fi
FileProvider_2016-04-03-201659_iPad-di-Matti...	33.1 kB	crash Fi
FileProvider_2016-04-03-201708_iPad-di-Matti...	33.0 kB	crash Fi

Welcome to iBackupBot
1 items selected

iTunes backup

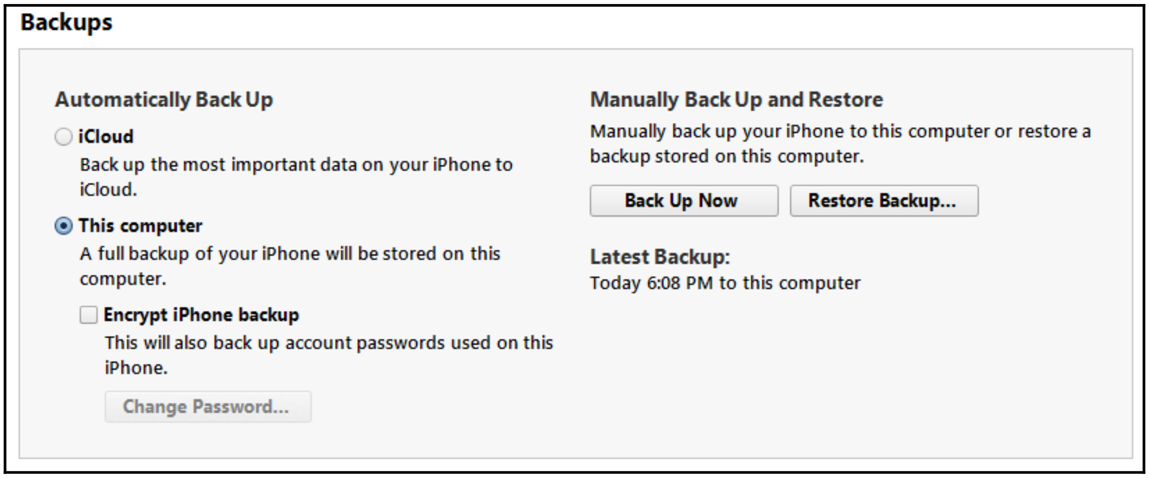
iTunes backup acquisition allows the analyst to recover more information than the AFC acquisition and in a more forensics way as it creates a backup for the device without altering any data. Regarding the passcode, the conditions are similar to direct acquisition: the analyst must know the passcode or have a valid lockdown certificate to perform this kind of acquisition. Before connecting the device, you also need to disable automatic syncing in the iTunes software. This acquisition can be performed in two ways, *using iTunes* or *using forensic software*.

Acquisition with iTunes

The acquisition through iTunes can be done in a very simple way using the backup function of the device. Once you start iTunes, you need to click on the name of the device to access detailed information. At this point, you need to check how the device is configured in relation to the backup operation. There may be the following three cases:

- The device is configured to perform a local backup not protected by a password
- The device is configured to perform a local backup with a password previously set by the device owner
- The device is configured to backup to iCloud

In the first two cases, simply click on the **Back Up Now** button to start the backup on the computer, as shown in the following screenshot:



Backups

Automatically Back Up

iCloud

Back up the most important data on your iPhone to iCloud.

[✓ This computer]

A full backup of your iPhone will be stored on this computer.

Encrypt iPhone backup

This will also back up account passwords used on this iPhone.

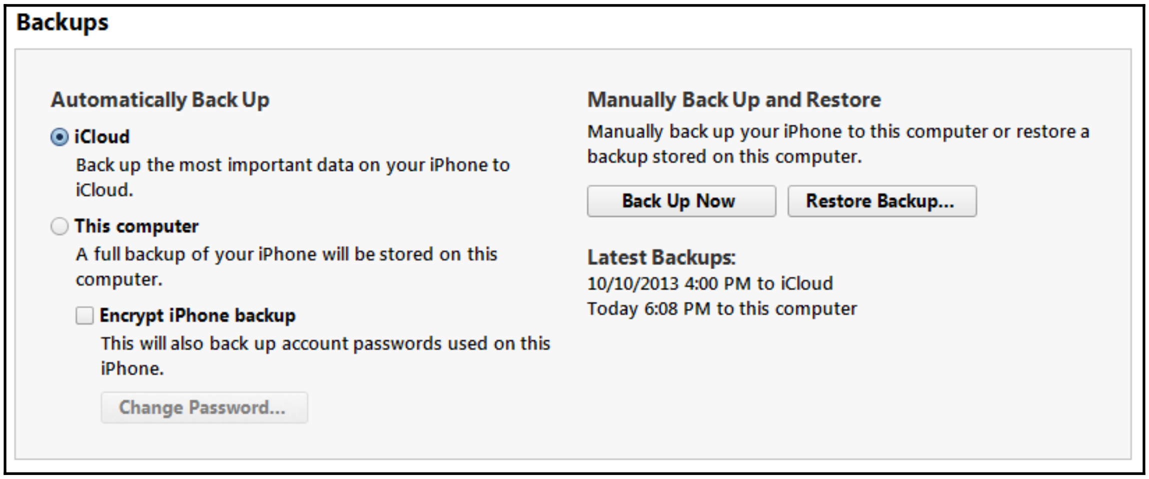
Manually Back Up and Restore

Manually back up your iPhone to this computer or restore a backup stored on this computer.

Latest Backup:
Today 6:08 PM to this computer

If the user has not chosen a password, the created backup can be analyzed with various tools. Otherwise, the analyst needs to crack the backup password before starting the analysis. Both the password cracking and backup structures will be discussed in `Chapter 4`, *Evidence Acquisition and Analysis from iTunes Backup*.

In the third case, before starting the backup, the analyst must change the option from **iCloud** to **This computer**. In this way, the backup will be performed locally and will not overwrite any existing data present in the previous backups on iCloud. The data acquisition from iCloud will be explained in `Chapter 5`, *Evidence Acquisition and Analysis from iCloud*.


Backups

Automatically Back Up

(●) iCloud
Back up the most important data on your iPhone to iCloud.

( ) This computer
A full backup of your iPhone will be stored on this computer.

[ ] Encrypt iPhone backup
This will also back up account passwords used on this iPhone.

Manually Back Up and Restore
Manually back up your iPhone to this computer or restore a backup stored on this computer.

Latest Backups:
10/10/2013 4:00 PM to iCloud
Today 6:08 PM to this computer

Acquisition with forensic tools

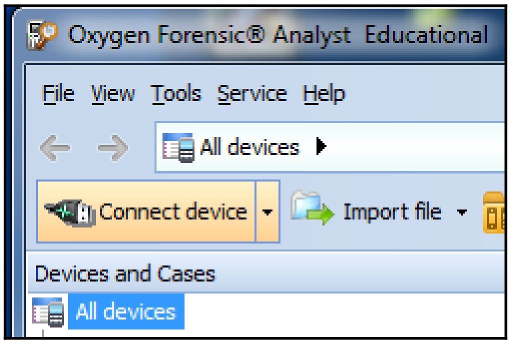
In the market, there are various forensic tools that can perform backup acquisition, for example, Cellebrite UFED 4PC/UFED Touch/UFED Physical Analyzer, Oxygen Forensic® Starter/Analyst/Detective, Mobile Phone Examiner, MobilEdit!, Lantern, and XRY.

For a detailed reference list, refer to `Appendix B`, *Tools for iOS Forensics*.

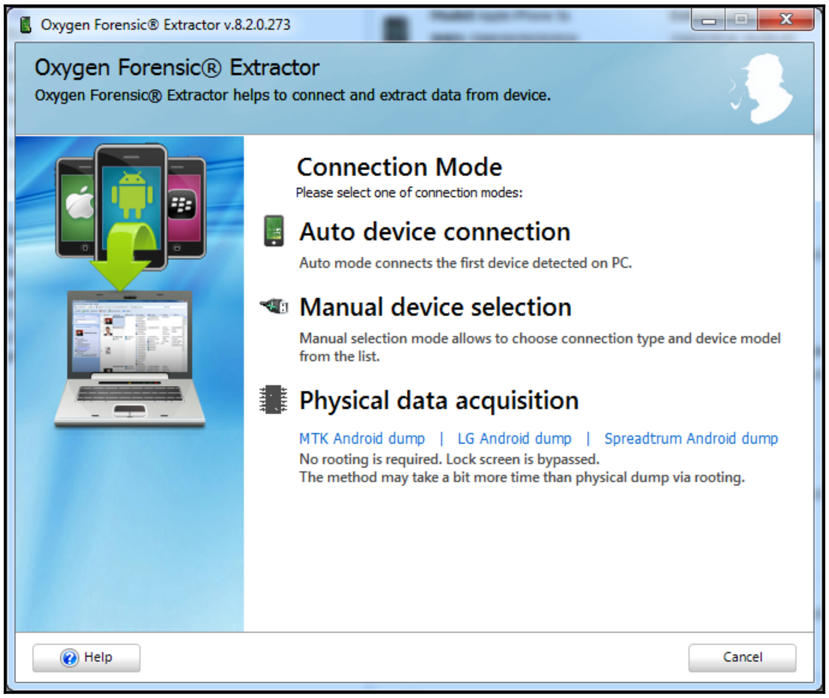
Case study – iTunes backup acquisition with Oxygen Forensic Analyst

Oxygen Forensic software is a commercial product that allows forensic analysts to perform iTunes backup acquisition of an iOS device. It is available in three licensing modes, **Starter**, **Analyst**, and **Detective**.

To start the extraction, it is necessary to click on the **Connect device** button on the main screen, as shown in the following screenshot:


Oxygen Forensic® Analyst Educational
File View Tools Service Help
All devices
Connect device
Import file
Devices and Cases
All devices

The software will then begin the extraction procedure, and you can choose the type of connection you want to start. You can choose between **Auto device connection** and **Manual device selection**, as shown in the following screenshot. For iDevices, it is generally sufficient to select the first option.



Oxygen Forensic® Extractor

Oxygen Forensic® Extractor helps to connect and extract data from device.

Connection Mode

Please select one of connection modes:

Auto device connection

Auto mode connects the first device detected on PC.

Manual device selection

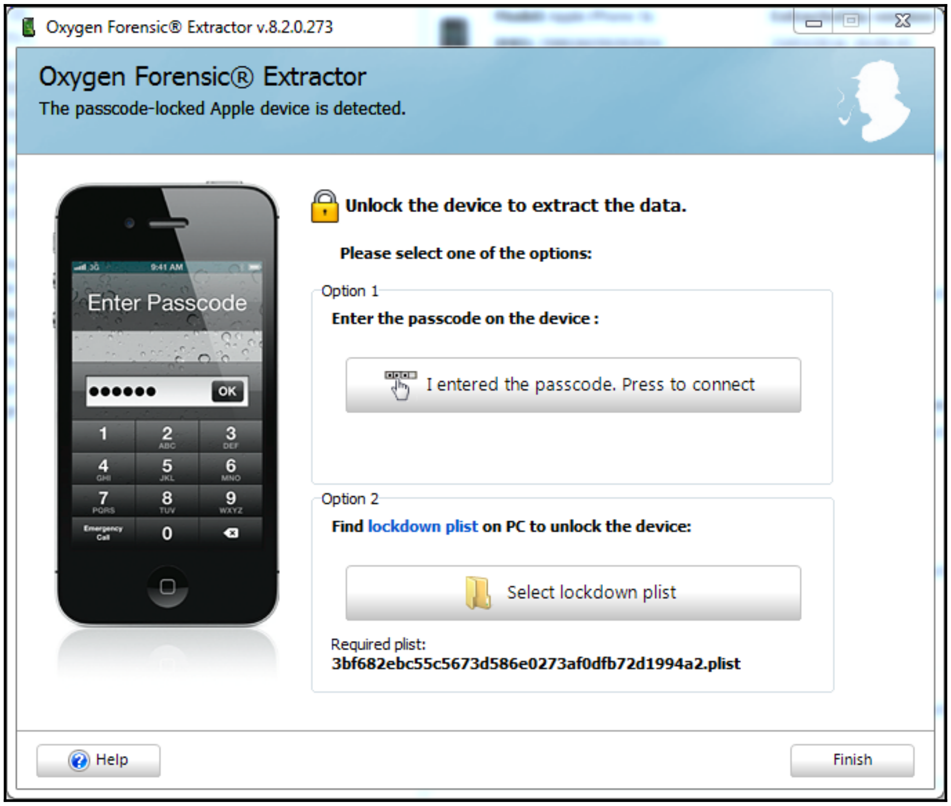
Manual selection mode allows to choose connection type and device model from the list.

Physical data acquisition

MTK Android dump | LG Android dump | Spreadtrum Android dump

No rooting is required. Lock screen is bypassed.
The method may take a bit more time than physical dump via rooting.

The software starts searching for a connected iDevice. If the device is locked with a passcode, the software asks the analyst to insert the passcode or to provide a lockdown certificate. The software provides the UDID for the iDevice, so it is easier for the analyst to search it on a computer previously synced with the device itself. If the analyst knows the passcode, he/she needs to insert it into the device, authorize the computer, and click on **I entered the passcode. Press to connect**. Otherwise, he/she can choose the **Select lockdown plist** option and provide the tool with a valid lockdown certificate.


Oxygen Forensic® Extractor v.8.2.0.273

Oxygen Forensic® Extractor

The passcode-locked Apple device is detected.

Unlock the device to extract the data.

Please select one of the options:

Option 1

Enter the passcode on the device :

I entered the passcode. Press to connect

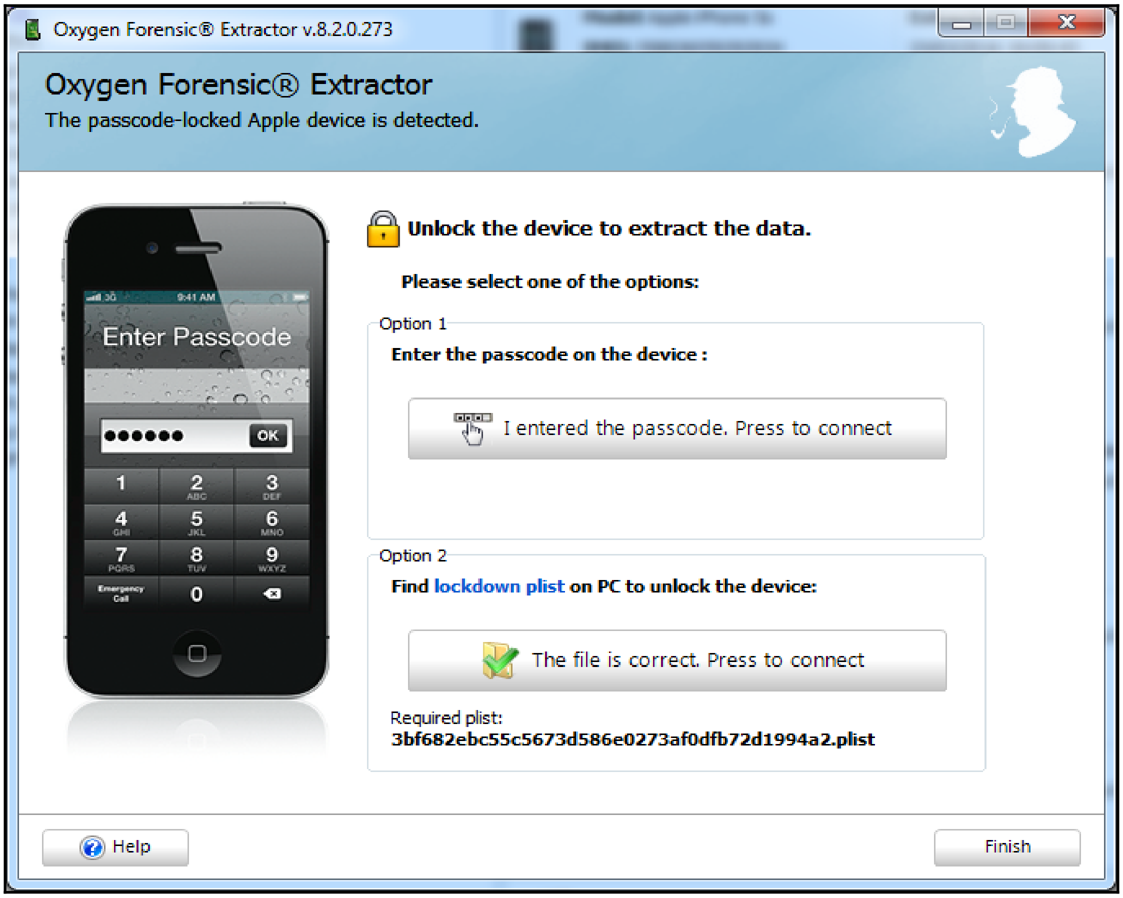
Option 2

Find lockdown plist on PC to unlock the device:

Select lockdown plist

Required plist:
3bf682ebc55c5673d586e0273af0dfb72d1994a2.plist

If the certificate is correct, the software displays a confirmation screen with a button to start the connection to the device, as shown in the following screenshot:



Oxygen Forensic® Extractor v.8.2.0.273

Oxygen Forensic® Extractor

The passcode-locked Apple device is detected.

Unlock the device to extract the data.

Please select one of the options:

Option 1

Enter the passcode on the device :

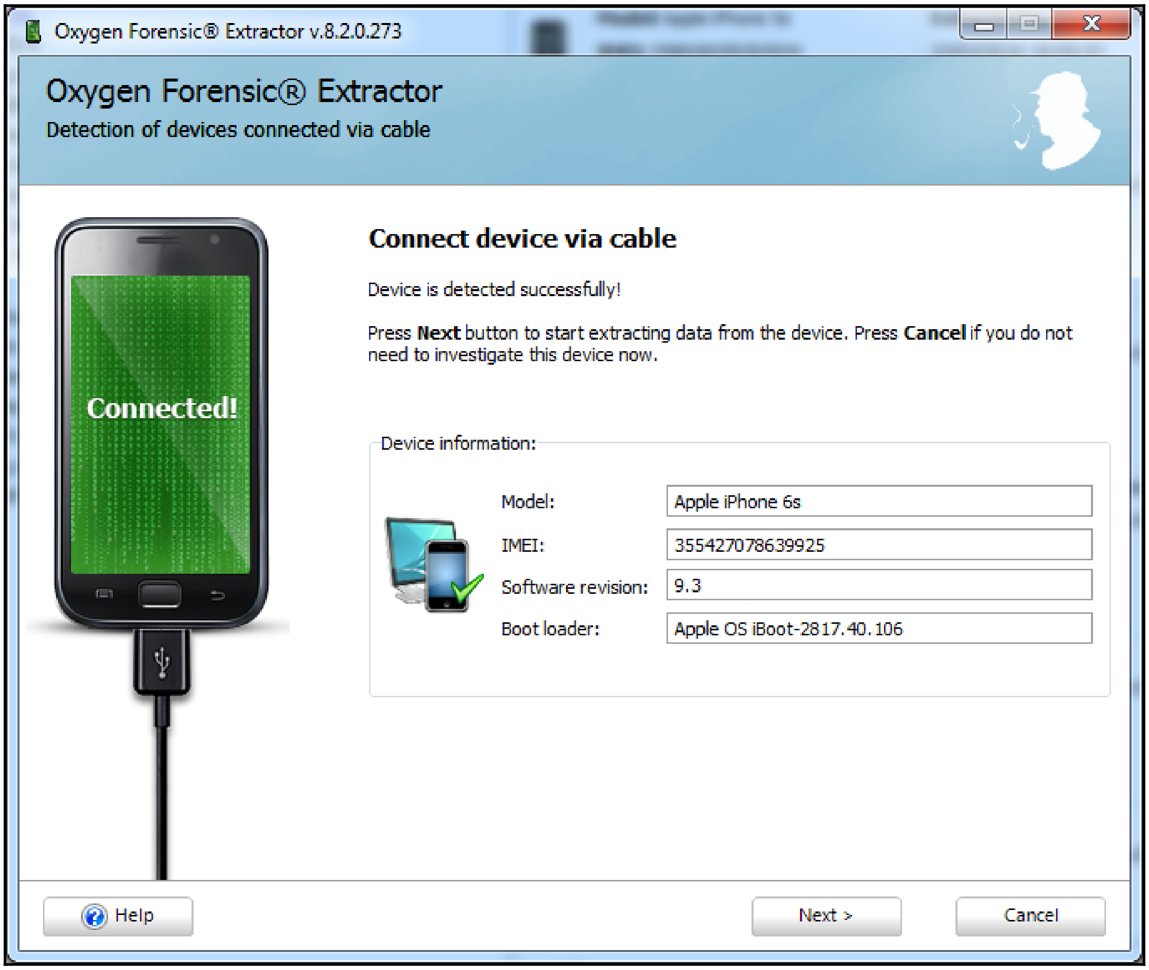
Option 2

Find lockdown plist on PC to unlock the device:

Required plist:
3bf682ebc55c5673d586e0273af0dfb72d1994a2.plist

Enter Passcode
OK
I entered the passcode. Press to connect
The file is correct. Press to connect
Help
Finish

At this point, the software displays information specific to the connected device (model, IMEI number, iOS version, and boot loader), as shown in the following screenshot:



Oxygen Forensic® Extractor v.8.2.0.273

Oxygen Forensic® Extractor

Detection of devices connected via cable

Connect device via cable

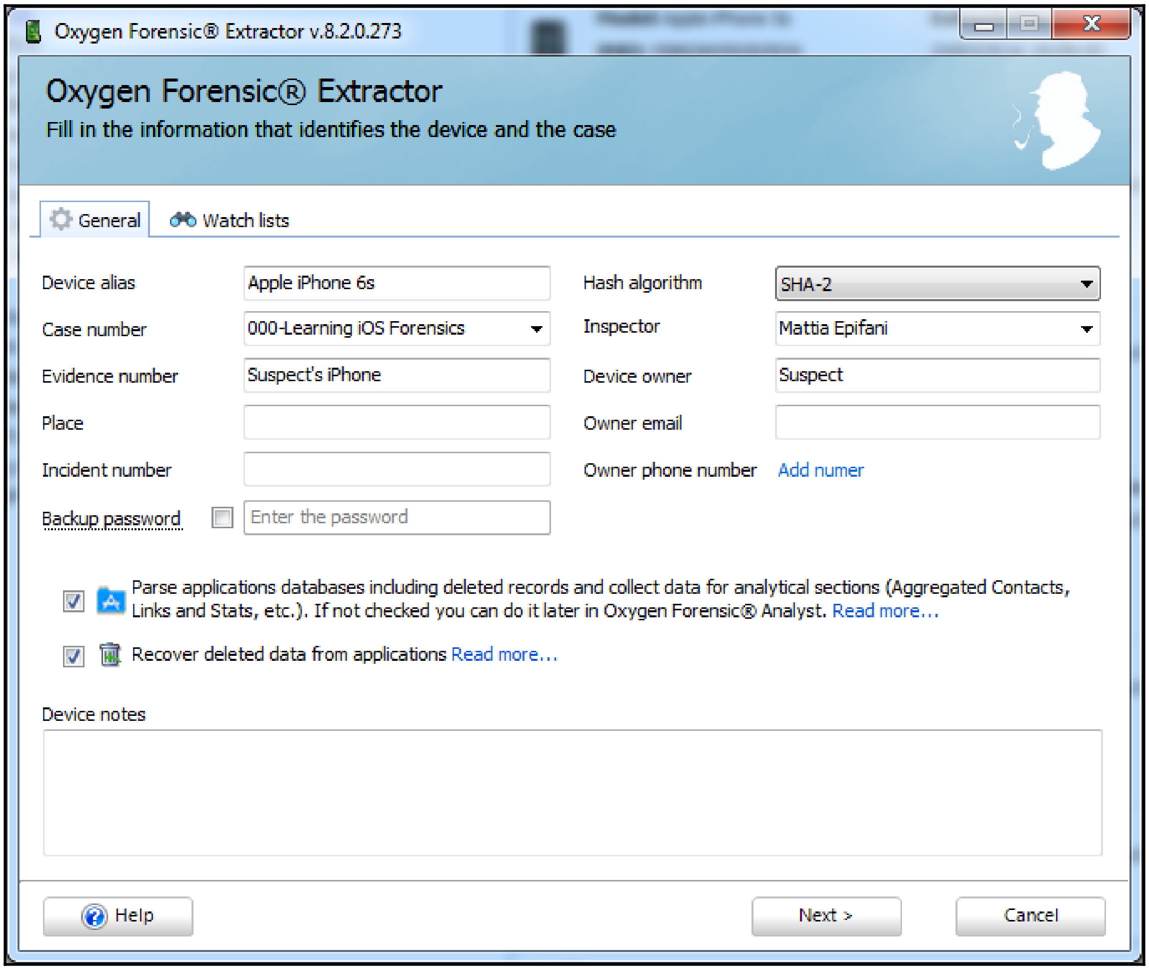
Device is detected successfully!

Press **Next** button to start extracting data from the device. Press **Cancel** if you do not need to investigate this device now.

Device information:

Model:	Apple iPhone 6s
IMEI:	355427078639925
Software revision:	9.3
Boot loader:	Apple OS iBoot-2817.40.106

The investigator can then enter information about the case, and if known, the backup password for the device.



Oxygen Forensic® Extractor v.8.2.0.273

Oxygen Forensic® Extractor

Fill in the information that identifies the device and the case

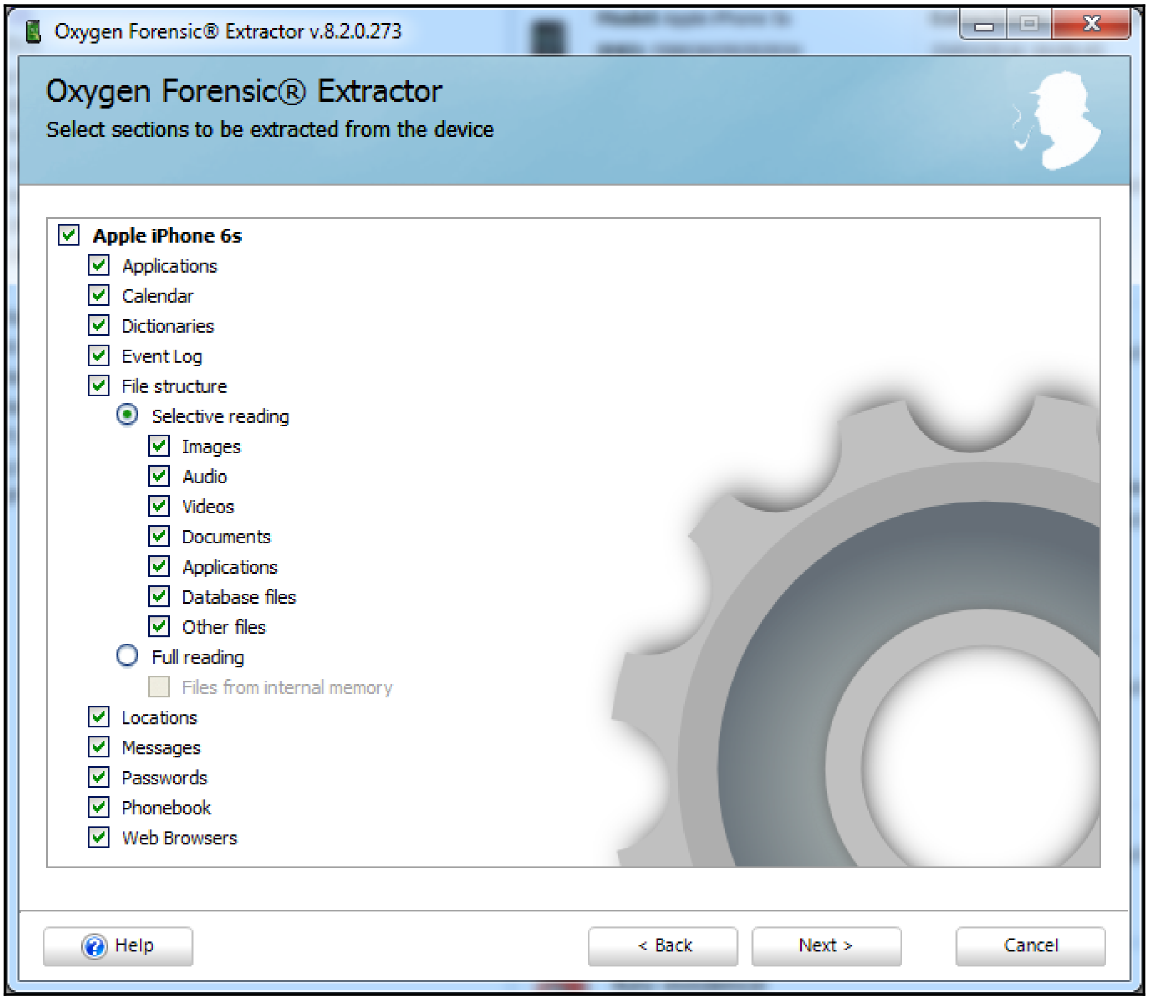
Device alias	Apple iPhone 6s	Hash algorithm	SHA-2
Case number	000-Learning iOS Forensics	Inspector	Mattia Epifani
Evidence number	Suspect's iPhone	Device owner	Suspect
Place		Owner email	
Incident number		Owner phone number	Add numer
Backup password	☐ Enter the password		

☑ Parse applications databases including deleted records and collect data for analytical sections (Aggregated Contacts, Links and Stats, etc.). If not checked you can do it later in Oxygen Forensic® Analyst. Read more...

☑ Recover deleted data from applications Read more...

Device notes

Then, the analyst can select the data they want to extract by choosing the ones supported by this method, as shown in the following screenshot:

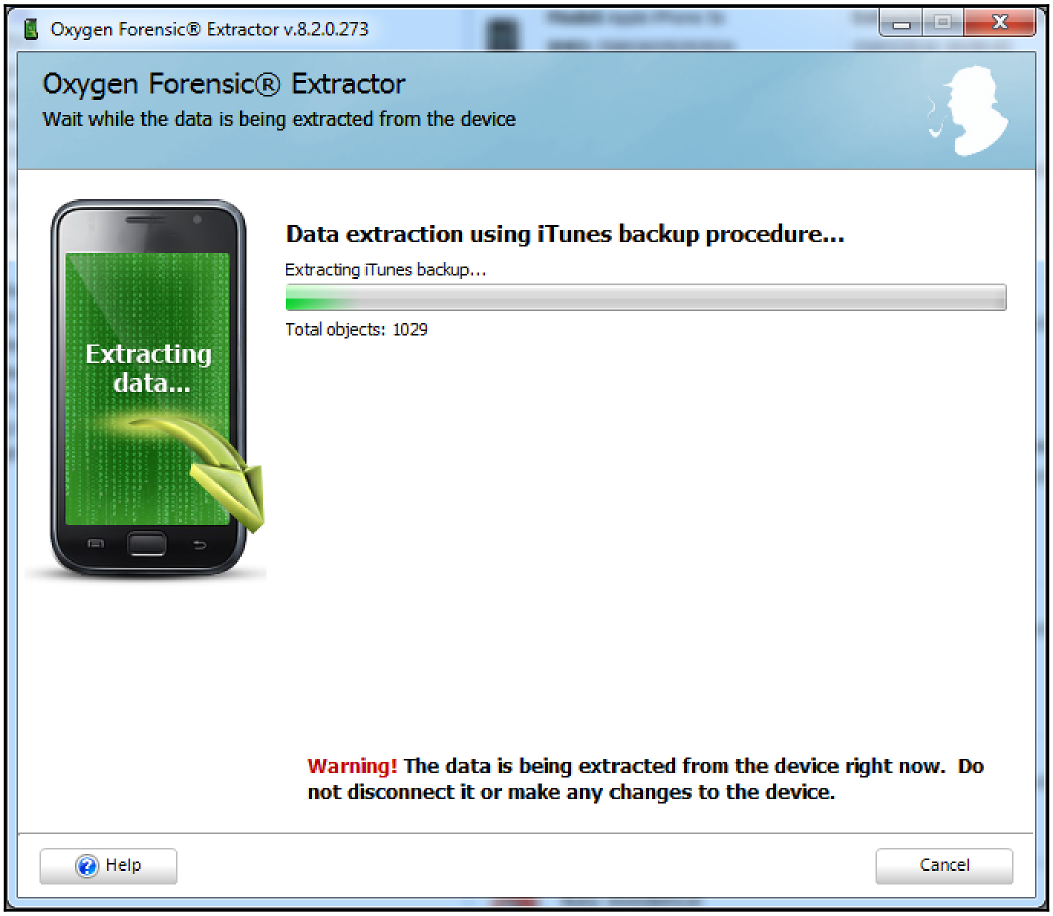


Oxygen Forensic® Extractor

Select sections to be extracted from the device

- ✓ **Apple iPhone 6s**
  - ✓ Applications
  - ✓ Calendar
  - ✓ Dictionaries
  - ✓ Event Log
  - ✓ File structure
    - ● Selective reading
      - ✓ Images
      - ✓ Audio
      - ✓ Videos
      - ✓ Documents
      - ✓ Applications
      - ✓ Database files
      - ✓ Other files
    - ○ Full reading
      - ☐ Files from internal memory
  - ✓ Locations
  - ✓ Messages
  - ✓ Passwords
  - ✓ Phonebook
  - ✓ Web Browsers

When clicking on the **Next** button, the acquisition procedure starts and displays a progress bar. It should be noted that during the extraction, the software also proceeds with parsing all the data found, including the search for deleted records within the database stored in the phone (for example, calls, SMS, chat, and so on). For this reason, the acquisition may require a large amount of time, but after that the analyst is ready to parse the data within the software, as shown in the following screenshot:



Oxygen Forensic® Extractor

Wait while the data is being extracted from the device

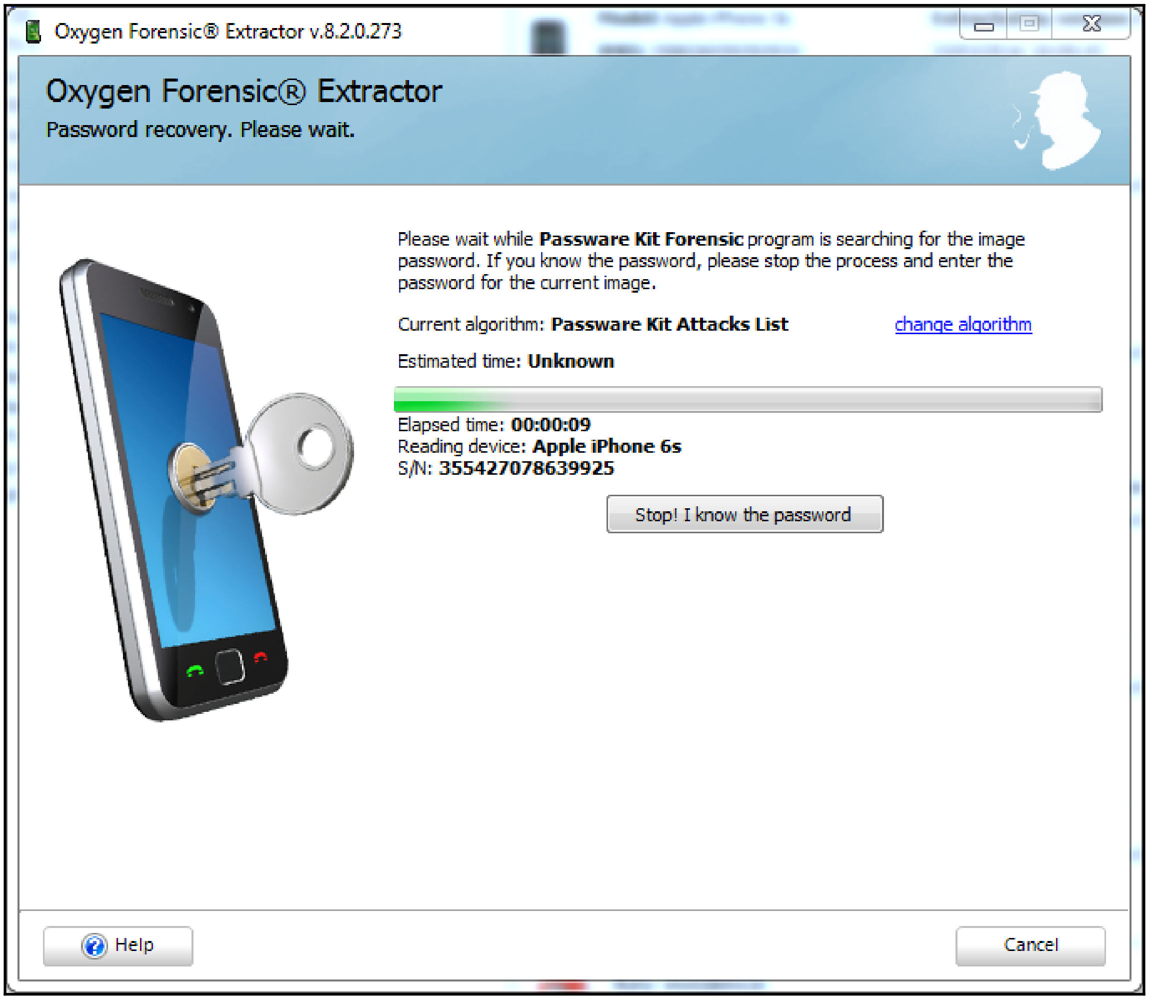
Data extraction using iTunes backup procedure...

Extracting iTunes backup...

Total objects: 1029

Warning! The data is being extracted from the device right now. Do not disconnect it or make any changes to the device.

If the device has a backup password previously set by the user, Oxygen can work with **Passware Kit Forensic** (if installed on the computer used to acquire data) trying to make an attack on the backup password. If the examiner knows the password, he/she has the chance to finish the attack and enter it manually. At the end of the cracking process, if the password has been detected, the software proceeds with the extraction of all the data, in a similar way as described previously. If the password is not found, the software extracts only the multimedia content (images, video, books, and so on) and does not provide information about the applications preinstalled or installed by the user:



Oxygen Forensic® Extractor

Password recovery. Please wait.

Please wait while **Passware Kit Forensic** program is searching for the image password. If you know the password, please stop the process and enter the password for the current image.

Current algorithm: **Passware Kit Attacks List**

Estimated time: **Unknown**

Elapsed time: **00:00:09**
Reading device: **Apple iPhone 6s**
S/N: **355427078639925**

Advanced logical acquisition

The advanced logical acquisition method was first introduced by the iOS security researcher Jonathan Zdziarski in his tool, **Waterboard**, released in June 2013. The author's description in his article states:

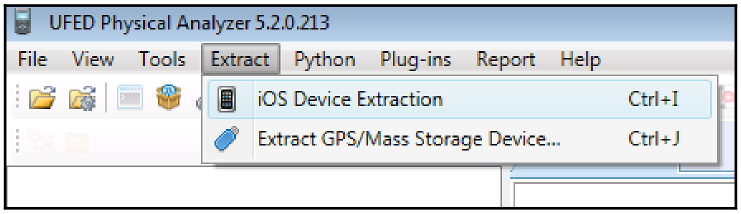
> *"Waterboard is an open source iOS forensic imaging tool, capable of performing an advanced logical acquisition of iOS devices by utilizing extended services and back doors in Apple's built-in lockdown services. These services can bypass Apple's mobile backup encryption and other encryption to deliver a clear text copy of much of the filesystem to any machine that can or has previously paired with the device."*

A detailed explanation can be found in the paper *Identifying Back Doors, Attack Points, and Surveillance Mechanisms in iOS Devices*, by *Jonathan Zdziarski* (refer to `Appendix A`, *References*). Currently, the Waterboard tool is no longer available and supported by Zdziarski, but there are few forensic tools offering the same feature, UFED Physical Analyzer, Oxygen Forensics Toolkit, and AccessData MPE.

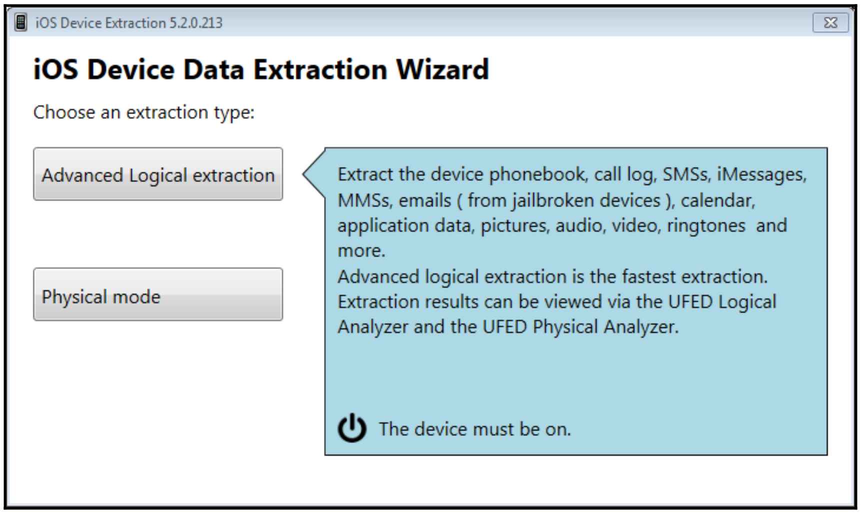
This method has been made unavailable starting from iOS 8, also due to the discovery of Zdziarski. On devices not yet updated, which are running iOS 7, the method is still usable; but even in this case it is necessary that the device is unlocked or that a valid lockdown certificate is available.

Case study – advanced logical acquisition with UFED Physical Analyzer

UFED Physical Analyzer is a software product from Cellebrite UFED and supplied with the purchase of UFED Touch or UFED 4PC. The advanced logical acquisition in UFED Physical Analyzer can be started through the main interface of the software under the menu item, **iOS Device Extraction** under **Extract**, as shown in the following screenshot:


UFED Physical Analyzer 5.2.0.213
File View Tools Extract Python Plug-ins Report Help
iOS Device Extraction Ctrl+I
Extract GPS/Mass Storage Device... Ctrl+J

The analyst can now choose **Advanced Logical extraction**.



iOS Device Extraction 5.2.0.213

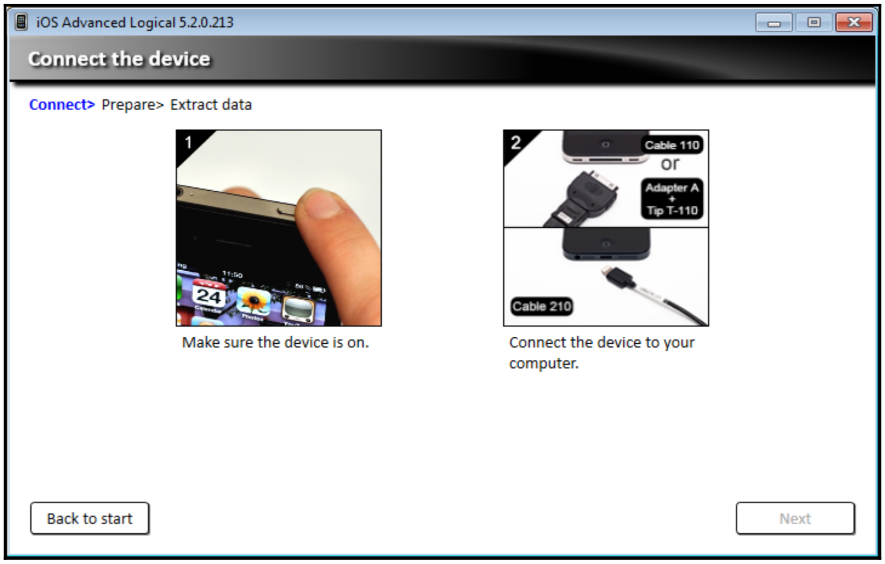
iOS Device Data Extraction Wizard

Choose an extraction type:

Extract the device phonebook, call log, SMSs, iMessages, MMSs, emails (from jailbroken devices), calendar, application data, pictures, audio, video, ringtones and more.
Advanced logical extraction is the fastest extraction.
Extraction results can be viewed via the UFED Logical Analyzer and the UFED Physical Analyzer.

The device must be on.

The software requires you to connect the turned-on device using the correct cable (30-pin connector or Lightning 8-pin connector), as shown as follows:



iOS Advanced Logical 5.2.0.213

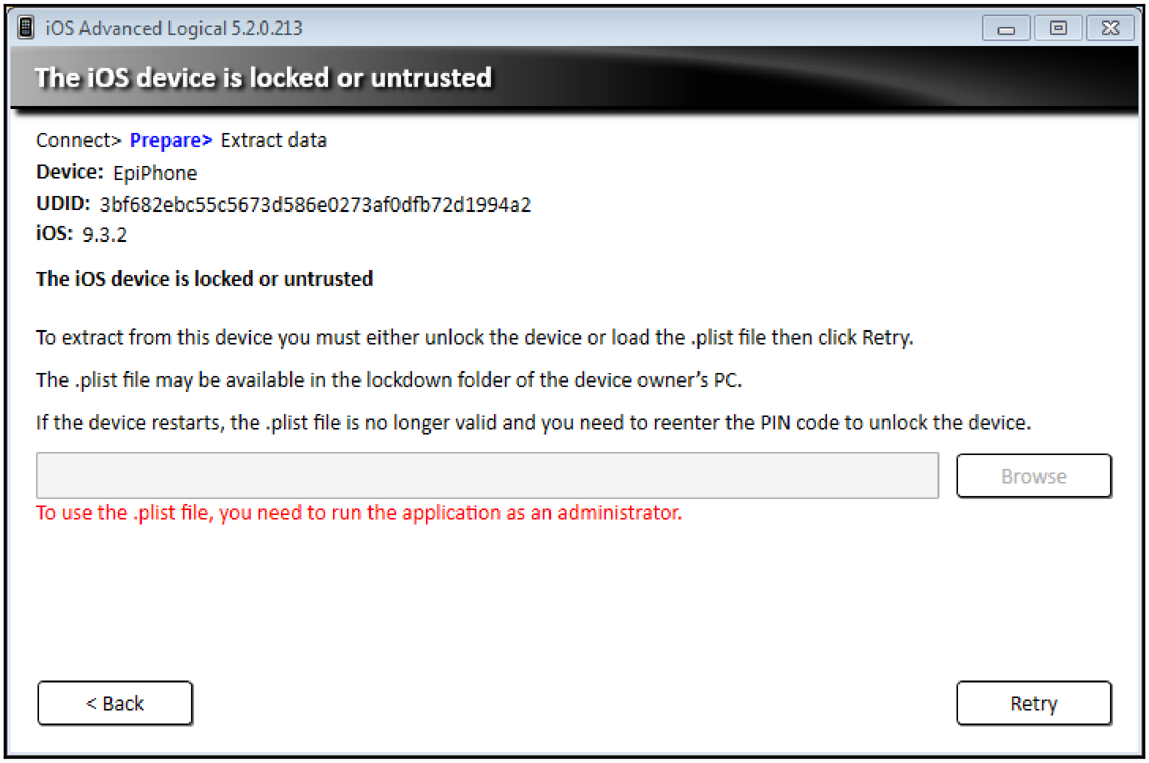
Connect the device

Connect> Prepare> Extract data

Make sure the device is on.

Connect the device to your computer.

The device must be powered on and unlocked; otherwise, the software displays a message stating **The iOS device is locked or untrusted**. To proceed, the analyst must unlock the device with the passcode or provide the right lockdown certificate.



iOS Advanced Logical 5.2.0.213

The iOS device is locked or untrusted

Connect> Prepare> Extract data

Device: EpiPhone

UDID: 3bf682ebc55c5673d586e0273af0dfb72d1994a2

iOS: 9.3.2

The iOS device is locked or untrusted

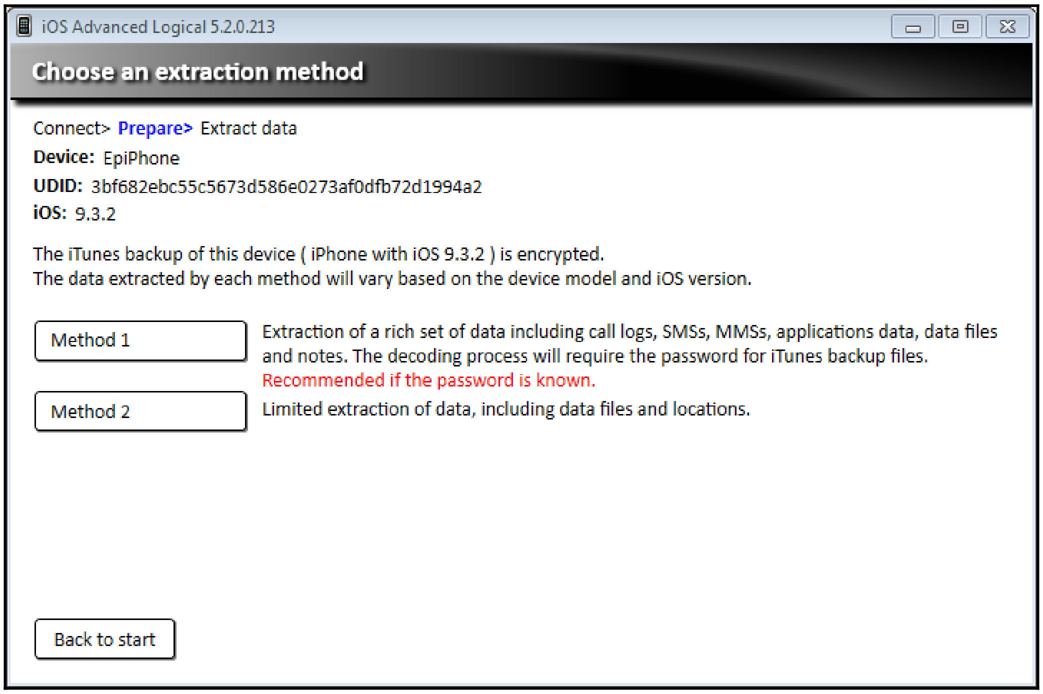
To extract from this device you must either unlock the device or load the .plist file then click Retry.

The .plist file may be available in the lockdown folder of the device owner's PC.

If the device restarts, the .plist file is no longer valid and you need to reenter the PIN code to unlock the device.

To use the .plist file, you need to run the application as an administrator.

The software checks whether a password is set on the backup device and shows two possible methods for the acquisition: **Method 1** corresponds to a device backup, while **Method 2** allows the analyst to extract data using both the AFC protocol and the lockdown service (advanced logical acquisition). If the device is jailbroken then **Method 3** will show up to allow a complete filesystem dump.


iOS Advanced Logical 5.2.0.213
Choose an extraction method
Connect> Prepare> Extract data
Method 1
Method 2
Back to start

Device: EpiPhone
UDID: 3bf682ebc55c5673d586e0273af0dfb72d1994a2
iOS: 9.3.2

The iTunes backup of this device (iPhone with iOS 9.3.2) is encrypted.
The data extracted by each method will vary based on the device model and iOS version.

Extraction of a rich set of data including call logs, SMSs, MMSs, applications data, data files and notes. The decoding process will require the password for iTunes backup files.
Recommended if the password is known.

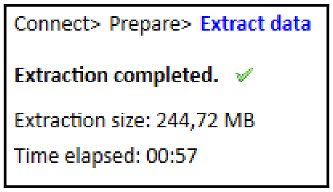
Limited extraction of data, including data files and locations.

If the device has a backup password with **Method 1**, the analyst must know the password or crack it (as explained in `Chapter 4`, *Evidence Acquisition and Analysis from iTunes Backup*), while with **Method 2**, it is possible to extract part of the data even without cracking the backup password. For this reason, when you need to acquire a device with a backup password, it is advisable to perform both acquisitions. In this way, you can definitely see some information, thanks to **Method 2**, and try to recover more details by cracking the encrypted acquisition carried out with **Method 1**.

If the device does not have a backup password then Physical Analyzer will propose the option to encrypt the backup with a known password. This is because certain files, such as the keychain, are in this case encrypted with the chosen backup password and so it is possible to recover more information (that is, user stored passwords).

Once you select the extraction method, the software initiates the procedure requiring the user to set the destination folder. Scanning takes a variable time depending on the chosen method (**Method 1** is performed in a single step and is faster than **Method 2**, which requires three steps), the memory size of the device, and the space occupied by files (especially media files such as pictures, videos, music, and so on).

Once the acquisition is complete, the software displays a report showing the amount of extracted data and the time taken, as shown in the following screenshot. From this window, the analyst can choose whether to return to the home screen or open the acquisition made in UFED Physical Analyzer for the analysis activities.


Connect> Prepare> Extract data

Extraction completed. ✓

Extraction size: 244,72 MB

Time elapsed: 00:57

Physical acquisition with forensic tools

Physical acquisition allows most of the content from an iOS device to be extracted. Unlike the AFC, backup, and advanced logical methods, the analyst can obtain a forensic copy of the device's internal NAND storage and access all the files stored there. Some examples of interesting information that can only be retrieved through physical acquisition are the e-mail messages and log files of the device. The physical acquisition is based on hardware vulnerabilities during the boot process. For this reason, this operation is not invasive on the data stored on the iDevice because it uploads an alternative operating system directly into RAM through which it can launch acquisition commands. Currently, it is supported on the first iPhone model 3G/3GS/4, iPad 1, and iPod touch 1/2/3/4.

If the device is not protected by a passcode, the physical acquisition can be carried out without problems by creating an image of both system and data partitions.

If the iDevice is protected with a passcode, it is necessary to highlight the following two cases:

- If the passcode is simple (4 digits), it can be cracked in 20 to 30 minutes, depending on the device type
- If the passcode is complex (multidigit or alphanumeric) the analyst has the following two options:

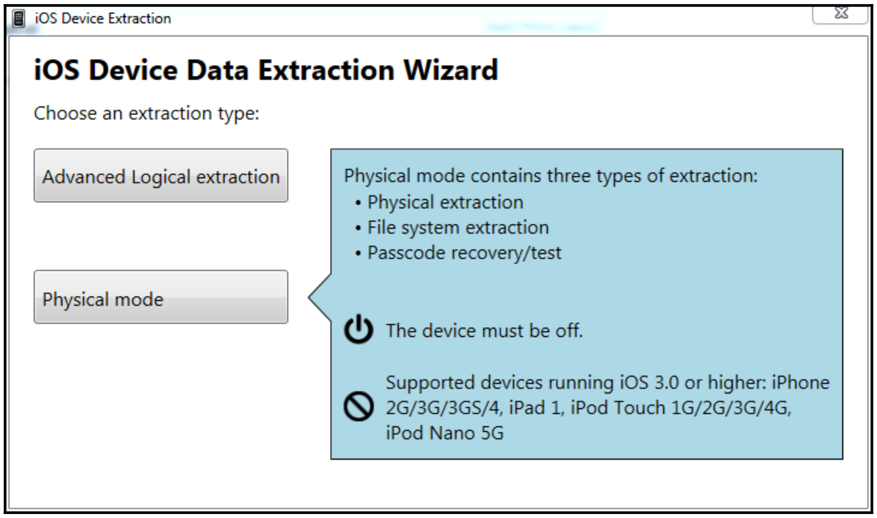
- Try a brute force or a dictionary attack on the passcode.
- Perform the physical acquisition without cracking the passcode. In this case, the physical acquisition will decode all the data whose encryption does not depend on the passcode, while other data (for example, e-mail, stored passwords, and so on) cannot be decrypted.

Several forensic tools can perform physical acquisition, such as iPhone data protection tools, UFED Physical Analyzer, Elcomsoft iOS Forensic Toolkit, Lantern, AccessData MPE+, iXAM, and XRY.

For a more comprehensive and detailed list of tools, books, and papers related to physical acquisition, refer to `Appendix A`, *References*, and `Appendix B`, *Tools for iOS Forensics*.

Case study – physical acquisition with UFED Physical Analyzer

Physical acquisition using UFED Physical Analyzer can be started through the main interface of the software from the menu item **iOS Device Extraction**, under **Extract**, as follows:


iOS Device Extraction

iOS Device Data Extraction Wizard

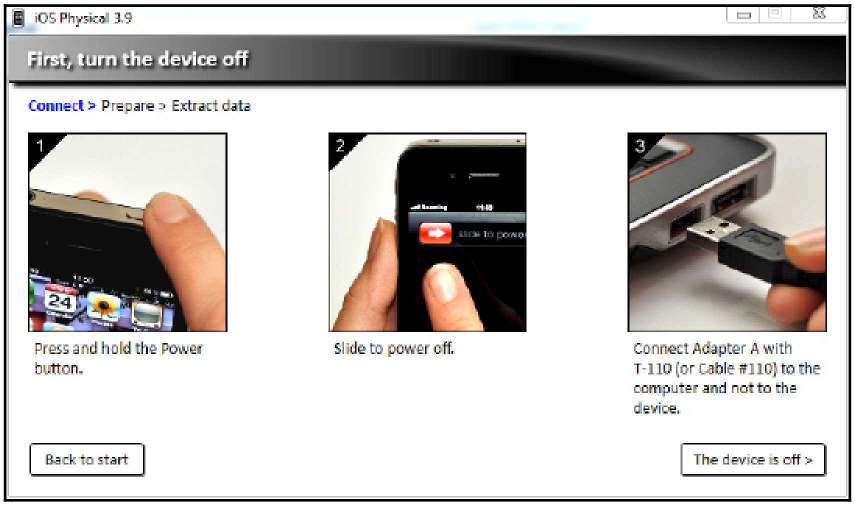
Choose an extraction type:

Physical mode contains three types of extraction:
- Physical extraction
- File system extraction
- Passcode recovery/test

The device must be off.

Supported devices running iOS 3.0 or higher: iPhone 2G/3G/3GS/4, iPad 1, iPod Touch 1G/2G/3G/4G, iPod Nano 5G

The device must be powered off and then the analyst can connect the correct cable to the computer (and not yet to the iDevice).

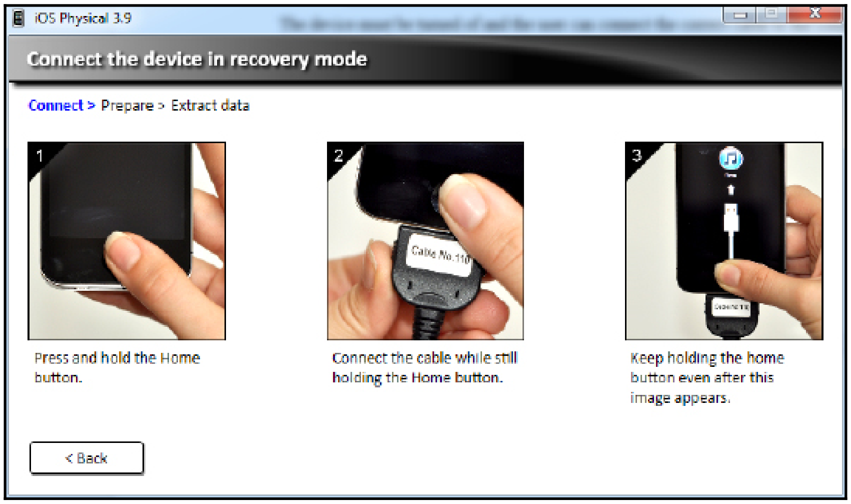


Press and hold the Power button.

Slide to power off.

Connect Adapter A with T-110 (or Cable #110) to the computer and not to the device.

Now, the investigator must connect the device in **Recovery** mode. It means that they need to press and hold the Home button and connect the device, as shown in the following screenshot:

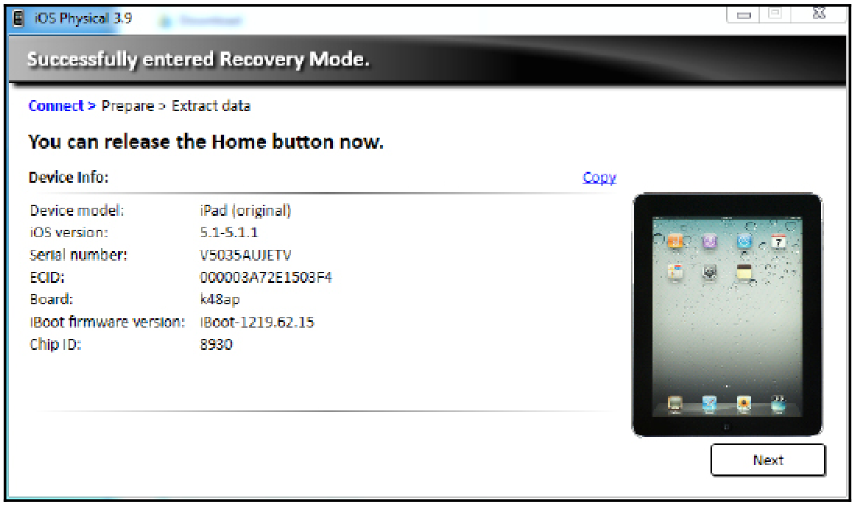

iOS Physical 3.9
Connect the device in recovery mode
Connect > Prepare > Extract data
1
2
3
< Back

Press and hold the Home button.

Connect the cable while still holding the Home button.

Keep holding the home button even after this image appears.

The software displays the information related to the device (the **iOS version**, **serial number**, **Board**, **IBoot firmware version**, **Chip ID**, and so on), as shown in the following screenshot:


iOS Physical 3.9

Successfully entered Recovery Mode.

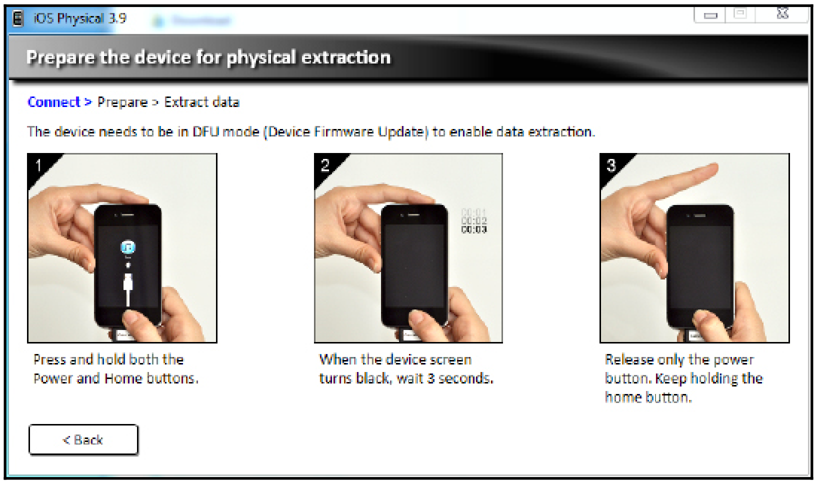
Connect > Prepare > Extract data

You can release the Home button now.

Device Info: Copy

Device model:	iPad (original)
iOS version:	5.1-5.1.1
Serial number:	V5035AUJETV
ECID:	000003A72E1503F4
Board:	k48ap
IBoot firmware version:	iBoot-1219.62.15
Chip ID:	8930

Now, the device must be set in DFU mode by pressing the **Power** and **Home** buttons together, and release the Power button 3 seconds after the device's screen becomes black.


iOS Physical 3.9

Prepare the device for physical extraction

Connect > Prepare > Extract data

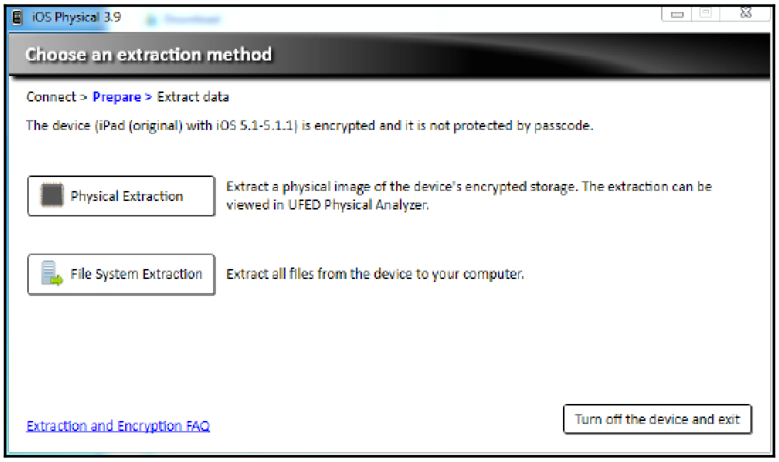
The device needs to be in DFU mode (Device Firmware Update) to enable data extraction.

1 Press and hold both the Power and Home buttons.

2 When the device screen turns black, wait 3 seconds.

3 Release only the power button. Keep holding the home button.

The software uploads the boot loader in memory and provides the analyst with two options: **Physical Extraction** and **File System Extraction**, as shown in the following screenshot. The first one extracts a physical image of the encrypted data partition, so the extraction can be viewed in UFED Physical Analyzer and in other analysis tools. UFED also provides information about the passcode protection. If the device is not protected by a passcode, it can start the acquisition immediately and decrypt all the files.


iOS Physical 3.9
Choose an extraction method
Physical Extraction
File System Extraction
Extraction and Encryption FAQ
Turn off the device and exit

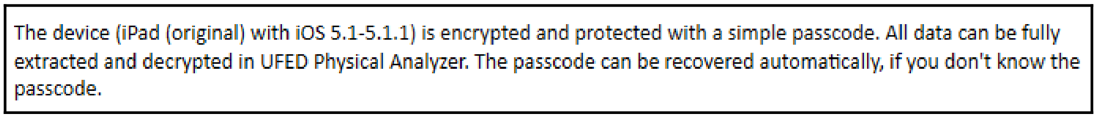
Connect > Prepare > Extract data

The device (iPad (original) with iOS 5.1-5.1.1) is encrypted and it is not protected by passcode.

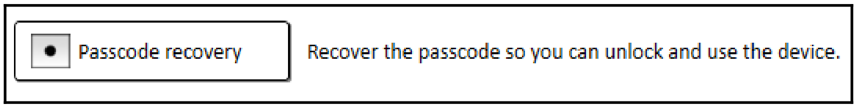
Physical Extraction — Extract a physical image of the device's encrypted storage. The extraction can be viewed in UFED Physical Analyzer.

File System Extraction — Extract all files from the device to your computer.

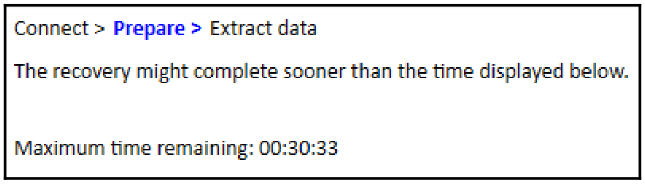
Otherwise, it depends on the passcode type. If the device has a simple passcode (four digits), it can be cracked in 20 to 30 minutes (depending on the iDevice type) by choosing the **Passcode recovery** option.



The device (iPad (original) with iOS 5.1-5.1.1) is encrypted and protected with a simple passcode. All data can be fully extracted and decrypted in UFED Physical Analyzer. The passcode can be recovered automatically, if you don't know the passcode.



Recover the passcode so you can unlock and use the device.

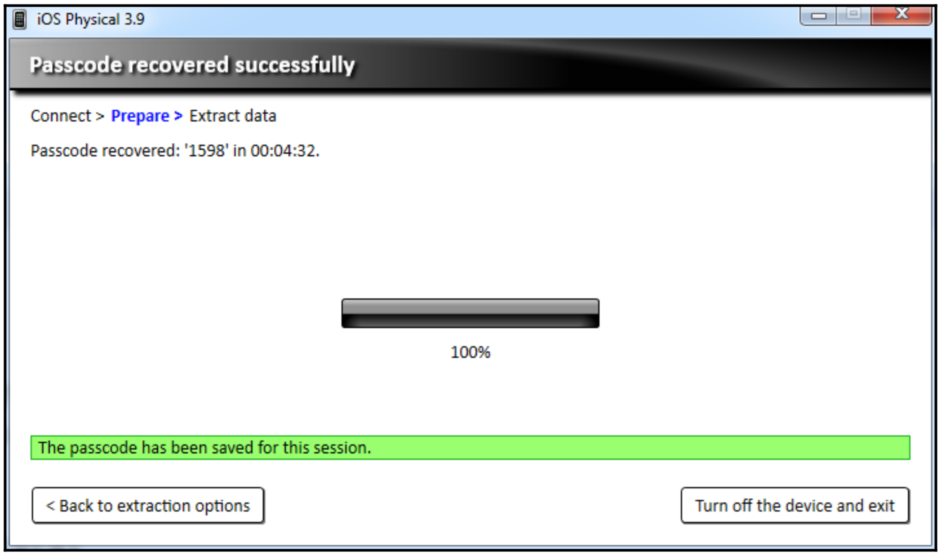


Connect > Prepare > Extract data

The recovery might complete sooner than the time displayed below.

Maximum time remaining: 00:30:33

At the end of the cracking stage, the software shows the passcode and gives the opportunity to start the acquisition, as follows:

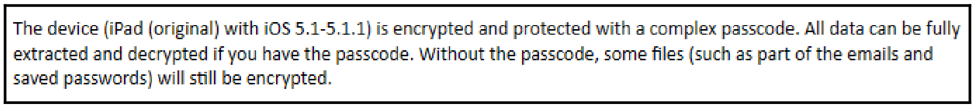

iOS Physical 3.9
Passcode recovered successfully
Connect > Prepare > Extract data

Passcode recovered: '1598' in 00:04:32.

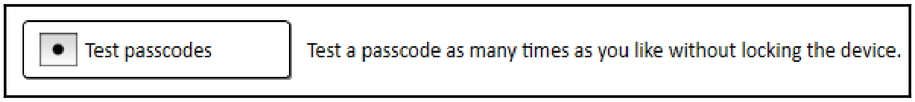
The passcode has been saved for this session.

< Back to extraction options
Turn off the device and exit

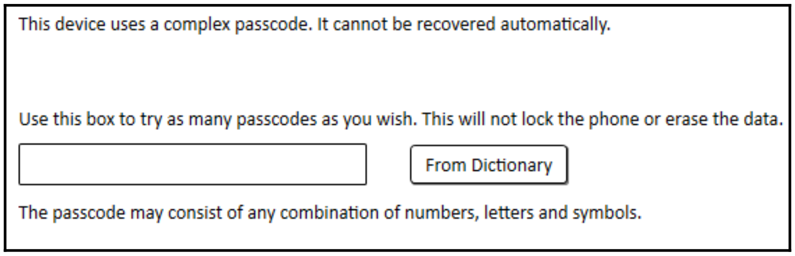
If the device has a complex passcode, the analyst has two options: either acquire a physical image without cracking the passcode (this means that some data will not be available, for example, e-mails, and stored passwords) or try to crack the passcode with a dictionary attack.



The device (iPad (original) with iOS 5.1-5.1.1) is encrypted and protected with a complex passcode. All data can be fully extracted and decrypted if you have the passcode. Without the passcode, some files (such as part of the emails and saved passwords) will still be encrypted.



Test a passcode as many times as you like without locking the device.



This device uses a complex passcode. It cannot be recovered automatically.

Use this box to try as many passcodes as you wish. This will not lock the phone or erase the data.

The passcode may consist of any combination of numbers, letters and symbols.

Dealing with a locked iDevice

The main problem, as already mentioned in the description of the various acquisition methods, is the presence of a lock code set by the user.

For devices up to iPhone 4, iPad 1, and iPod touch up to version 4, you can work around the presence of a lock code with the ability to load into RAM an operating system different from the original iOS: if the passcode is simple (four digits) the brute force time is a maximum of 30 minutes, increasing by several orders of magnitude (months or years) as the complexity of the passcode increases. In any case, for these types of device, the encryption is restricted to certain types of file (mainly e-mails and third-party application data), while for the system data the encryption does not depend on the passcode: for this reason, a physical acquisition allows access to such data.

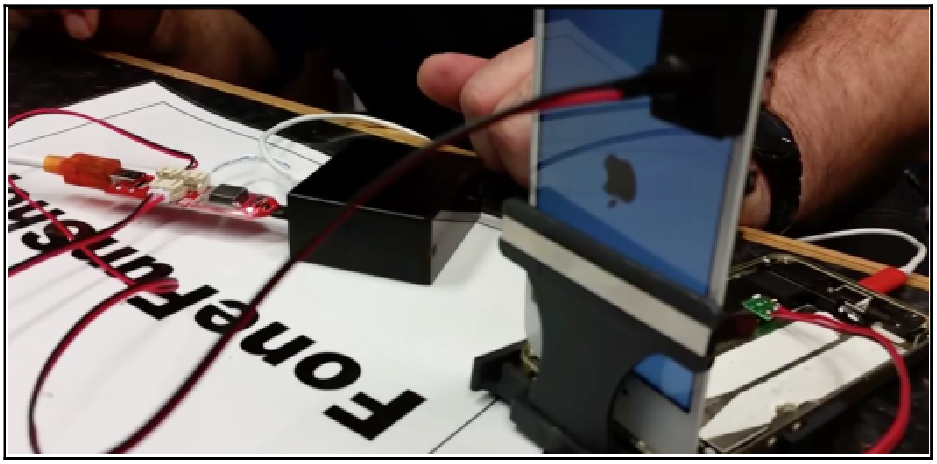
For the later devices, that is, all those also compatible with iOS 9, there are three available techniques to violate or bypass the passcode lock:

- Use a valid lockdown certificate
- Use a tool that allows submitting the passcode to the device via USB port instead of the touch screen
- Get support from companies that developed proprietary techniques able to brute force the passcode

The first technique is applicable to all versions of iOS, including iOS 9. In this case, it is necessary that two conditions are met: the device, even if locked, must have been found switched on and maintained in this condition, and the same must have been unlocked at least once in the past 48 hours. It is therefore clear that this method should be used immediately with respect to the seizure or discovery of the device, immediately seeking a computer that may hold a valid certificate and proceeding with an acquisition as quickly as possible. The use of a valid certificate allows for all devices, including those with iOS 9, the acquisition by AFC Protocol, and acquisition through backup. On devices with iOS 7 it also allows to make the advanced logical acquisition.

The second method can be used on all devices but only up to iOS 8.1. Several hardware devices have been made available on the market and are all based, essentially, on the principle of passing the passcode via USB and not through the touch screen. The best-known and most commonly used forensic software is the UFED User Lock Code Recovery Tool by Cellebrite, available in the licenses of the software UFED 4PC and UFED Touch. It runs on iOS 7. Other devices, which are not born as forensic tools but very much used in the community, are IP-BOX, MFC dongle, SecureView's svStrike, and HDB Box.

These devices, through an additional adapter, can also be used on operating systems up to iOS 8.1. It's important to note that none of these tools manage to overcome the limitation, which can be activated by the user, of secure wiping after 10 incorrect attempts at entering the code: this limitation stays in place also when the code is submitted to the phone through the USB port. For this reason, these techniques need to be applied if and only if it is certain that wiping is not active on the device.



The alternative solution is to get support from some specialized company that has developed proprietary techniques to recover the passcode that have not yet been implemented in software and hardware on the market. At the time of writing this edition of the book, the only company offering this service is Cellebrite through the **Cellebrite Advanced Investigative Services** (**CAIS**) center. As reported in the company's website (`http://www.cellebrite.com/Pages/cellebrite-unlocking-services-now-available-for-apple-ios-9-and-samsung-galaxy-s6-and-s7-devices`), the service is available for all devices with a 32-bit processor (that is, iPhone 4s/5/5c, iPad 2/3G/4G, iPad mini 1G, and iPod touch 5G) with any version of iOS 8 operating system (that is, 8 – 8.0/8.0.1/8.0.2/8.1/8.1.1/8.1.2/8.1.3/8.2/8.3/8.4/8.4.1) or iOS 9 operating system (that is, 9.0/9.0.1/9.0.2/9.1/9.2/9.2.1/9.3/9.3.1/9.3.2). The company ensures that there will be no physical invasiveness on the device (that is, there is no need of any hardware intervention) and the non-activation of the wiping function after 10 incorrect attempts.

From a Real Case
At the beginning of 2016, one of the authors of the book was appointed Expert Witness from the Court of Milan as part of the criminal proceedings against Alexander Boettcher, accused together with his girlfriend Martina Levato, of having made four acid attacks against her ex-boyfriends. At the time when the couple was arrested (December 2014), Boettcher got his iPhone 5 running iOS 8.0 seized, which was locked with a 4-digit passcode that he said he did not remember. At that time, there were no known techniques that allowed bypassing the code without activating Apple protection systems (for example, wiping encryption keys after 10 incorrect entries), and for this reason the prosecutor and the Judiciary Police have decided to analyze the other devices, finding a backup on the computer of the suspect made in September 2014. During the hearings in January 2016, it was deemed to be useful by the Judges evaluating new possibilities to access the data. They then appointed one of the authors asking technical advice to evaluate new possibilities in this regard. The consultant therefore decided to contact the CAIS service offered by the company Cellebrite and in February 2016 brought the phone to their EU headquarters in Munich, Germany, where the passcode was detected after two days of work, allowing in this way the acquisition of the device through the backup mode and AFC protocol. The news has been the subject of attention by the Italian national press for the notoriety of the case, and the international press, given the temporal proximity with having Apple vs. FBI (refer, for example, to *Apple-FBI Case: this is how the iPhone for the investigation of acid attacks in Milan was unlocked* – `http://www.lastampa.it/216/3/1/tecnologia/news/come-stato-sbloccato-liphone-nel-caso-delle-aggressioni-con-lacido-a-milano-3hmkkOwVQsiCFtkjjJbt5O/pagina.html`, and *It Might Cost The FBI Just $ 1.500 To Get Into Terrorist's iPhone*: `http://www.forbes.com/sites/thomasbrewster/216/3/23/cellebrite-apple-iphone-fbi-syed-farook-alexander-boettcher/`).

In addition to these techniques, other methods have been used in practice to bypass the lock code. The best known are the use of vulnerabilities on the specific version of the operating system that allow access to the device's screen (often associated with the application Siri), or forcing the generation of a backup to iCloud and bringing the device into the proximity of a known Wi-Fi network.

iOS device jailbreaking

iOS jailbreaking is the process of removing limitations on the iOS devices through the use of software and hardware exploits. It enables root access to the iOS filesystem and allows additional applications not available in the official Apple App Store to be downloaded. Various jailbreaking tools have been developed; an always updated list can be found at `http://theiphonewiki.com/wiki/Jailbreak`. Currently, the latest available tools are **Evasi0n** (`http://evasi0n.com/`) for iOS 7, **Taig** (`http://www.taig.com/en/`) for iOS 8, and **Pangu** (`http://en.pangu.io/`) for iOS 7, 8, and 9.

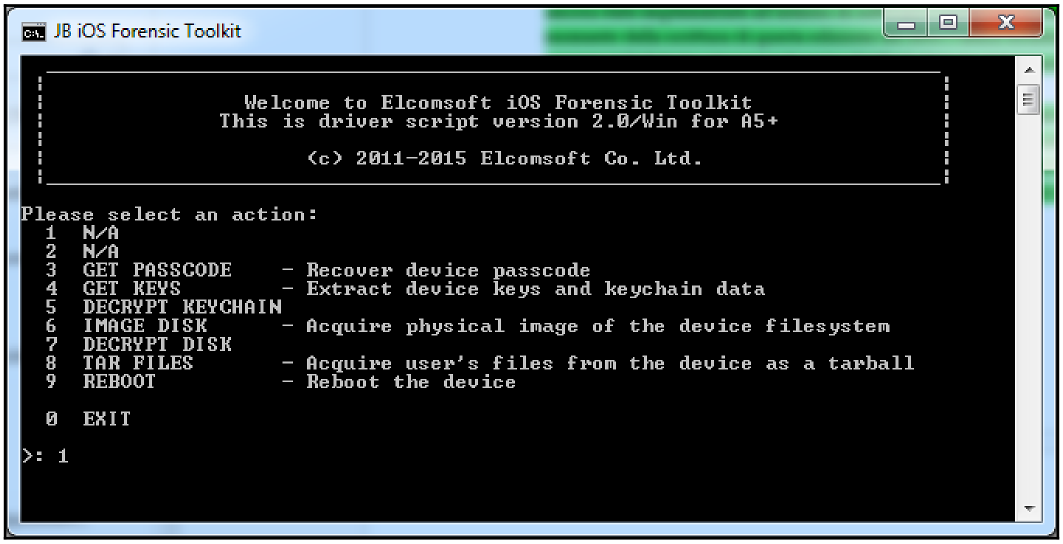
Jailbreaking is an invasive activity on the device system partition, so it cannot be considered as a forensic operation. However, it is useful to note that for devices after iPhone 4, it is the only way to make a physical acquisition. It is therefore necessary that the device is already jailbroken or that the investigator can jailbreak it. On newer devices, in order to jailbreak the device, the analyst needs to know the passcode, since it requires actions to be performed directly on the unlocked device.

Case study – physical acquisition with Elcomsoft iOS Forensic Toolkit

At the time of writing, jailbreaking is supported up to iOS 9.3.3 by Pangu jailbreaking, and the **Elcomsoft iOS Forensic Toolkit** (**EIFT**) software is the only commercial tool that permits a physical acquisition of a jailbroken device. It currently works on both 32- and 64-bit devices, but in the first case, it can extract a physical image, while in the second it can recover a full filesystem dump of the device. It is important to mention that although a physical image is different from a filesystem image, in the current state of the art there is effectively no difference between a filesystem and a physical image of these devices from a usable content point of view, because the unallocated space of a physical image is encrypted, and at the time of writing, there are still no known techniques to carve and decrypt from the unallocated space. As a result, you are effectively getting the content of a filesystem image with your physical image because you can't read the unallocated areas. Moreover, both Cellebrite and MPE+ support getting a full filesystem dump of jailbroken devices.

If the seized device is already jailbroken and the root password was not changed by the user (recall from `Chapter 1`, *Digital and Mobile Forensics*, that the default root password is *alpine*) it is also possible to try cracking the passcode.

This commercial tool can be used both on Windows or Mac. The following screenshots show the acquisition procedure performed with EIFT version 2.0 of a jailbroken iPad mini first generation with a passcode and iOS 9.0.2.

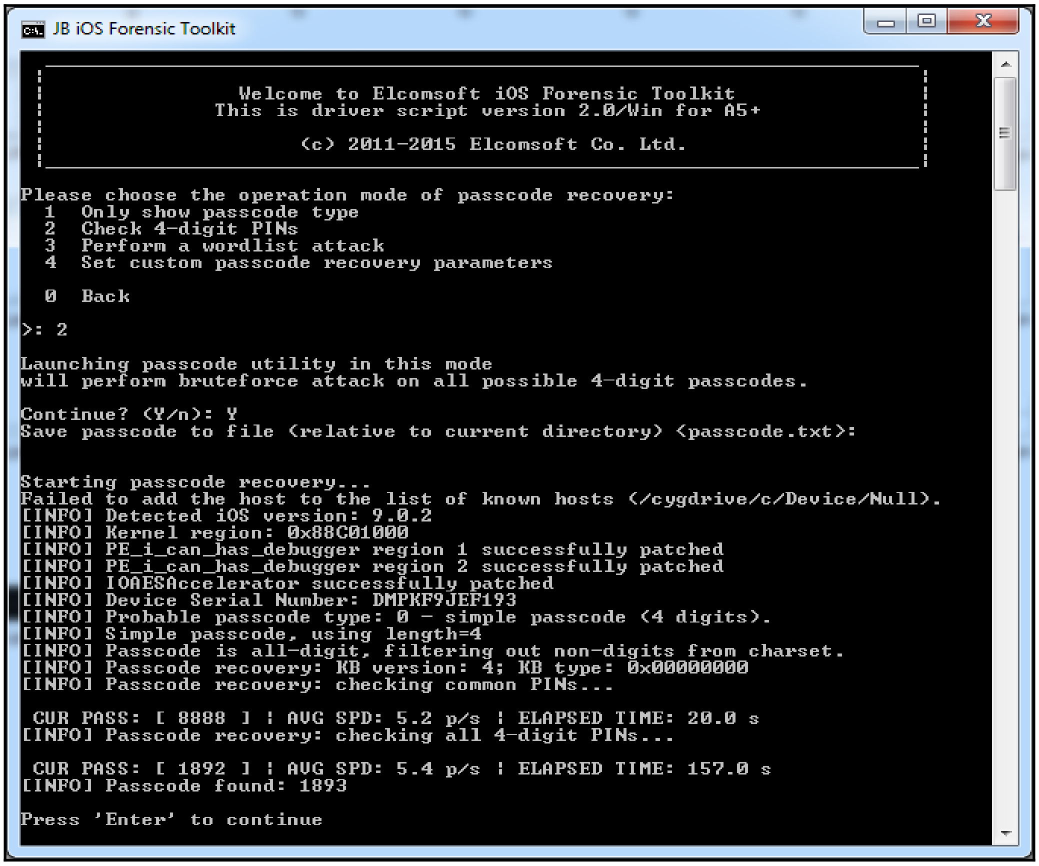


```
 ___________________________________________________________________________
|                                                                           |
|                  Welcome to Elcomsoft iOS Forensic Toolkit                |
|                This is driver script version 2.0/Win for A5+              |
|                                                                           |
|                       (c) 2011-2015 Elcomsoft Co. Ltd.                    |
|___________________________________________________________________________|

Please select an action:
  1  N/A
  2  N/A
  3  GET PASSCODE     - Recover device passcode
  4  GET KEYS         - Extract device keys and keychain data
  5  DECRYPT KEYCHAIN
  6  IMAGE DISK       - Acquire physical image of the device filesystem
  7  DECRYPT DISK
  8  TAR FILES        - Acquire user's files from the device as a tarball
  9  REBOOT           - Reboot the device

  0  EXIT

>: 1
```

The wizard is very simple and basically involves the following steps:

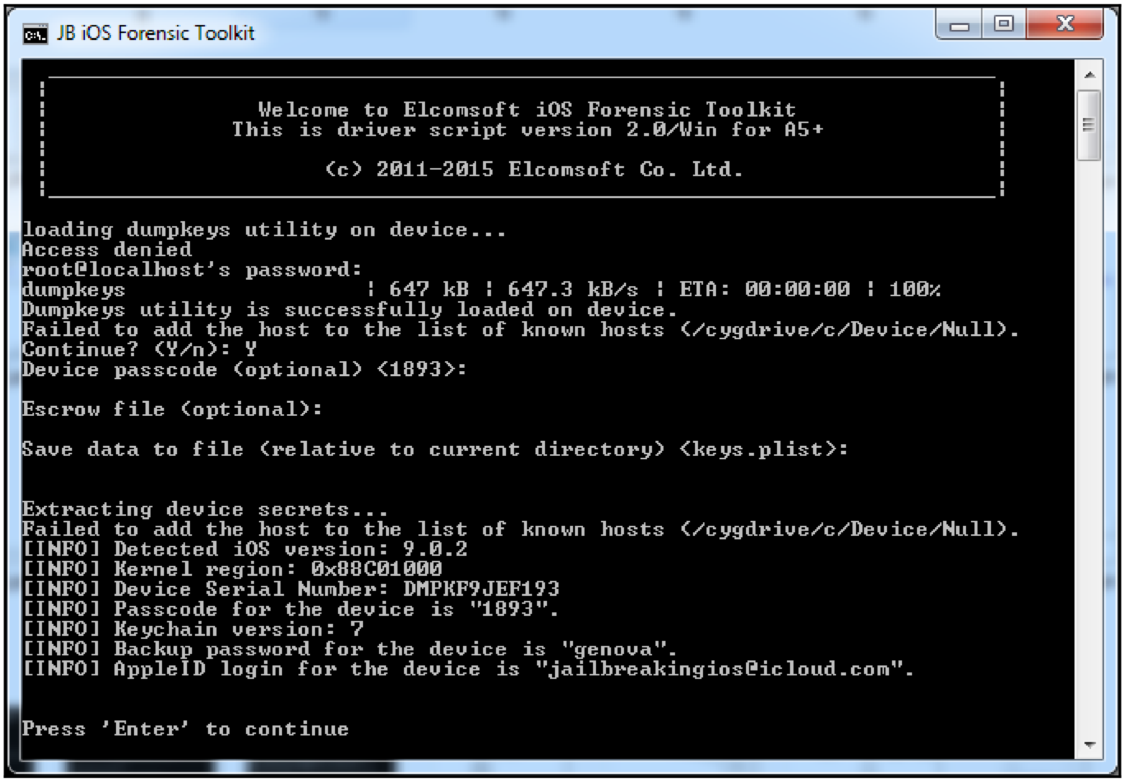
1. Get the passcode (option `3`)
2. Extraction of the encryption keys (option `4`)
3. Physical acquisition of the system partition (in plain text) and data partition (encrypted) (option `6`)
4. Decryption of the data partition with the extracted keys (option `7`)

The first step loads a tool on the device to crack the passcode and then, with a simple passcode in this case, starts the cracking activity. At the end, it prints the passcode on the screen and saves it in a `.txt` file.



```
 |------------------------------------------------------------------------|
 |                                                                        |
 |                 Welcome to Elcomsoft iOS Forensic Toolkit              |
 |               This is driver script version 2.0/Win for A5+            |
 |                                                                        |
 |                     (c) 2011-2015 Elcomsoft Co. Ltd.                   |
 |------------------------------------------------------------------------|

Please choose the operation mode of passcode recovery:
  1  Only show passcode type
  2  Check 4-digit PINs
  3  Perform a wordlist attack
  4  Set custom passcode recovery parameters

  0  Back

>: 2

Launching passcode utility in this mode
will perform bruteforce attack on all possible 4-digit passcodes.

Continue? (Y/n): Y
Save passcode to file (relative to current directory) <passcode.txt>:


Starting passcode recovery...
Failed to add the host to the list of known hosts (/cygdrive/c/Device/Null).
[INFO] Detected iOS version: 9.0.2
[INFO] Kernel region: 0x88C01000
[INFO] PE_i_can_has_debugger region 1 successfully patched
[INFO] PE_i_can_has_debugger region 2 successfully patched
[INFO] IOAESAccelerator successfully patched
[INFO] Device Serial Number: DMPKF9JEF193
[INFO] Probable passcode type: 0 - simple passcode (4 digits).
[INFO] Simple passcode, using length=4
[INFO] Passcode is all-digit, filtering out non-digits from charset.
[INFO] Passcode recovery: KB version: 4; KB type: 0x00000000
[INFO] Passcode recovery: checking common PINs...

 CUR PASS: [ 8888 ] | AVG SPD: 5.2 p/s | ELAPSED TIME: 20.0 s
[INFO] Passcode recovery: checking all 4-digit PINs...

 CUR PASS: [ 1892 ] | AVG SPD: 5.4 p/s | ELAPSED TIME: 157.0 s
[INFO] Passcode found: 1893

Press 'Enter' to continue
```

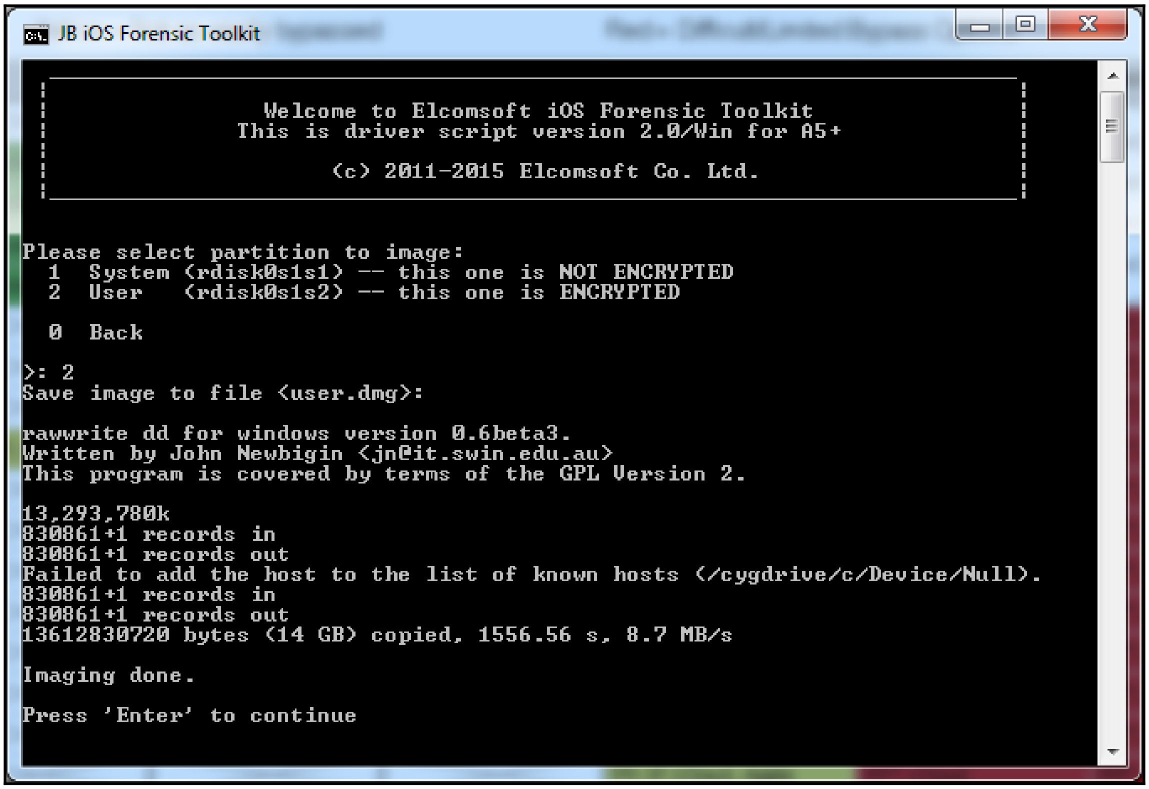
The second step extracts encryption keys from the device; these keys are necessary to decrypt the physical acquisition. It saves the decryption keys in a `plist` file (default name `keys.plist`) and prints the backup password (if it exists) and the AppleID for the device on the screen.



JB iOS Forensic Toolkit

```
 ________________________________________________________________________
|                                                                        |
|                  Welcome to Elcomsoft iOS Forensic Toolkit             |
|                This is driver script version 2.0/Win for A5+           |
|                                                                        |
|                   (c) 2011-2015 Elcomsoft Co. Ltd.                     |
|________________________________________________________________________|

loading dumpkeys utility on device...
Access denied
root@localhost's password:
dumpkeys                    | 647 kB | 647.3 kB/s | ETA: 00:00:00 | 100%
Dumpkeys utility is successfully loaded on device.
Failed to add the host to the list of known hosts (/cygdrive/c/Device/Null).
Continue? (Y/n): Y
Device passcode (optional) <1893>:

Escrow file (optional):

Save data to file (relative to current directory) <keys.plist>:


Extracting device secrets...
Failed to add the host to the list of known hosts (/cygdrive/c/Device/Null).
[INFO] Detected iOS version: 9.0.2
[INFO] Kernel region: 0x88C01000
[INFO] Device Serial Number: DMPKF9JEF193
[INFO] Passcode for the device is "1893".
[INFO] Keychain version: 7
[INFO] Backup password for the device is "genova".
[INFO] AppleID login for the device is "jailbreakingios@icloud.com".


Press 'Enter' to continue
```

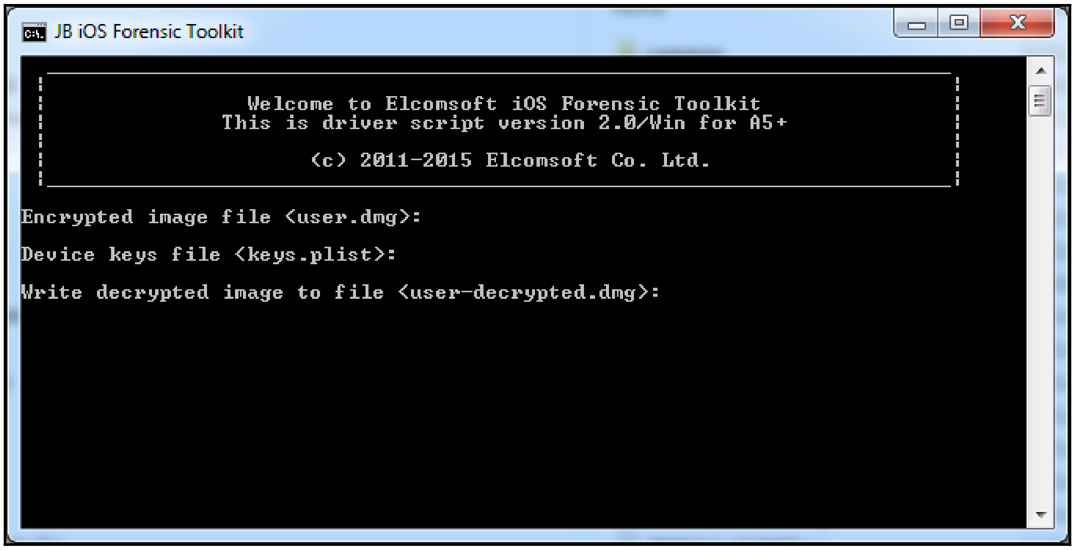
The third step creates a forensic image of the system and data partition. The first one is not encrypted, while the second one is. This step produces in output two DD files; the second one needs to be decrypted.



JB iOS Forensic Toolkit

```
 --------------------------------------------------------------------------
|                                                                          |
|                Welcome to Elcomsoft iOS Forensic Toolkit                 |
|              This is driver script version 2.0/Win for A5+               |
|                                                                          |
|                    (c) 2011-2015 Elcomsoft Co. Ltd.                      |
|                                                                          |
 --------------------------------------------------------------------------

Please select partition to image:
  1  System (rdisk0s1s1) -- this one is NOT ENCRYPTED
  2  User   (rdisk0s1s2) -- this one is ENCRYPTED

  0  Back

>: 2
Save image to file (user.dmg):

rawwrite dd for windows version 0.6beta3.
Written by John Newbigin <jn@it.swin.edu.au>
This program is covered by terms of the GPL Version 2.

13,293,780k
830861+1 records in
830861+1 records out
Failed to add the host to the list of known hosts (/cygdrive/c/Device/Null).
830861+1 records in
830861+1 records out
13612830720 bytes (14 GB) copied, 1556.56 s, 8.7 MB/s

Imaging done.

Press 'Enter' to continue
```

The fourth step takes as input the encrypted data partition and the `keys.plist` file and produces an unencrypted version of the DD image as output. This image can be imported and analyzed in various tools, both specifically for mobile devices (for example, UFED Physical Analyzer, Oxygen Forensic, and Internet Evidence Finder) or traditional forensics toolkit (for example, Encase, FTK, and X-Ways).



JB iOS Forensic Toolkit

```
Welcome to Elcomsoft iOS Forensic Toolkit
This is driver script version 2.0/Win for A5+

(c) 2011-2015 Elcomsoft Co. Ltd.

Encrypted image file <user.dmg>:

Device keys file <keys.plist>:

Write decrypted image to file <user-decrypted.dmg>:
```

Apple support for law enforcement

On a regular basis, Apple publishes a document on its website called *Legal Process Guidelines for U.S. Law Enforcement*. These guidelines contain information on how to request Apple support to recover information from iCloud or from an iDevice, and specify the data that Apple, in some cases, can extract from a passcode-protected device. Apple's latest available version (`https://www.apple.com/privacy/docs/legal-process-guidelines-us.pdf`) states:

“For all devices running iOS 8.0 and later versions, Apple will not perform iOS data extractions as data extraction tools are no longer effective. The files to be extracted are protected by an encryption key that is tied to the user's passcode, which Apple does not possess. For iOS devices running iOS versions earlier than iOS 8.0, upon receipt of a valid search warrant issued upon a showing of probable cause, Apple can extract certain categories of active data from passcode locked iOS devices. Specifically, the user generated active files on an iOS device that are contained in Apple's native apps and for which the data is not encrypted using the passcode (“user generated active files”), can be extracted and provided to law enforcement on external media. Apple can perform this data extraction process on iOS devices running iOS 4 through iOS 7. Please note the only categories of user generated active files that can be provided to law enforcement, pursuant to a valid search warrant, are: SMS, iMessage, MMS, photos, videos, contacts, audio recording, and call history. Apple cannot provide: e-mail, calendar entries, or any third-party app data.”

This method was used by the South Africa police, who requested help from Apple in order to access data stored on Oscar Pistorious' iPhone.

Apple versus FBI - The San Bernardino shooting case

In December 2015, 14 innocent people lost their lives in the San Bernardino terrorist attack. The shooters, killed by the police, had managed to destroy all their devices but one prior to the attack, the work phone of Syed Rizwan Farook.

Farook's phone was an iPhone 5s, running iOS 8, protected with a four-digit pin code, and it was found powered off (although there is quite some confusion on this point, as at a first stage the device appeared to have been found powered on). As you have learned in this chapter, there is no known technique that's able to bypass and unlock a device in this scenario. The only chance the FBI had was to try the luck of guessing/brute forcing the correct PIN within the first nine attempts before the device would wipe itself after 10 incorrect PIN entries. For this reason, on February 2016, the FBI reached Apple with a compelling order (`https://assets.documentcloud.org/documents/27145/SB-Shooter-Order-Compelling-Apple-Asst-iPhone.pdf`) asking to provide assistance to accomplish the following:

1. It will bypass or disable the autoerase function whether or not it has been enabled
2. It will enable the FBI to submit passcodes to the SUBJECT DEVICE for testing electronically via the physical device port, Bluetooth, Wi-Fi, or other protocol available
3. It will ensure that when the FBI submits passcodes to the SUBJECT DEVICE, software running on the device will not purposefully introduce any additional delay between passcode attempts beyond what is incurred by Apple hardware

Furthermore, in the court order, it was also specified that Apple's assistance may include providing software to the FBI that will be coded by Apple with a unique identifier of the phone so that the software would only load and execute on the SUBJECT DEVICE. What all this means is that the FBI asked Apple to develop a backdoor into iOS devices.

Apple refused to comply with the order stating, in an open letter by its CEO, Tim Cook (`http://www.apple.com/customer-letter/`), that:

> *"The United States government has demanded that Apple take an unprecedented step which threatens the security of our customers. We oppose this order, which has implications far beyond the legal case at hand. This moment calls for public discussion, and we want our customers and people around the country to understand what is at stake."*

On March 28, the FBI announced that it had unlocked the suspect's iPhone with a third party's help, without giving any further details on the *who* and the *how*. Coincidently enough, on the very same day, the FBI cut an order of *$218,004.85* to Cellebrite (), which of course led the speculation on the identity of the third party who was able to unlock the iPhone. However, on April 7, the FBI Director, James Comey, indicated that the tool had cost more than $1.3 million and the Washington Post reported, from an anonymous source, that the FBI had paid professional hackers which used a zero-day vulnerability to bypass the 10 PIN attempt limitation, allowing the FBI to brute force the four-digit PIN without the risk of wiping the device (`https://www.washingtonpost.com/world/national-security/fbi-paid-professional-hackers-one-time-fee-to-crack-san-bernardino-iphone/216/4/12/5397814a-de-11e6-9d36-33d198ea26c5_story.html`).

Leaving out the legal and ethical aspects, there are still a few things worth considering. First of all, it is clear that, technically speaking, Apple would have been able to comply with the court order, but as the last point on the resolution proves, there were other ways to obtain the result. Another point to highlight is that the only reason the FBI was able to make such a request is because the user had chosen a weak four-digit PIN. If he had picked a complex passcode instead, it would have been unfeasible to brute force it even without the 10 attempts limitation due to the password derivation function used by Apple (`PBKDF2`), in which the encryption routines implement an 80 ms delay to compute the key, resulting in six years just to brute force a six-digit alphanumeric key, and as an obvious consequence, exponentially more time for longer keys.

Last but not least, there was one big mistake made by the FBI from a forensics point of view. As soon as the iPhone had been recovered, the FBI made the San Bernardino County (which was the legal owner of the phone) reset the iCloud password. If the iPhone had really been found switched off, this would not have made difference since, in order to connect to a Wi-Fi network, a valid access code was required to be entered at least once after reboot. However, if the phone was found powered on, as it may look like from conflicting statements and testimonies, resetting the password made the automatic iCloud backup unfeasible as the iCloud password on the phone would still have been the old (at this point invalid) one.

Finally, this has definitely been a controversial case for different reasons, but nevertheless an interesting case study to learn from.

iOS Acquisition - choose the best method

As you may have probably understood from reading this chapter, being able to choose the right technique to acquire an iOS device is not trivial and it depends on many different parameters. In this section we try to list, for each of the different device families, the best acquisition method available depending on the different versions of the operating system and whether a passcode is set or not.

iPhone 3G/3GS/4, iPad 1

For these devices, it is always advisable to perform a physical acquisition using one of the forensic software that supports it. If a simple passcode is set (four digits), it is always possible to crack it in a short time, and therefore obtain a forensic image of the device's internal memory (both data and system partitions). In the case of a complex passcode being set, it is possible to obtain a physical acquisition, although not all files in it will be decrypted.

iPhone 4s, 5, 5c, iPad 2/3/4, iPad Mini 1

For these devices, we must distinguish according to the presence of the passcode and the operating system version.

In the case of devices with no passcode, it is always possible to carry out the acquisition via the AFC protocol (with varying results depending on the operating system version, but always with the ability to extract the Crash Logs) and via iTunes backup

The user may have set a password on the backup and such a password can be cracked offline as described in `Chapter 4`, *Evidence Acquisition and Analysis from iTunes Backup*.

On devices with iOS 7, it is also possible to acquire via advanced logical.

In the case of devices protected by passcode, we have several techniques depending on the operating system version. Until iOS 7, you can do the following: use a lockdown certificate to carry out the acquisition directly (via AFC, advanced logical, or Backup), use the Apple service in support of law enforcement, use the CAIS service offered by Cellebrite, use a hardware tool to unlock the device (for example, IP-BOX) or detect the presence of an iCloud backup (refer to `Chapter 5`, *Evidence Acquisition and Analysis from iCloud* for details). Starting from iOS 8, it is no longer possible to get support from Apple, while you still can, with certain limitations, use the lockdown certificate (only for backup and AFC acquisitions, and only if the phone was found switched on and unlocked at least once in the past 48 hours) and hardware devices for unlocking (up to iOS 8.1). The use of CAIS service by Cellebrite and iCloud backup is also a viable option.

iPhone 5s, 6, 6Plus, 6s, 6s Plus, iPad Air 1/2, iPad Mini 2/3/4, iPad Pro

Similar to the previous section, an unlocked device can always be acquired via the AFC protocol and iTunes backup. For devices such as iPhone 5s, iPad Air, and iPad Mini 2 with iOS 7 it is also possible to acquire via advanced logical.

In the case of locked devices, we have to distinguish by device and operating system version.

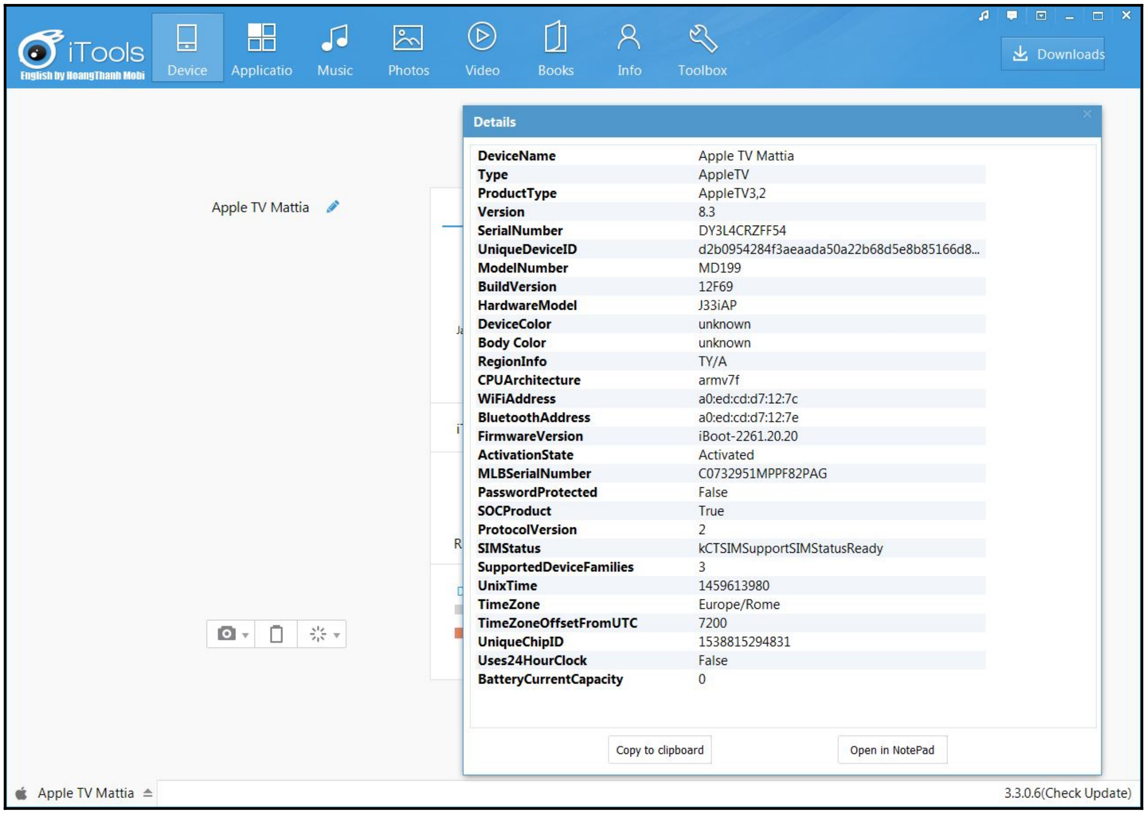
For devices such as iPhone 5s, iPad Air, and iPad Mini 2 with iOS 7, it is possible to use the lockdown certificate, request support from Apple, or use unlocking hardware devices.

For all devices with iOS 8 or 9, the only practicable roads are using a valid lockdown certificate or acquiring iCloud backups.

Apple TV

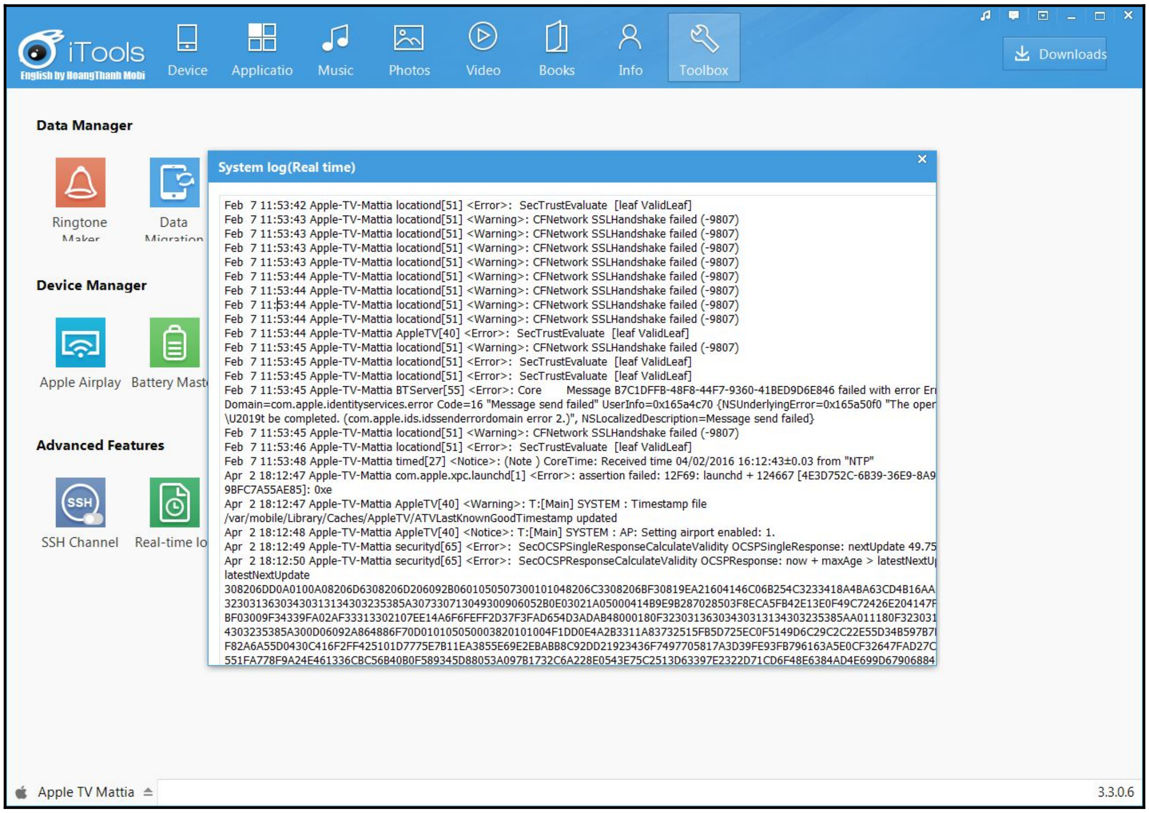
The acquisition of data from Apple TV devices is particularly complex, as there are no available native backup procedures. Tests carried out by the authors on the Apple TV version 3 have allowed us to understand that the AFC protocol is active on the device. Since the device cannot be password protected, this method is always applicable.

The following screenshots show the general details extracted with the iTools software from an Apple TV version 3:



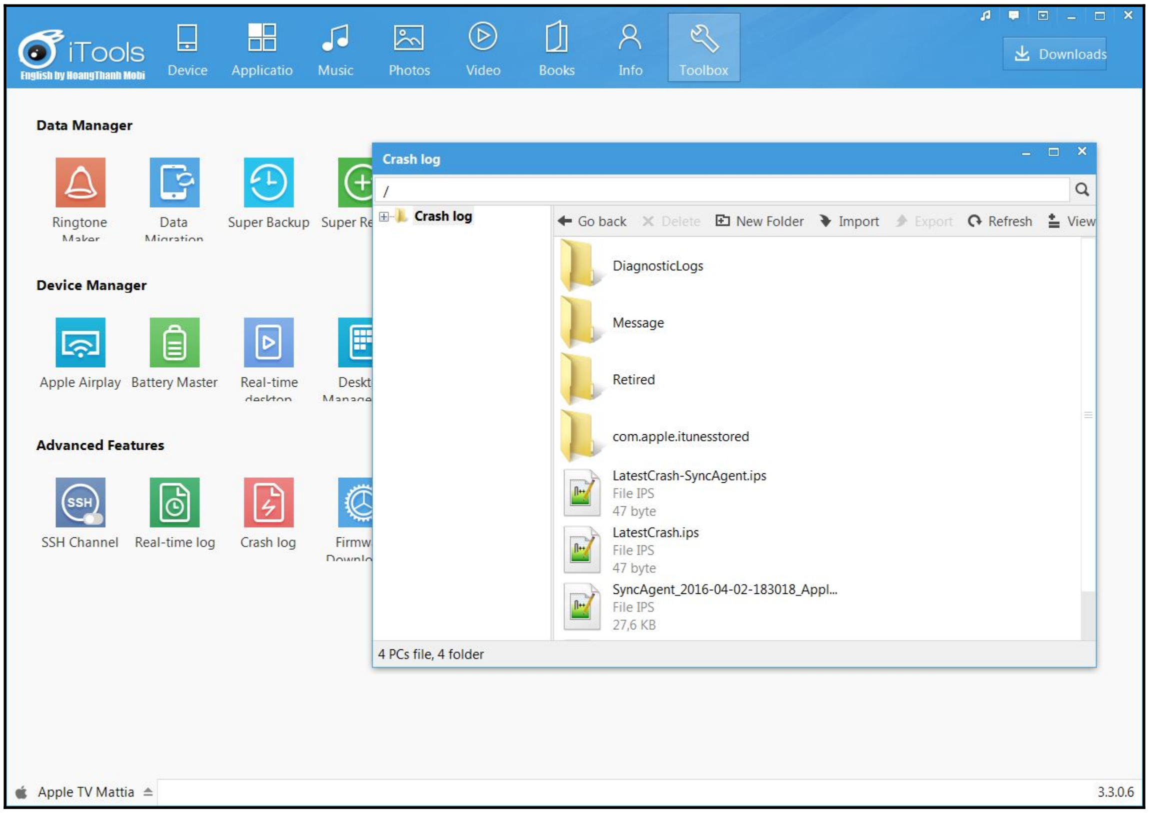
DeviceName	Apple TV Mattia
Type	AppleTV
ProductType	AppleTV3,2
Version	8.3
SerialNumber	DY3L4CRZFF54
UniqueDeviceID	d2b0954284f3aeaada50a22b68d5e8b85166d8...
ModelNumber	MD199
BuildVersion	12F69
HardwareModel	J33iAP
DeviceColor	unknown
Body Color	unknown
RegionInfo	TY/A
CPUArchitecture	armv7f
WiFiAddress	a0:ed:cd:d7:12:7c
BluetoothAddress	a0:ed:cd:d7:12:7e
FirmwareVersion	iBoot-2261.20.20
ActivationState	Activated
MLBSerialNumber	C0732951MPPF82PAG
PasswordProtected	False
SOCProduct	True
ProtocolVersion	2
SIMStatus	kCTSIMSupportSIMStatusReady
SupportedDeviceFamilies	3
UnixTime	1459613980
TimeZone	Europe/Rome
TimeZoneOffsetFromUTC	7200
UniqueChipID	1538815294831
Uses24HourClock	False
BatteryCurrentCapacity	0

Real Time Log permits us to identify the latest activities performed by the device (that is, last days of activity, Wi-Fi connection, and so on).



```
Feb  7 11:53:42 Apple-TV-Mattia locationd[51] <Error>:  SecTrustEvaluate  [leaf ValidLeaf]
Feb  7 11:53:43 Apple-TV-Mattia locationd[51] <Warning>: CFNetwork SSLHandshake failed (-9807)
Feb  7 11:53:43 Apple-TV-Mattia locationd[51] <Warning>: CFNetwork SSLHandshake failed (-9807)
Feb  7 11:53:43 Apple-TV-Mattia locationd[51] <Warning>: CFNetwork SSLHandshake failed (-9807)
Feb  7 11:53:43 Apple-TV-Mattia locationd[51] <Warning>: CFNetwork SSLHandshake failed (-9807)
Feb  7 11:53:44 Apple-TV-Mattia locationd[51] <Warning>: CFNetwork SSLHandshake failed (-9807)
Feb  7 11:53:44 Apple-TV-Mattia locationd[51] <Warning>: CFNetwork SSLHandshake failed (-9807)
Feb  7 11:53:44 Apple-TV-Mattia locationd[51] <Warning>: CFNetwork SSLHandshake failed (-9807)
Feb  7 11:53:44 Apple-TV-Mattia locationd[51] <Warning>: CFNetwork SSLHandshake failed (-9807)
Feb  7 11:53:44 Apple-TV-Mattia AppleTV[40] <Error>:  SecTrustEvaluate  [leaf ValidLeaf]
Feb  7 11:53:45 Apple-TV-Mattia locationd[51] <Warning>: CFNetwork SSLHandshake failed (-9807)
Feb  7 11:53:45 Apple-TV-Mattia locationd[51] <Error>:  SecTrustEvaluate  [leaf ValidLeaf]
Feb  7 11:53:45 Apple-TV-Mattia locationd[51] <Error>:  SecTrustEvaluate  [leaf ValidLeaf]
Feb  7 11:53:45 Apple-TV-Mattia BTServer[55] <Error>: Core       Message B7C1DFFB-48F8-44F7-9360-41BED9D6E846 failed with error Er
Domain=com.apple.identityservices.error Code=16 "Message send failed" UserInfo=0x165a4c70 {NSUnderlyingError=0x165a50f0 "The oper
\U2019t be completed. (com.apple.ids.idssenderrordomain error 2.)", NSLocalizedDescription=Message send failed}
Feb  7 11:53:45 Apple-TV-Mattia locationd[51] <Warning>: CFNetwork SSLHandshake failed (-9807)
Feb  7 11:53:46 Apple-TV-Mattia locationd[51] <Error>:  SecTrustEvaluate  [leaf ValidLeaf]
Feb  7 11:53:48 Apple-TV-Mattia timed[27] <Notice>: (Note ) CoreTime: Received time 04/02/2016 16:12:43±0.03 from "NTP"
Apr  2 18:12:47 Apple-TV-Mattia com.apple.xpc.launchd[1] <Error>: assertion failed: 12F69: launchd + 124667 [4E3D752C-6B39-36E9-8A9
9BFC7A55AE85]: 0xe
Apr  2 18:12:47 Apple-TV-Mattia AppleTV[40] <Warning>: T:[Main] SYSTEM : Timestamp file
/var/mobile/Library/Caches/AppleTV/ATVLastKnownGoodTimestamp updated
Apr  2 18:12:48 Apple-TV-Mattia AppleTV[40] <Notice>: T:[Main] SYSTEM : AP: Setting airport enabled: 1.
Apr  2 18:12:49 Apple-TV-Mattia securityd[65] <Error>:  SecOCSPSingleResponseCalculateValidity OCSPSingleResponse: nextUpdate 49.75
Apr  2 18:12:50 Apple-TV-Mattia securityd[65] <Error>:  SecOCSPResponseCalculateValidity OCSPResponse: now + maxAge > latestNextU
latestNextUpdate
308206DD0A0100A08206D6308206D206092B0601050507300101048206C3308206BF30819EA21604146C06B254C3233418A4BA63CD4B16AA
32303136303430313134303235385A30733071304930090605280E03021A05000414B9E9B287028503F8ECA5FB42E13E0F49C72426E204147
BF03009F34339FA02AF33313302107EE14A6F6FEFF2D37F3FAD654D3ADAB48000180F32303136303430313134303235385AA011180F323031
4303235385A300D06092A864886F70D010105050003820101004F1DD0E4A2B3311A83732515FB5D725EC0F5149D6C29C2C22E55D34B597B7
F82A6A55D0430C416F2FF425101D7775E7B11EA3855E69E2EBABB8C92DD21923436F7497705817A3D39FE93FB796163A5E0CF32647FAD27C
551FA778F9A24E461336CBC56B40B0F589345D88053A097B1732C6A228E0543E75C2513D63397E2322D71CD6F48E6384AD4E699D67906884
```

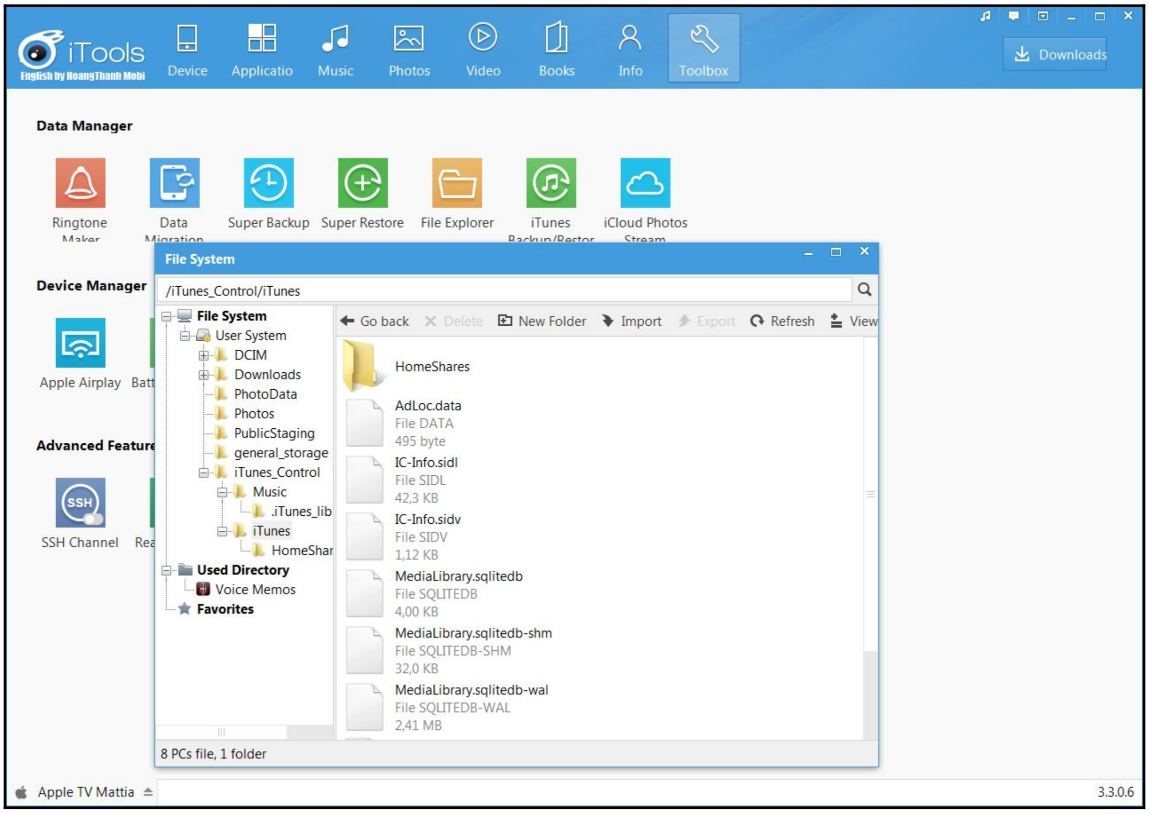
With iTools, it is also possible to extract Crash Logs, which allow the reconstruction of a timeline of the device's usage.



DiagnosticLogs
Message
Retired
com.apple.itunesstored
LatestCrash-SyncAgent.ips File IPS 47 byte
LatestCrash.ips File IPS 47 byte
SyncAgent_2016-04-02-183018_Appl... File IPS 27,6 KB

4 PCs file, 4 folder

Finally, it is also possible to extract a limited part of the filesystem containing, for example, the iTunes library, from which you can locate the account information used and the purchases made.



/iTunes_Control/iTunes

```
HomeShares
AdLoc.data                 File DATA          495 byte
IC-Info.sidl               File SIDL          42,3 KB
IC-Info.sidv               File SIDV          1,12 KB
MediaLibrary.sqlitedb      File SQLITEDB      4,00 KB
MediaLibrary.sqlitedb-shm  File SQLITEDB-SHM  32,0 KB
MediaLibrary.sqlitedb-wal  File SQLITEDB-WAL  2,41 MB
```

8 PCs file, 1 folder

Apple Watch

The acquisition of the Apple Watch data is still a little explored and tested area. Of particular interest is the presentation of Sarah Edwards and Heather Mahalik *Times a' Ticking… to Forensicate the Apple Watch*, in 2015. The presentation shows the test results of the analysis of an iPhone backup device synchronized with an Apple Watch and the identification of different information of interest (information about the device, installed applications, address book, e-mails, voicemails, passbook, and so on)

Summary

In this chapter, we introduced the four most commonly used methods to acquire data from iDevices-AFC, iTunes backup, advanced logical, and physical. The backup acquisition can be performed on any device but the device needs to be unlocked, the analyst needs to know the passcode, or the analyst has a valid lockdown certificate extracted from a computer the device was previously synced with. If the user sets a password on the backup, the resulting acquisition is encrypted and so the analyst needs to try cracking the backup password (this topic is covered in detail in `Chapter 4`, *Evidence Acquisition and Analysis from iTunes Backup*). If the device is locked and the analyst doesn't know the code, or he/she doesn't have a valid lockdown certificate, other techniques are available request Apple support (up to iOS 7), use a hardware unlocking tool (up to iOS 8.1), use a third-party service (up to iOS 9.3.2 and limited to 32-bit devices) or search for a backup on iCloud.

The advanced logical acquisition can be performed with the same conditions of the backup acquisition, but it can bypass the restrictions imposed by the backup password and extract the contents in clear text without the need to crack the backup password. Apple stopped this method from working starting from iOS 8.0.

Physical acquisition depends on the device and the operating system installed as follows:

- iPhone 2G/3G/3GS and iPod touch 1/2 with iOS 3 don't implement encryption and so it is always possible to perform physical acquisition and the lock code can be cracked instantaneously. The resulting image is not encrypted. So, it is possible to carve deleted records.
- On iPhone 3GS/4, iPad 1, and iPod touch 3/4 with iOS 4/5/6/7, it is always possible to perform physical acquisition. If the lock code is four digits long, it can be cracked in less than 20 minutes. So, it is possible to recover all the files. If a complex passcode is in use, the analyst can try to crack it with a brute force or dictionary attack. If it's not possible to crack it, it is possible to perform physical acquisition, and decode the filesystem (with the extracted filesystem key) and all the files whose encryption does not depend on passcode.
- On iPhone 4s/5/5c, iPad2/3/4, iPad mini 1, and iPod touch 5 with iOS 4/5/6/7/8/9, physical acquisition is possible only if the device is already jailbroken or if it is possible to jailbreak it (this means that the analyst must know the code).
- On iPhone 5s/6/6 Plus/6s/6s Plus, iPad Air 1/2, iPad mini 2/3/4, and iPad Pro, it is not currently possible to perform a physical acquisition although there are studies and research on it. If the device is jailbroken it is possible, to extract a full filesystem dump.

In `Chapter 4`, *Evidence Acquisition and Analysis from iTunes Backup*, we will see how to acquire and analyze forensics evidences in the case of an iTunes Backup.

Self-test questions

Q1. What is the mode in which iOS devices operate to upgrade the operating system called?

1. Normal
2. Recovery
3. Device Firmware Upgrade
4. Update

Q2. Where are the lockdown certificates stored on Windows 7/8?

1. `C:\Program Data\Apple\Lockdown`
2. `C:\Users\[username]\AppData\Roaming\Apple Computer\Lockdown`
3. `C:\Users\[username]\AppData\Local\Apple Computer\Lockdown`
4. `C:\Windows\Apple Computer\Lockdown`

Q3. Which of the following tools can be used to perform physical acquisition of a jailbroken iPhone 4s?

1. iOS Forensic Toolkit
2. Oxygen Forensics Suite
3. Cellebrite UFED Touch
4. Mobile Phone Examiner

Q4. What is the latest iPhone model that can be physically acquired even if it is not jailbroken?

1. iPhone 3GS
2. iPhone 4
3. iPhone 5
4. iPhone 6

Q5. What is the device identifier for iOS devices called?

1. ECID
2. UDID
3. Serial Number
4. MAC Address

4
Evidence Acquisition and Analysis from iTunes Backup

The goal of this chapter is to introduce you to the different types of local backups (encrypted or unencrypted), the structure of a backup, the techniques and software available to extract meaningful data from it, and to show you how to crack an encrypted backup while extracting the password saved in it. These concepts are really useful because sometimes the analyst may not have the iOS device or cannot access it, but may have access to a computer containing an iTunes backup.

iTunes backup

The Apple iTunes software allows users to create two different types of local (meaning stored on a local computer) backup of their iOS devices-encrypted and unencrypted. An unencrypted backup is completely accessible, while an encrypted one is protected with a password chosen by the owner of the device. The first time that the user sets a password for the backup, it is saved inside the iDevice, and every subsequent backup is encrypted with the same password (until the user decides to change or remove it). For this reason, if a password has already been set when performing a forensic acquisition, we would get an encrypted backup (refer to `Chapter 3`, *Evidence Acquisition from iDevices*, for the different techniques used to acquire a device with a backup password set).

iTunes backup folders

The folder where the backup data is stored depends on your computer's operating system. iTunes saves the backup files in these folders:

- **Mac**: `~/Library/Application Support/MobileSync/Backup/`
- **Windows XP**: `\Documents and Settings\(username)\Application Data\Apple Computer\MobileSync\Backup\`
- **Windows Vista, Windows 7, Windows 8, and Windows 10**: `\Users\(username)\AppData\Roaming\Apple Computer\MobileSync\Backup\`

Inside these folders, there is a subfolder for each iDevice that has been backed up with the same computer. The name of the subfolder is equivalent to the UDID of the device, which is a 40-character-long hexadecimal string. This means that iTunes holds only one backup for each device and copies only the files that have been modified since the last backup. When a device is updated to a new OS version and then restored, the last backup created before the update is not overwritten the first time you create a new backup. In particular, the old backup folder is renamed by appending the timestamp of the backup at the end of the folder name.

iTunes backup content

According to Apple specifications (refer to the article available at `http://support.apple.com/kb/ht4946`, as mentioned in `Appendix A`, *References*), an iTunes backup includes nearly all of your device's data and settings, but doesn't contain the following:

- Content from the iTunes and App Stores, or PDFs downloaded directly to iBooks
- Content synced from iTunes, such as imported MP3s, videos, books, and photos
- Photos already stored in the cloud, such as My Photo Stream, and iCloud Photo Library
- Touch ID settings
- Apple Pay information and settings
- Activity, Health, Website, and Keychain data (although these elements are available in an encrypted backup)

One of the main differences between an unencrypted backup and an encrypted one are related to the `Keychain` file. Inside an unencrypted backup, this file is saved encrypted with a key that depends on the device's UID, and therefore cannot be cracked offline or reactivated on a different device from the one used to generate the backup. Instead, in an encrypted backup, the `Keychain` file is encrypted with the backup password. This can be technically explained as follows:

- If the device does not have a backup password set by the user, when performing the acquisition, it is possible to create an encrypted backup choosing a known password, and later be able to access the passwords saved in the Keychain without the need to crack anything
- If the device has a backup password set by the user, when performing the acquisition, it is possible to create an encrypted backup, and then trying to crack the password in order to extract those saved in the keychain

In particular, the `Keychain` file contains the following types of passwords:

- Passwords and settings of the Wi-Fi networks the device has been connected to
- Passwords of the e-mail accounts configured in Apple Mail
- VPN credentials
- Credentials (passwords or tokens) of all third-party apps that use keychain as the password container

iTunes backup structure

In a backup folder, there are some standard files with fixed names and contents and hundreds of files with long hashed filenames consisting of 40 hexadecimal characters. The file name acts like a unique identifier for every file copied from the iDevice. In fact, each file is named as the result of an SHA-1 hash calculated on the original full name of the file in the following form:

```
Domain-[subdomain-]fullpath/filename.ext
```

Consider the following example:

```
AppDomain-com.skype.skype-Library/Preferences/com.skype.skype.plist
```

Here, `AppDomain` is the name of domain, `Com.skype.skype` is the subdomain, and `Library/Preferences/com.skype.skype.plist` is the path and the name of file.

Calculating the SHA-1 hash for `AppDomain-com.skype.skype-Library/Preferences/com.skype.skype.plist` gives us `bc0e135b1c68521fa4710e3edadd6e74364fc50a`.

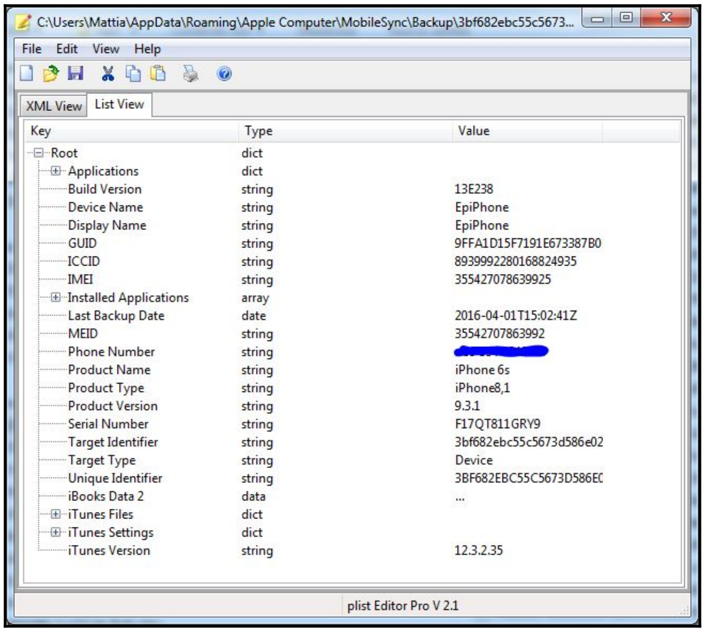
This is actually the 40-character-long string we're talking about in this context.

The meaning of the elements named in the domain and subdomain is explained later in this chapter.

Standard backup files

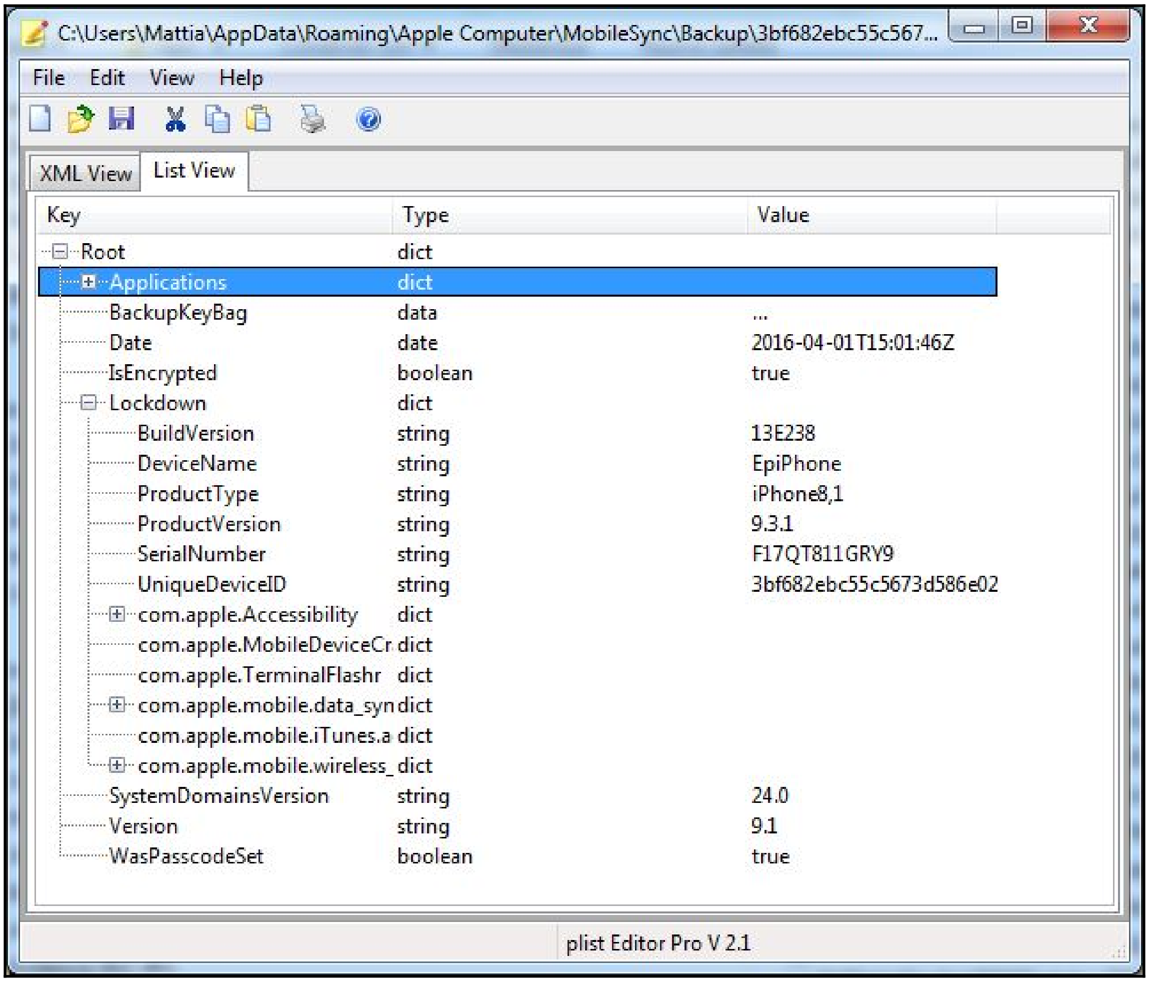
These files are created by the backup service and store information about the backup itself. The most useful files are as follows:

- `Info.plist`: This is a `plist` file in plain text and stores data about the backed up device (such as date of backup creation, phone number, device name, GUID, ICCID, IMEI, product type, iOS version, serial numbers, sync settings, UDID, and so on) and the iTunes software used to create the backup (iTunes version number and iTunes settings):


C:\Users\Mattia\AppData\Roaming\Apple Computer\MobileSync\Backup\3bf682ebc55c5673...
File Edit View Help
XML View
List View
plist Editor Pro V 2.1

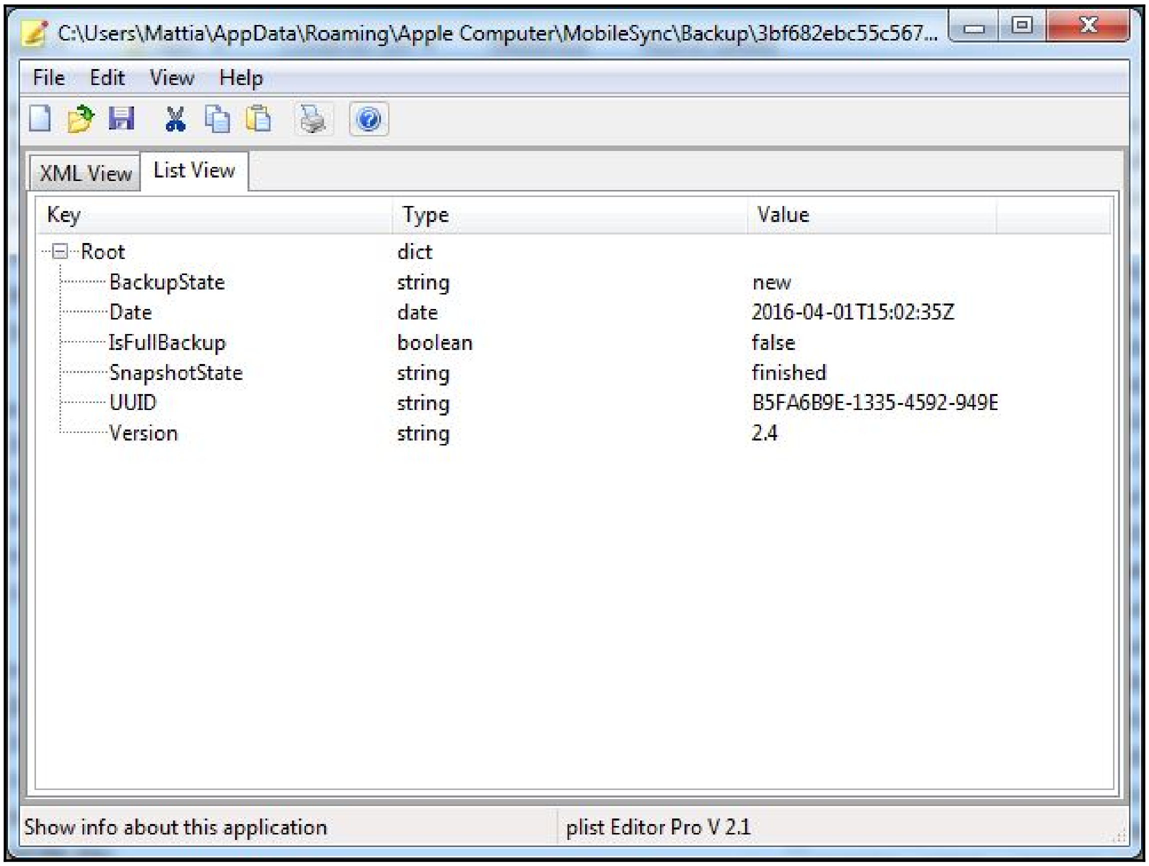
Key	Type	Value
Root	dict	
Applications	dict	
Build Version	string	13E238
Device Name	string	EpiPhone
Display Name	string	EpiPhone
GUID	string	9FFA1D15F7191E673387B0
ICCID	string	8939992280168824935
IMEI	string	355427078639925
Installed Applications	array	
Last Backup Date	date	2016-04-01T15:02:41Z
MEID	string	35542707863992
Phone Number	string	[redacted]
Product Name	string	iPhone 6s
Product Type	string	iPhone8,1
Product Version	string	9.3.1
Serial Number	string	F17QT811GRY9
Target Identifier	string	3bf682ebc55c5673d586e02
Target Type	string	Device
Unique Identifier	string	3BF682EBC55C5673D586EC
iBooks Data 2	data	...
iTunes Files	dict	
iTunes Settings	dict	
iTunes Version	string	12.3.2.35

- `Manifest.plist`: This is a `plist` file and it describes the content of the backup. Inside this file, we can find the list of applications installed on the backed up device. For every application, there is the name and the particular version. Inside the file, there is also the date the backup was made, the backup type (encrypted versus unencrypted), and some information about the iDevice (for example, if the passcode was set on the device at the time of backup) and the iTunes software used:


C:\Users\Mattia\AppData\Roaming\Apple Computer\MobileSync\Backup\3bf682ebc55c567...
File Edit View Help
XML View
List View
plist Editor Pro V 2.1

Key	Type	Value
Root	dict	
Applications	dict	
BackupKeyBag	data	...
Date	date	2016-04-01T15:01:46Z
IsEncrypted	boolean	true
Lockdown	dict	
BuildVersion	string	13E238
DeviceName	string	EpiPhone
ProductType	string	iPhone8,1
ProductVersion	string	9.3.1
SerialNumber	string	F17QT811GRY9
UniqueDeviceID	string	3bf682ebc55c5673d586e02
com.apple.Accessibility	dict	
com.apple.MobileDeviceCr	dict	
com.apple.TerminalFlashr	dict	
com.apple.mobile.data_syn	dict	
com.apple.mobile.iTunes.a	dict	
com.apple.mobile.wireless_	dict	
SystemDomainsVersion	string	24.0
Version	string	9.1
WasPasscodeSet	boolean	true

- `Status.plist`: This is a `plist` file in the binary format, and it stores information about the status of completion of the backup, whether the backup was made successfully or not:


C:\Users\Mattia\AppData\Roaming\Apple Computer\MobileSync\Backup\3bf682ebc55c567...
File
Edit
View
Help
XML View
List View

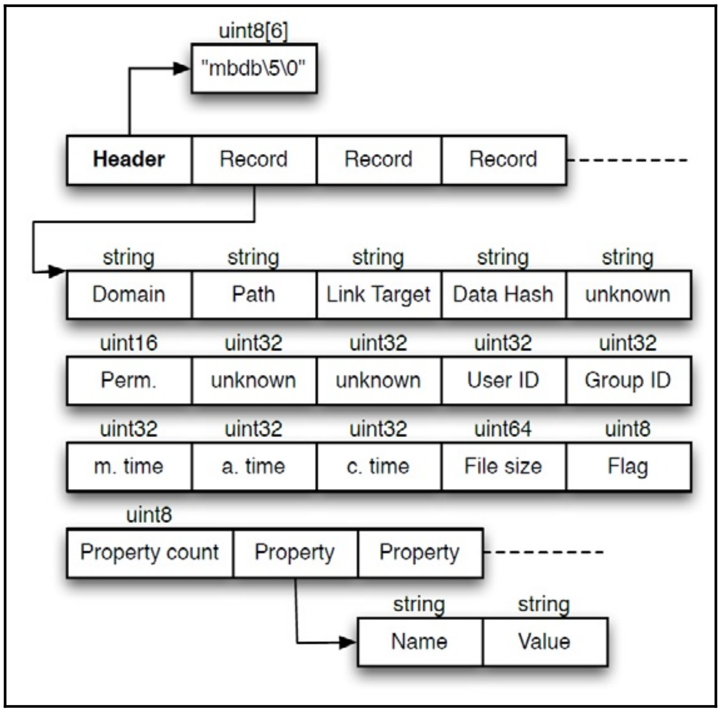
Key	Type	Value
Root	dict	
BackupState	string	new
Date	date	2016-04-01T15:02:35Z
IsFullBackup	boolean	false
SnapshotState	string	finished
UUID	string	B5FA6B9E-1335-4592-949E
Version	string	2.4

Show info about this application

plist Editor Pro V 2.1

- `Manifest.mbdb`: This is a binary file that stores the descriptions of all the other files in the backup folder. It contains a record for each element in the backup (including symbolic link and folders, which of course don't have a corresponding element among the backup files). Each record contains the following parameters:
    - **Domain**: This shows the domain the element belongs to. Domains are a way to functionally categorize elements in the device backup.
    - **Path**: This shows the full path of the element.
    - **Link Target**: This shows the target of the element if the element itself is a symbolic link.
    - **DataHash**: This contains the SHA-1 hash of file content.
    - **User ID and Group ID**: These contain the owner and the group IDs.
    - **m. time**: This shows the time (in the Unix time format) when the actual content of the file was last modified.
    - **a. time**: This shows the time when the file was last accessed.
    - **c. time**: This shows the time when changes were last made to the file or to the folder's node.
    - **Length**: This shows the size of the file in bytes (size is 0 for link or folder).
    - **Mode and inode**: These contain the Unix file permissions and the inode number. A really interesting thing to note from a forensics point of view is that these four files are stored unencrypted , even if the backup is encrypted with password. It means that the information contained there is accessible without cracking the password.

For a detailed explanation of the analysis of an encrypted backup, we suggest you read the research performed by Hal Pomeranz (refer to `Appendix A`, *References*). The preceding parameters are explained in the following diagram:



The first level of the hierarchy of the backup files is their domain. The domain for each file is written in its corresponding record in the `Manifest.mbdb` file. Each file has a domain name chosen from the following list:

- **App domain**: This contains data related to the installed apps.
- **App Group domain**: This is found starting from iOS 9 and contains specific data for third-party apps.

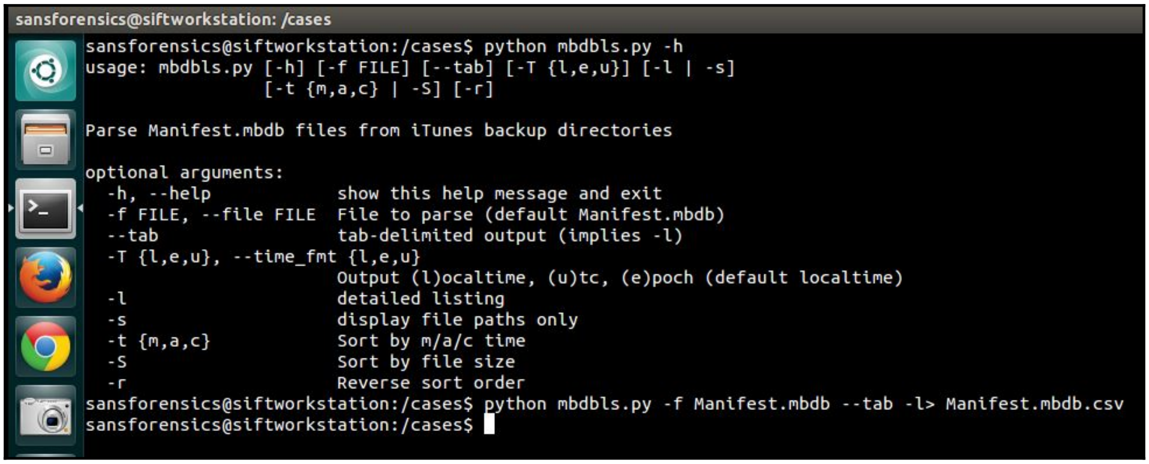
- **App Plugin domain**: This is found starting from iOS 9 and contains plugins of third-party apps.
- **Camera Roll domain**: This contains multimedia elements related to the Camera application, such as images, videos, video previews, and image thumbnails.
- **Health domain**: This is found starting from iOS 9 and contains personal health data. This domain is available only in encrypted backup.

- **Home domain**: This contains data related to the standard application that comes preinstalled with iOS.
- **Keychain domain**: This contains encrypted data related to the keychain.
- **Managed Preferences domain**: This does not typically contain data.
- **Media domain**: This contains multimedia elements not related to the Camera application, such as multimedia elements from MMSes and audio recordings.
- **Mobile Device domain**: This contains the provisioning profiles.
- **Root domain**: This contains cache data related to the geolocation capabilities of the device.
- **System Preferences domain**: This contains configuration files for core components of iOS.
- **Wireless domain**: This contains data about the mobile phone component of the device.

Elements in the App domain are further divided into subdomains related to the applications they belong to, while elements in the other domains don't use this feature. When the subdomain is used, the domain string is written as `<domain>-<subdomain>`. Details about the backup structure are available at `https://theiphonewiki.com/wiki/ITunes_Backup`.

Case study – parsing Manifest.mbdb with Mbdbls Python script

Mbdbls is an open source Python script written by Hal Pomeranz and available on his GitHub account (refer to `Appendix B`, *Tools For iOS Forensics*). It parses a `Maninfest.mbdb` file and for each file, it extracts domain, path and filename, creation date, last accessed date, last modified date, and size. The following screenshot shows the execution of Mbdbls on a SANS SIFT workstation VM against a `Manifest.mbdb`:



```
sansforensics@siftworkstation: /cases
sansforensics@siftworkstation:/cases$ python mbdbls.py -h
usage: mbdbls.py [-h] [-f FILE] [--tab] [-T {l,e,u}] [-l | -s]
                 [-t {m,a,c} | -S] [-r]

Parse Manifest.mbdb files from iTunes backup directories

optional arguments:
  -h, --help            show this help message and exit
  -f FILE, --file FILE  File to parse (default Manifest.mbdb)
  --tab                 tab-delimited output (implies -l)
  -T {l,e,u}, --time_fmt {l,e,u}
                        Output (l)ocaltime, (u)tc, (e)poch (default localtime)
  -l                    detailed listing
  -s                    display file paths only
  -t {m,a,c}            Sort by m/a/c time
  -S                    Sort by file size
  -r                    Reverse sort order
sansforensics@siftworkstation:/cases$ python mbdbls.py -f Manifest.mbdb --tab -l> Manifest.mbdb.csv
sansforensics@siftworkstation:/cases$ 
```

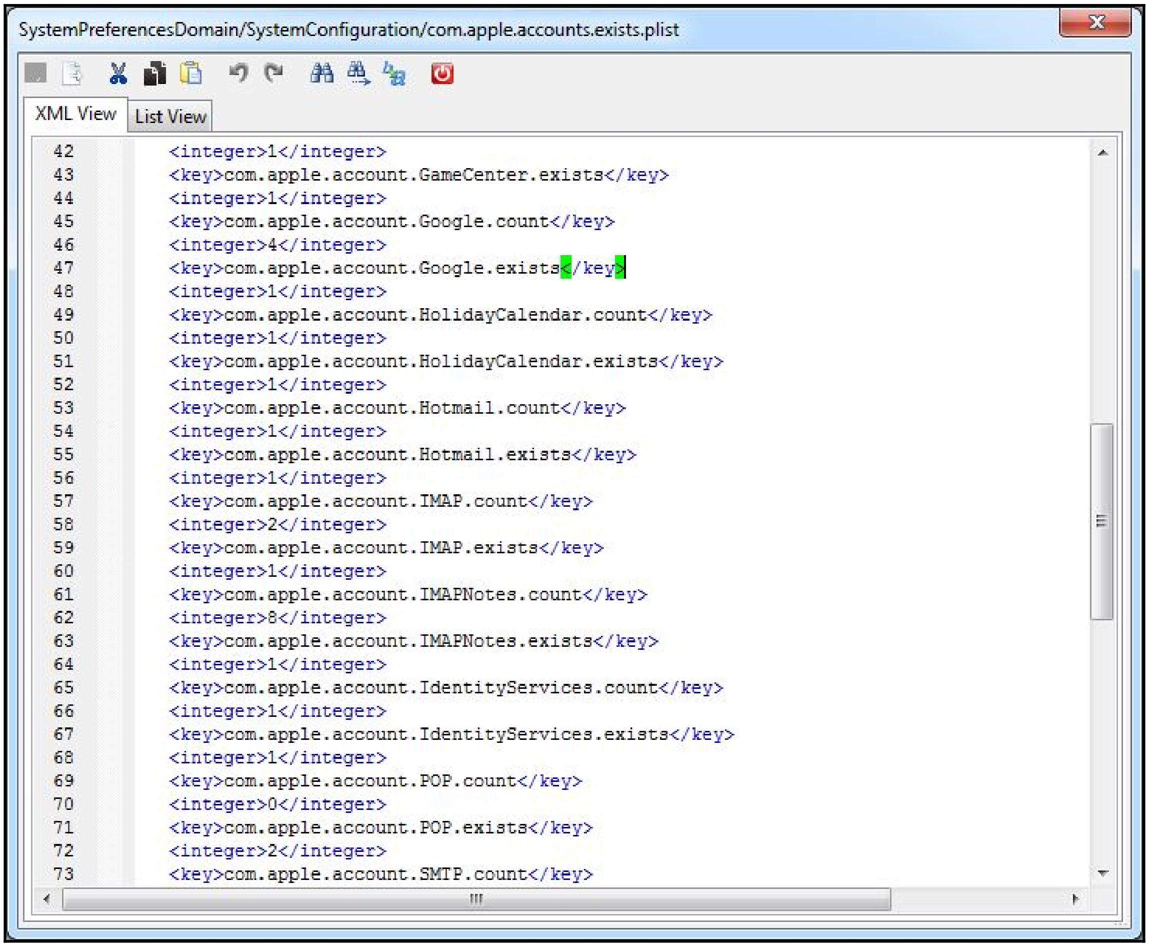
The tool provides various options for the output: for example, it can order by creation, last modified or last accessed dates or by file size. Be careful to take into consideration the appropriate time zone setting.

iTunes backup relevant files

This section provides a complete overview of the most relevant files and folders available in a backup. For every file or folder, the full path (`domain-subdomain`) and a description are provided. Only the system and pre-installed apps are listed. A more detailed analysis of these files is provided in `Chapter 6`, *Analyzing iOS Devices:*

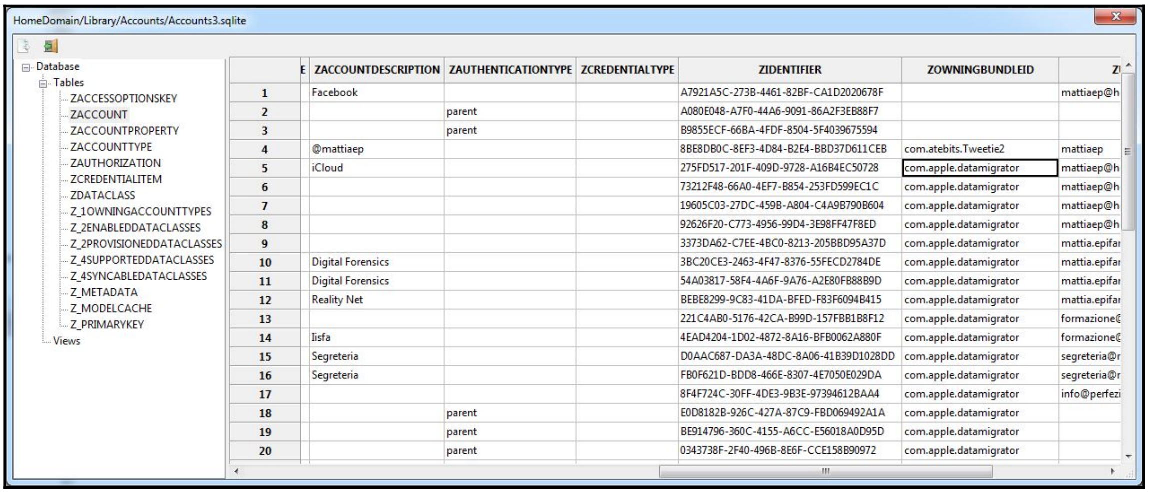
- Domain: `SystemPreferencesDomain/SystemConfiguration/`
- Filename: `com.apple.accounts.exists.plist`

- Description: Information about accounts configured in the device and grouped by type (for example, Apple, Google, Facebook, E-mail, and so on).


SystemPreferencesDomain/SystemConfiguration/com.apple.accounts.exists.plist

```
    <integer>1</integer>
    <key>com.apple.account.GameCenter.exists</key>
    <integer>1</integer>
    <key>com.apple.account.Google.count</key>
    <integer>4</integer>
    <key>com.apple.account.Google.exists</key>
    <integer>1</integer>
    <key>com.apple.account.HolidayCalendar.count</key>
    <integer>1</integer>
    <key>com.apple.account.HolidayCalendar.exists</key>
    <integer>1</integer>
    <key>com.apple.account.Hotmail.count</key>
    <integer>1</integer>
    <key>com.apple.account.Hotmail.exists</key>
    <integer>1</integer>
    <key>com.apple.account.IMAP.count</key>
    <integer>2</integer>
    <key>com.apple.account.IMAP.exists</key>
    <integer>1</integer>
    <key>com.apple.account.IMAPNotes.count</key>
    <integer>8</integer>
    <key>com.apple.account.IMAPNotes.exists</key>
    <integer>1</integer>
    <key>com.apple.account.IdentityServices.count</key>
    <integer>1</integer>
    <key>com.apple.account.IdentityServices.exists</key>
    <integer>1</integer>
    <key>com.apple.account.POP.count</key>
    <integer>0</integer>
    <key>com.apple.account.POP.exists</key>
    <integer>2</integer>
    <key>com.apple.account.SMTP.count</key>
```

- Domain: `HomeDomain/Library/Accounts/`
- Filename: `Account3.sqlite`
- Description: Details of the accounts configured in the device (for example, username and type of stored credentials such as password, OAuth, and so on).

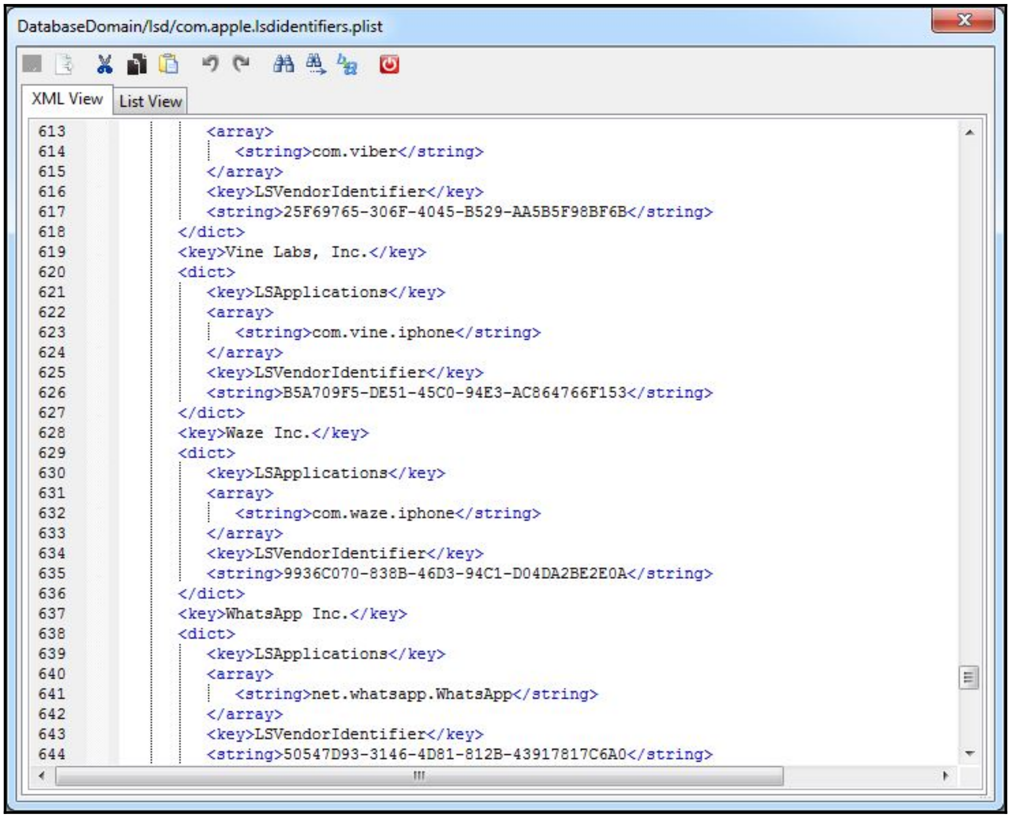


HomeDomain/Library/Accounts/Accounts3.sqlite

- Database
  - Tables
    - ZACCESSOPTIONSKEY
    - ZACCOUNT
    - ZACCOUNTPROPERTY
    - ZACCOUNTTYPE
    - ZAUTHORIZATION
    - ZCREDENTIALITEM
    - ZDATACLASS
    - Z_1OWNINGACCOUNTTYPES
    - Z_2ENABLEDDATACLASSES
    - Z_2PROVISIONEDDATACLASSES
    - Z_4SUPPORTEDDATACLASSES
    - Z_4SYNCABLEDATACLASSES
    - Z_METADATA
    - Z_MODELCACHE
    - Z_PRIMARYKEY
  - Views

	E	ZACCOUNTDESCRIPTION	ZAUTHENTICATIONTYPE	ZCREDENTIALTYPE	ZIDENTIFIER	ZOWNINGBUNDLEID	Z
1		Facebook			A7921A5C-273B-4461-82BF-CA1D2020678F		mattiaep@h
2			parent		A080E048-A7F0-44A6-9091-86A2F3EB88F7		
3			parent		B9855ECF-66BA-4FDF-8504-5F4039675594		
4		@mattiaep			8BE8DB0C-8EF3-4D84-B2E4-BBD37D611CEB	com.atebits.Tweetie2	mattiaep
5		iCloud			275FD517-201F-409D-9728-A16B4EC50728	com.apple.datamigrator	mattiaep@h
6					73212F48-66A0-4EF7-B854-253FD599EC1C	com.apple.datamigrator	mattiaep@h
7					19605C03-27DC-459B-A804-C4A9B790B604	com.apple.datamigrator	mattiaep@h
8					92626F20-C773-4956-99D4-3E98FF47F8ED	com.apple.datamigrator	mattiaep@h
9					3373DA62-C7EE-4BC0-8213-205BBD95A37D	com.apple.datamigrator	mattia.epifar
10		Digital Forensics			3BC20CE3-2463-4F47-8376-55FECD2784DE	com.apple.datamigrator	mattia.epifar
11		Digital Forensics			54A03817-58F4-4A6F-9A76-A2E80FB88B9D	com.apple.datamigrator	mattia.epifar
12		Reality Net			BEBE8299-9C83-41DA-BFED-F83F6094B415	com.apple.datamigrator	mattia.epifar
13					221C4AB0-5176-42CA-B99D-157FBB1B8F12	com.apple.datamigrator	formazione@
14		Iisfa			4EAD4204-1D02-4872-8A16-BFB0062A880F	com.apple.datamigrator	formazione@
15		Segreteria			D0AAC687-DA3A-48DC-8A06-41B39D1028DD	com.apple.datamigrator	segreteria@r
16		Segreteria			FB0F621D-BDD8-466E-8307-4E7050E029DA	com.apple.datamigrator	segreteria@r
17					8F4F724C-30FF-4DE3-9B3E-97394612BAA4	com.apple.datamigrator	info@perfezi
18			parent		E0D8182B-926C-427A-87C9-FBD069492A1A	com.apple.datamigrator	
19			parent		BE914796-360C-4155-A6CC-E56018A0D95D	com.apple.datamigrator	
20			parent		0343738F-2F40-496B-8E6F-CCE158B90972	com.apple.datamigrator	

- Domain: `DatabaseDomain/lsd/`
- Filename: `com.apple.lsdidentifiers.plist`
- Description: Launch Services Daemon Identifiers information and correlation between the application name and GUID.



DatabaseDomain/lsd/com.apple.lsdidentifiers.plist

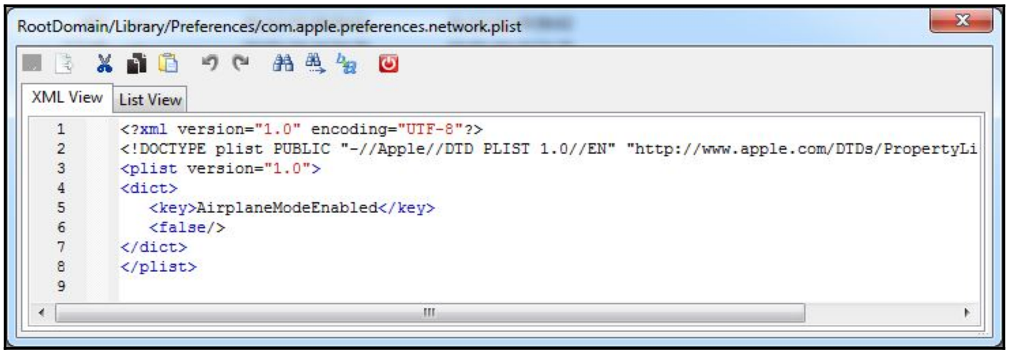
XML View | List View

```
            <array>
                <string>com.viber</string>
            </array>
            <key>LSVendorIdentifier</key>
            <string>25F69765-306F-4045-B529-AA5B5F98BF6B</string>
        </dict>
        <key>Vine Labs, Inc.</key>
        <dict>
            <key>LSApplications</key>
            <array>
                <string>com.vine.iphone</string>
            </array>
            <key>LSVendorIdentifier</key>
            <string>B5A709F5-DE51-45C0-94E3-AC864766F153</string>
        </dict>
        <key>Waze Inc.</key>
        <dict>
            <key>LSApplications</key>
            <array>
                <string>com.waze.iphone</string>
            </array>
            <key>LSVendorIdentifier</key>
            <string>9936C070-838B-46D3-94C1-D04DA2BE2E0A</string>
        </dict>
        <key>WhatsApp Inc.</key>
        <dict>
            <key>LSApplications</key>
            <array>
                <string>net.whatsapp.WhatsApp</string>
            </array>
            <key>LSVendorIdentifier</key>
            <string>50547D93-3146-4D81-812B-43917817C6A0</string>
```

- Domain: `HealthDomain/Health`
- Filename: `healthdb.sqlite`
- Description: List of devices and applications that accessed/updates user's health information from `healthdb_secure.sqlite`, and related timestamp of the event.

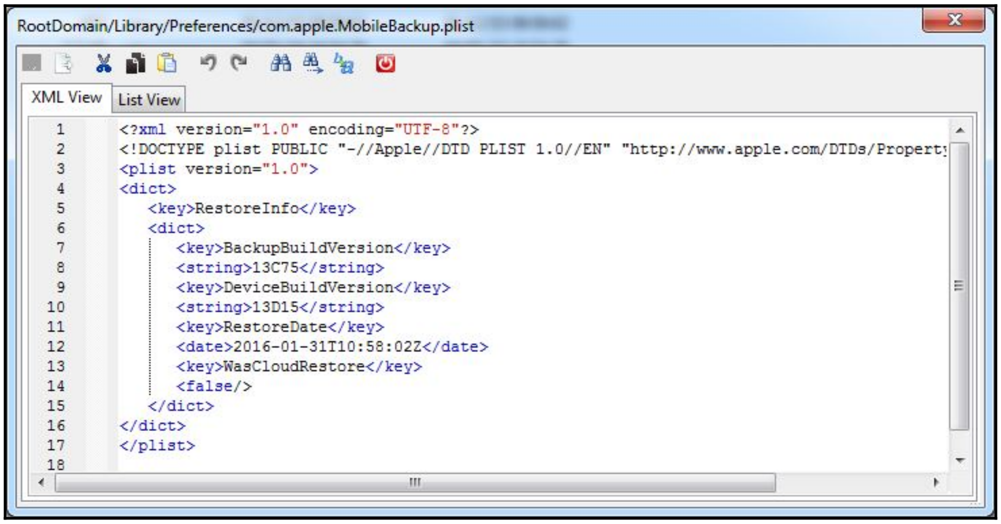
- Domain: `RootDomain/Library/Preferences`
- Filename: `com.apple.preferences.network.plist`

- Description: Airplane mode enabled or disabled at the time of backup.


RootDomain/Library/Preferences/com.apple.preferences.network.plist
XML View
List View

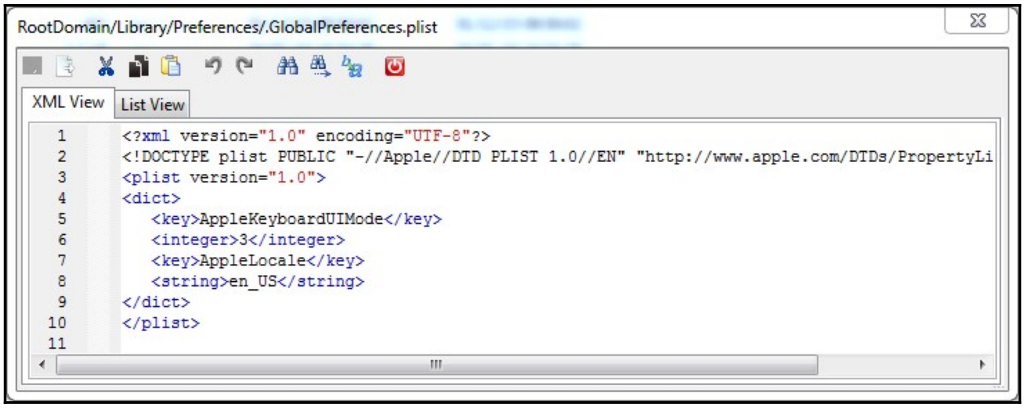
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyLi
<plist version="1.0">
<dict>
   <key>AirplaneModeEnabled</key>
   <false/>
</dict>
</plist>
```

- Domain: `RootDomain/Library/Preferences`
- Filename: `com.apple.MobileBackup.plist`
- Description: Information about the last device reset. In particular, it contains the backup version of iOS, the iOS version installed on the device at the time of recovery, the recovery date, and whether the backup was restored from iCloud.



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/Propert
<plist version="1.0">
<dict>
   <key>RestoreInfo</key>
   <dict>
      <key>BackupBuildVersion</key>
      <string>13C75</string>
      <key>DeviceBuildVersion</key>
      <string>13D15</string>
      <key>RestoreDate</key>
      <date>2016-01-31T10:58:02Z</date>
      <key>WasCloudRestore</key>
      <false/>
   </dict>
</dict>
</plist>
```

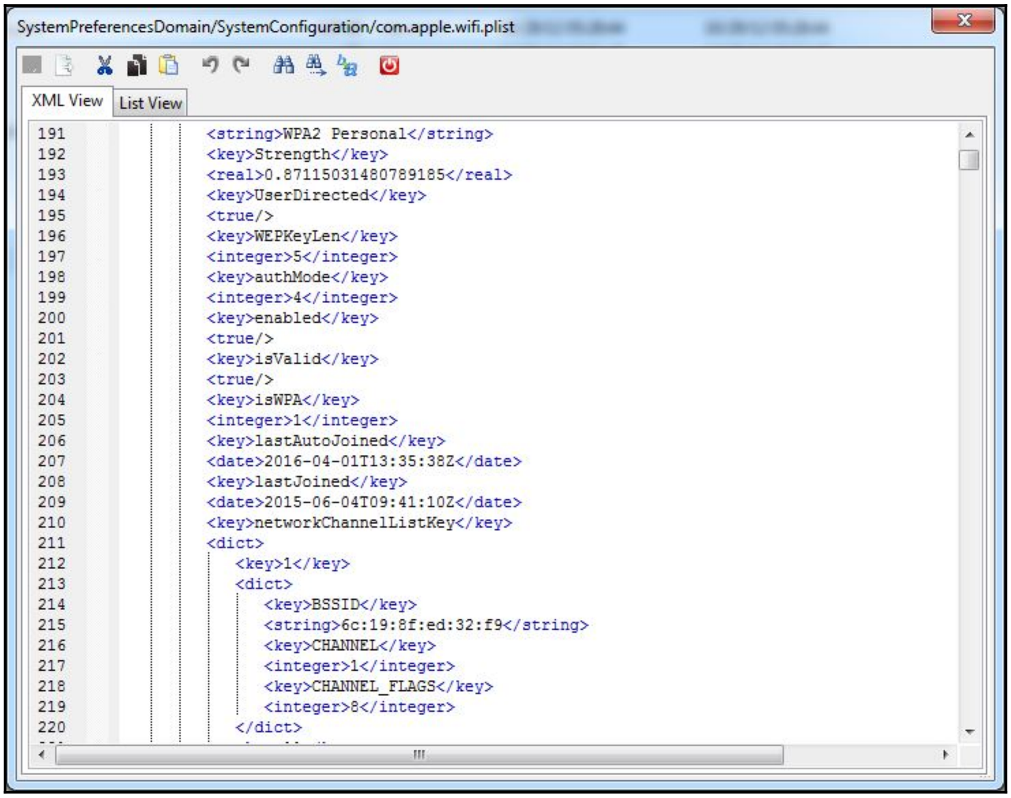
- Domain: `RootDomain/Library/Preferences`
- Filename: `GlobalPreferences`
- Description: Language set.


RootDomain/Library/Preferences/.GlobalPreferences.plist

XML View | List View

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyLi
<plist version="1.0">
<dict>
   <key>AppleKeyboardUIMode</key>
   <integer>3</integer>
   <key>AppleLocale</key>
   <string>en_US</string>
</dict>
</plist>
```

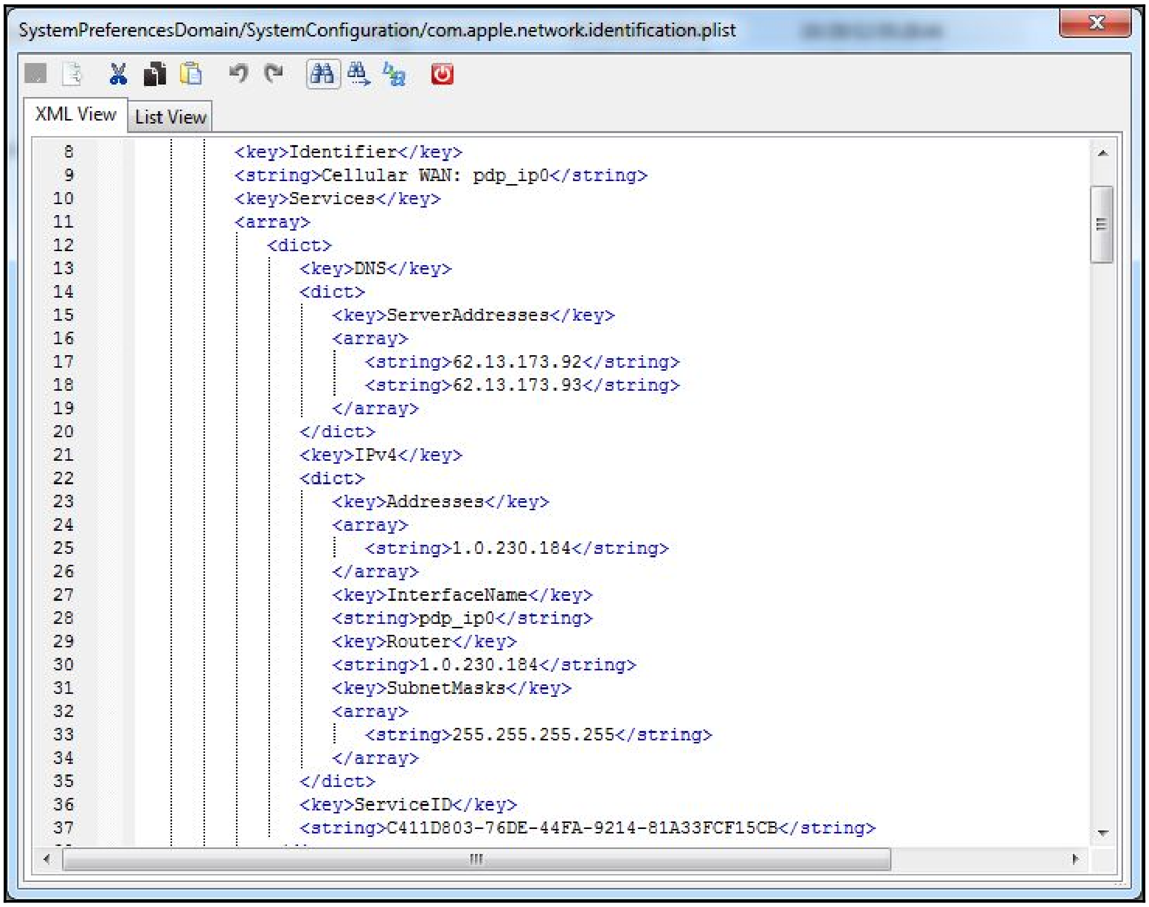
- Domain: `SystemPreferencesDomain/SystemConfiguration/`
- Filename: `com.apple.wifi.plist - preferences.plist`
- Description: Information about the Wi-Fi networks configured in the device.


SystemPreferencesDomain/SystemConfiguration/com.apple.wifi.plist

XML View | List View

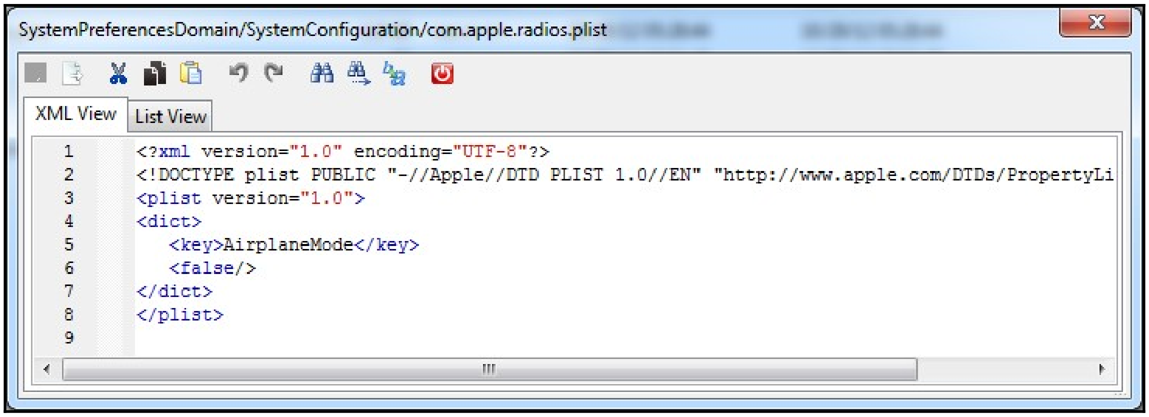
```
          <string>WPA2 Personal</string>
          <key>Strength</key>
          <real>0.87115031480789185</real>
          <key>UserDirected</key>
          <true/>
          <key>WEPKeyLen</key>
          <integer>5</integer>
          <key>authMode</key>
          <integer>4</integer>
          <key>enabled</key>
          <true/>
          <key>isValid</key>
          <true/>
          <key>isWPA</key>
          <integer>1</integer>
          <key>lastAutoJoined</key>
          <date>2016-04-01T13:35:38Z</date>
          <key>lastJoined</key>
          <date>2015-06-04T09:41:10Z</date>
          <key>networkChannelListKey</key>
          <dict>
             <key>1</key>
             <dict>
                <key>BSSID</key>
                <string>6c:19:8f:ed:32:f9</string>
                <key>CHANNEL</key>
                <integer>1</integer>
                <key>CHANNEL_FLAGS</key>
                <integer>8</integer>
             </dict>
```

- Domain: `SystemPreferencesDomain/SystemConfiguration/`
- Filename: `com.apple.network.identification.plist`
- Description: TCP/IP settings for latest connections (both Wi-Fi and cellular networks).


SystemPreferencesDomain/SystemConfiguration/com.apple.network.identification.plist
XML View
List View

```
<key>Identifier</key>
<string>Cellular WAN: pdp_ip0</string>
<key>Services</key>
<array>
    <dict>
        <key>DNS</key>
        <dict>
            <key>ServerAddresses</key>
            <array>
                <string>62.13.173.92</string>
                <string>62.13.173.93</string>
            </array>
        </dict>
        <key>IPv4</key>
        <dict>
            <key>Addresses</key>
            <array>
                <string>1.0.230.184</string>
            </array>
            <key>InterfaceName</key>
            <string>pdp_ip0</string>
            <key>Router</key>
            <string>1.0.230.184</string>
            <key>SubnetMasks</key>
            <array>
                <string>255.255.255.255</string>
            </array>
        </dict>
        <key>ServiceID</key>
        <string>C411D803-76DE-44FA-9214-81A33FCF15CB</string>
```

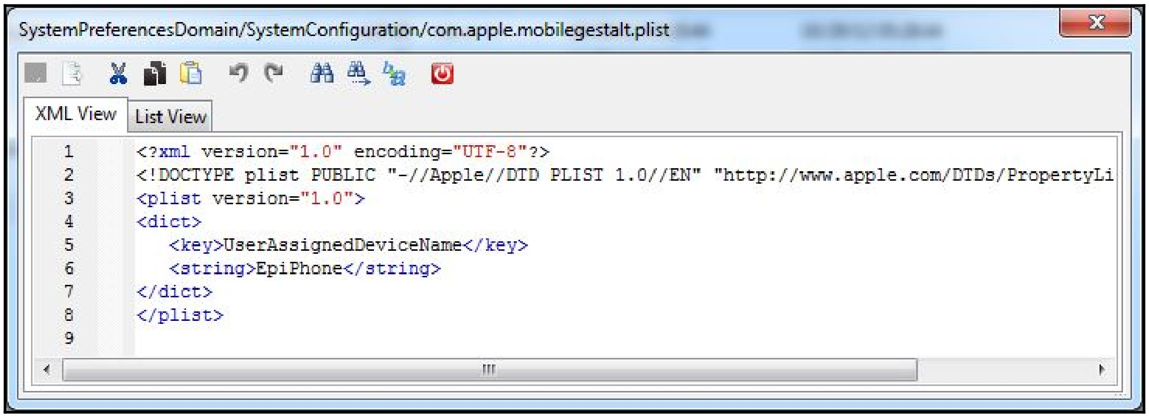
- Domain: `SystemPreferencesDomain/SystemConfiguration/`
- Filename: `com.apple.radios.plist`
- Description: Airplane mode enabled or disabled at the time of backup.



SystemPreferencesDomain/SystemConfiguration/com.apple.radios.plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyLi
<plist version="1.0">
<dict>
   <key>AirplaneMode</key>
   <false/>
</dict>
</plist>
```

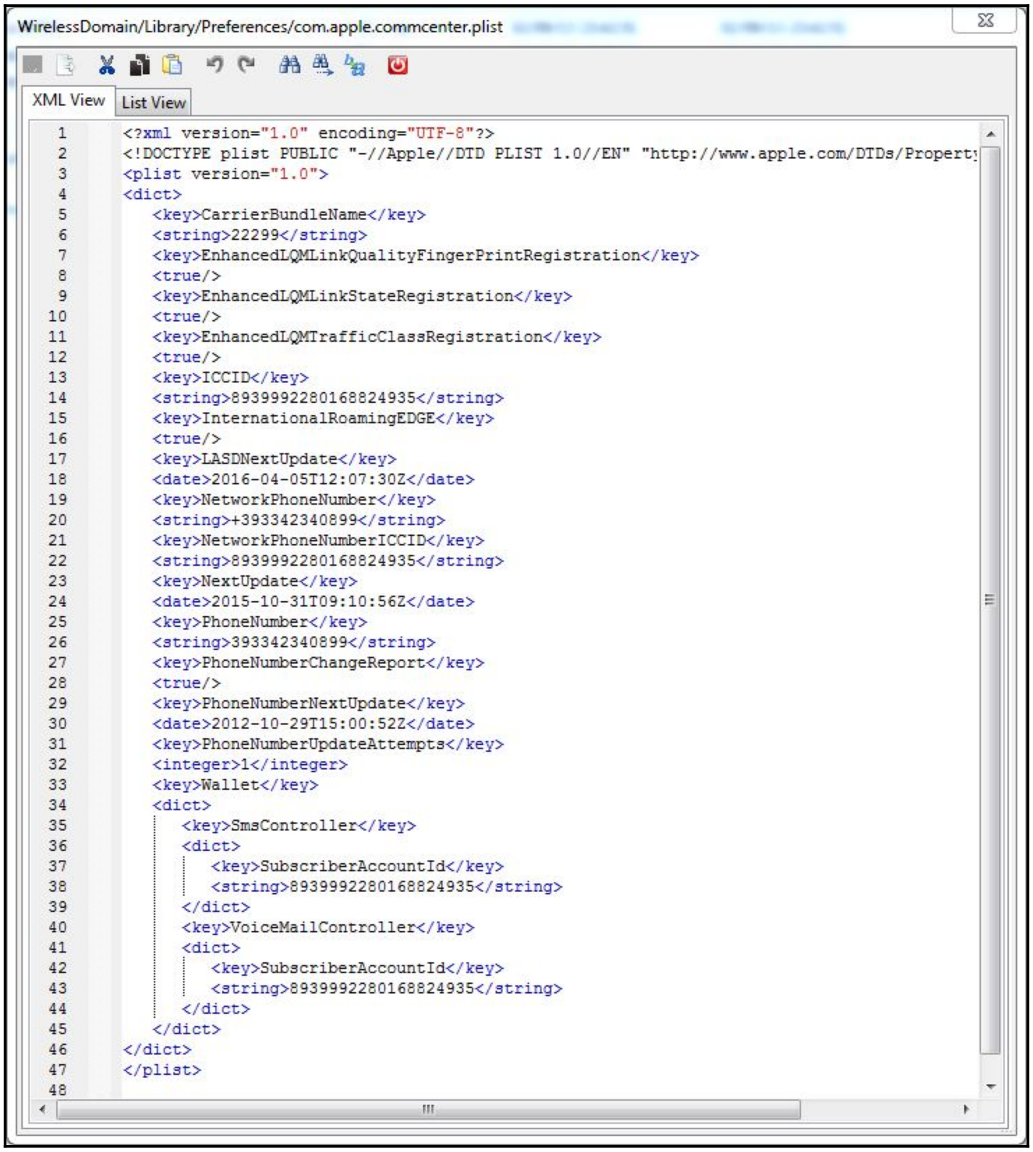
- Domain: `SystemPreferencesDomain/SystemConfiguration/`
- Filename: `com.apple.mobilegestalt.plist`
- Description: Device name.



SystemPreferencesDomain/SystemConfiguration/com.apple.mobilegestalt.plist

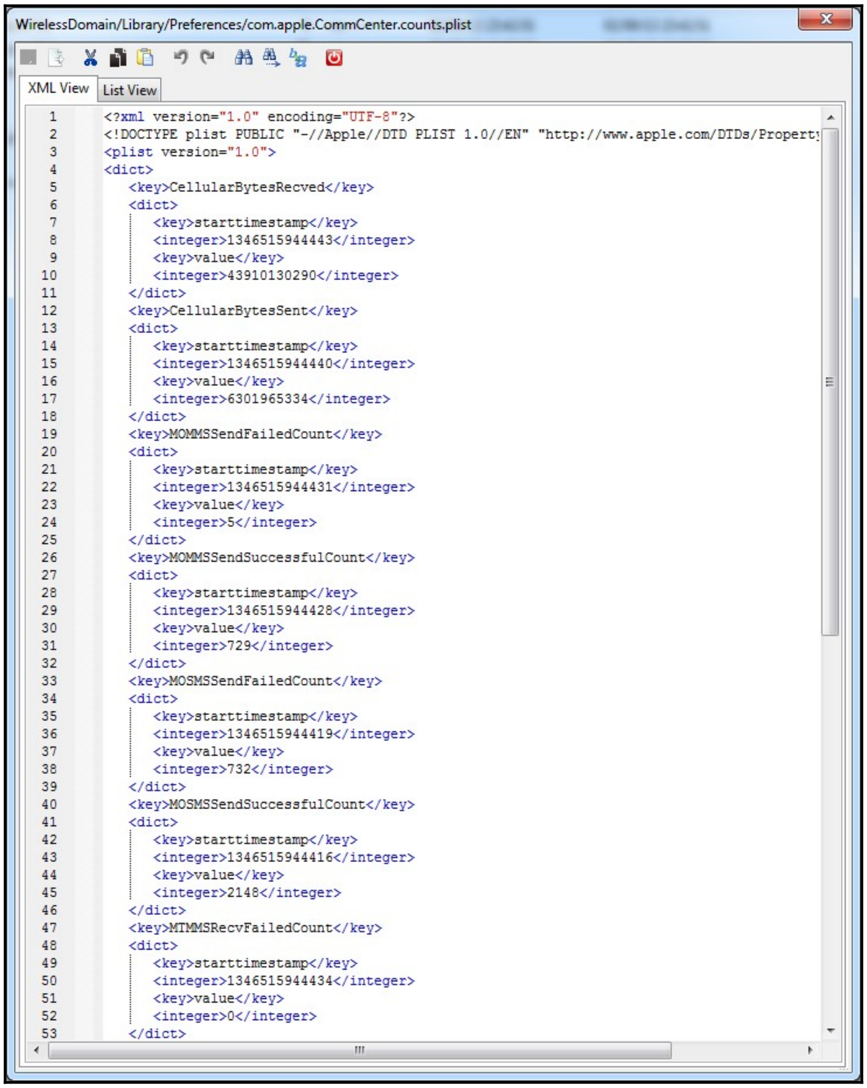
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyLi
<plist version="1.0">
<dict>
   <key>UserAssignedDeviceName</key>
   <string>EpiPhone</string>
</dict>
</plist>
```

- Domain: `WirelessDomain/Library/Preferences/`
- Filename: `com.apple.commcenter.plist`
- Description: Telco provider in use and information about the SIM card.


WirelessDomain/Library/Preferences/com.apple.commcenter.plist
XML View
List View

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/Propert
<plist version="1.0">
<dict>
   <key>CarrierBundleName</key>
   <string>22299</string>
   <key>EnhancedLQMLinkQualityFingerPrintRegistration</key>
   <true/>
   <key>EnhancedLQMLinkStateRegistration</key>
   <true/>
   <key>EnhancedLQMTrafficClassRegistration</key>
   <true/>
   <key>ICCID</key>
   <string>8939992280168824935</string>
   <key>InternationalRoamingEDGE</key>
   <true/>
   <key>LASDNextUpdate</key>
   <date>2016-04-05T12:07:30Z</date>
   <key>NetworkPhoneNumber</key>
   <string>+393342340899</string>
   <key>NetworkPhoneNumberICCID</key>
   <string>8939992280168824935</string>
   <key>NextUpdate</key>
   <date>2015-10-31T09:10:56Z</date>
   <key>PhoneNumber</key>
   <string>393342340899</string>
   <key>PhoneNumberChangeReport</key>
   <true/>
   <key>PhoneNumberNextUpdate</key>
   <date>2012-10-29T15:00:52Z</date>
   <key>PhoneNumberUpdateAttempts</key>
   <integer>1</integer>
   <key>Wallet</key>
   <dict>
      <key>SmsController</key>
      <dict>
         <key>SubscriberAccountId</key>
         <string>8939992280168824935</string>
      </dict>
      <key>VoiceMailController</key>
      <dict>
         <key>SubscriberAccountId</key>
         <string>8939992280168824935</string>
      </dict>
   </dict>
</dict>
</plist>
```

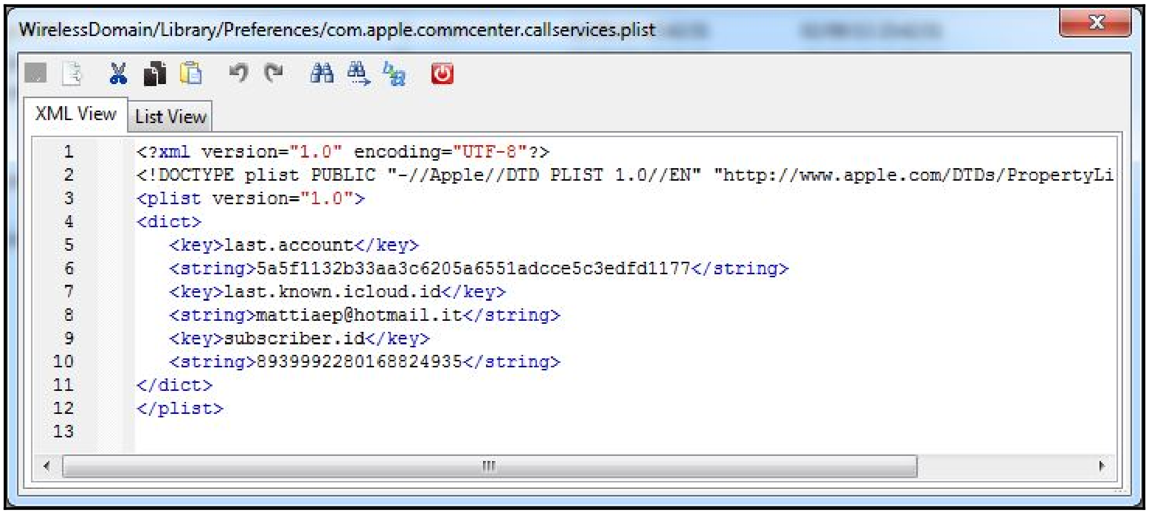
- Domain: `WirelessDomain/Library/Preferences/`
- Filename: `com.apple.commcenter.counts.plist`
- Description: Statistics on the use of data and cellular network (for example, bytes received/sent, SMS received/sent, and so on).



WirelessDomain/Library/Preferences/com.apple.CommCenter.counts.plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/Propert
<plist version="1.0">
<dict>
   <key>CellularBytesRecved</key>
   <dict>
      <key>starttimestamp</key>
      <integer>1346515944443</integer>
      <key>value</key>
      <integer>43910130290</integer>
   </dict>
   <key>CellularBytesSent</key>
   <dict>
      <key>starttimestamp</key>
      <integer>1346515944440</integer>
      <key>value</key>
      <integer>6301965334</integer>
   </dict>
   <key>MOMMSSendFailedCount</key>
   <dict>
      <key>starttimestamp</key>
      <integer>1346515944431</integer>
      <key>value</key>
      <integer>5</integer>
   </dict>
   <key>MOMMSSendSuccessfulCount</key>
   <dict>
      <key>starttimestamp</key>
      <integer>1346515944428</integer>
      <key>value</key>
      <integer>729</integer>
   </dict>
   <key>MOSMSSendFailedCount</key>
   <dict>
      <key>starttimestamp</key>
      <integer>1346515944419</integer>
      <key>value</key>
      <integer>732</integer>
   </dict>
   <key>MOSMSSendSuccessfulCount</key>
   <dict>
      <key>starttimestamp</key>
      <integer>1346515944416</integer>
      <key>value</key>
      <integer>2148</integer>
   </dict>
   <key>MTMMSRecvFailedCount</key>
   <dict>
      <key>starttimestamp</key>
      <integer>1346515944434</integer>
      <key>value</key>
      <integer>0</integer>
   </dict>
```

- Domain: `WirelessDomain/Library/Preferences/`
- Filename: `com.apple.commcenter.callservices.plist`
- Description: iCloud account e-mail address.


WirelessDomain/Library/Preferences/com.apple.commcenter.callservices.plist
XML View
List View

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyLi
<plist version="1.0">
<dict>
   <key>last.account</key>
   <string>5a5f1132b33aa3c6205a6551adcce5c3edfd1177</string>
   <key>last.known.icloud.id</key>
   <string>mattiaep@hotmail.it</string>
   <key>subscriber.id</key>
   <string>8939992280168824935</string>
</dict>
</plist>
```

- Domain: `WirelessDomain/Library/Preferences/`
- Filename: `csidata`
- Description: Device settings for cellular network (for example, cellular data enabled/disabled).

- Domain: `HomeDomain/Library/MobileBluetooth/`
- Filename: `com.apple.MobileBluetooth.ledevices.plist`
- Description: Information about the Bluetooth devices seen by the iOS device.

- Domain: `HomeDomain/Library/TCC`
- Filename: `TCC.db`
- Description: Access privileges assigned to the applications (for example, Address Book, Photos, Facebook, Microphone, Camera, and so on).

- Domain: `HomeDomain/Library/SpringBoard`
- Filename: `IconState.plist`
- Description: Arrangement of the icons within the SpringBoard, divided by window.

- Domain: `HomeDomain/Library/AddressBook/`
- Filename: `AddressBook.sqlitedb`
- Description: User personal contacts, such as names, phone numbers, e-mail addresses, and so on.

- Domain: `HomeDomain/Library/AddressBook/`
- Filename: `AddressBookImage.sqlitedb`
- Description: Images associated with the contacts in the Address Book.

- Domain: `HomeDomain/Library/Calendar/`
- Filename: `Calendar.sqlitedb`
- Description: All the user's calendars and associated events.

- Domain: `HomeDomain/Library/CallHistoryDB/`
- Filename: `CallHistory.storedata`
- Description: The list of outgoing, incoming, and missed calls made through the cellular network, as well as FaceTime.

- Domain: `WirelessDomain/Library/CallHistory/`
- Filename: `Call_history.db`
- Description: File used as Call History database until iOS 7. If the phone has been upgraded or restored, you can find information on old calls.

- Domain: `HomeDomain/Library/Voicemail/`
- Filename: `Voicemail.db`
- Description: Voicemail database.

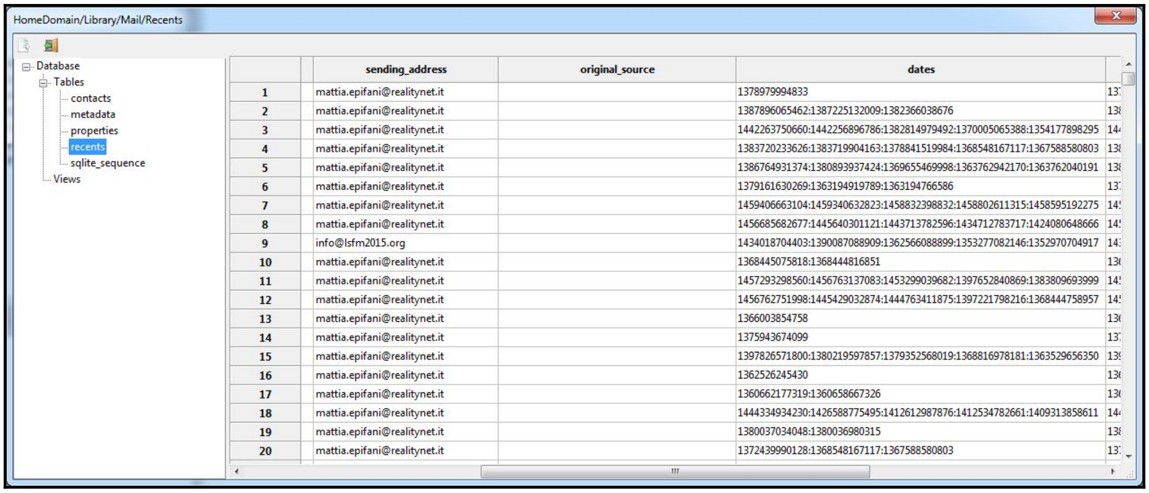
- Domain: `HomeDomain/Library/SMS/`
- Filename: `sms.db`
- Description: SMS and iMessages sent and received, including the date it has been received, read, and delivered (for iMessage), the text, and the type (sent/received).

- Domain: `HomeDomain/Library/SMS/`
- Folder name: `Drafts`
- Description: Drafts of both SMS and iMessage.

- Domain: `MediaDomain/Library/SMS/`
- Folder name: `Attachments`
- Description: Attachments received via MMS.

- Domain: `HomeDomain/Library/`
- Folder name: `DataAccess`
- Description: It contains a subfolder for each mailbox configured within the Mail application. Inside each subfolder, there is a file named `mboxCache.plist` containing the structure of the mailbox folder.

- Domain: `HomeDomain/Library/Mail`
- Filename: `Recents`
- Description: Information about the receivers and sent date of e-mails recently sent.



		sending_address	original_source	dates	
1		mattia.epifani@realitynet.it		1378979994833	13
2		mattia.epifani@realitynet.it		1387896065462:1387225132009:1382366038676	13
3		mattia.epifani@realitynet.it		1442263750660:1442256896786:1382814979492:1370005065388:1354177898295	14
4		mattia.epifani@realitynet.it		1383720233626:1383719904163:1378841519984:1368548167117:1367588580803	13
5		mattia.epifani@realitynet.it		1386764931374:1380893937424:1369655469998:1363762942170:1363762040191	13
6		mattia.epifani@realitynet.it		1379161630269:1363194919789:1363194766586	13
7		mattia.epifani@realitynet.it		1459406663104:1459340632823:1458832398832:1458802611315:1458595192275	14
8		mattia.epifani@realitynet.it		1456685682677:1445640301121:1443713782596:1434712783717:1424080648666	14
9		info@lsfm2015.org		1434018704403:1390087088909:1362566088899:1353277082146:1352970704917	14
10		mattia.epifani@realitynet.it		1368445075818:1368444816851	13
11		mattia.epifani@realitynet.it		1457293298560:1456763137083:1453299039682:1397652840869:1383809693999	14
12		mattia.epifani@realitynet.it		1456762751998:1445429032874:1444763411875:1397221798216:1368444758957	14
13		mattia.epifani@realitynet.it		1366003854758	13
14		mattia.epifani@realitynet.it		1375943674099	13
15		mattia.epifani@realitynet.it		1397826571800:1380219597857:1379352568019:1368816978181:1363529656350	13
16		mattia.epifani@realitynet.it		1362526245430	13
17		mattia.epifani@realitynet.it		1360662177319:1360658667326	13
18		mattia.epifani@realitynet.it		1444334934230:1426588775495:1412612987876:1412534782661:1409313858611	14
19		mattia.epifani@realitynet.it		1380037034048:1380036980315	13
20		mattia.epifani@realitynet.it		1372439990128:1368548167117:1367588580803	13

- Domain: `AppDomain/com.apple.mobilemail/Preferences/`
- Filename: `com.apple.mobilemail.plist`
- `com.apple.MailAccount-ExtProperties.plist`
- Description: Apple Mail application configuration and settings of the configured accounts.

- Domain: `HomeDomain/Library/Notes`
- Filename: `Notes.sqlite`
- Description: Saved notes, grouped by account. Starting from iOS 9.3, users can optionally encrypt every single note with a password.

- Domain: `HomeDomain/Library/Safari/`
- Filename: `bookmarks.db`
- Description: Safari bookmarks.

- Domain: `AppDomain/com.apple.mobilesafari/Library/Preferences/`
- Filename: `com.apple.mobilesafari.it`
- Description: Safari configuration file.

- Domain: `AppDomain/com.apple.mobilesafari/Library/Safari/`
- Filename: `History.db`
- Description: Safari navigation history.

- Domain: `AppDomain/com.apple.mobilesafari/Library/Safari/`
- Filename: `SuspendedState.plist`
- Description: It contains the status of all currently active tabs in Safari.

- Domain: `AppDomain/com.apple.mobilesafari/Library/Safari/`
- Folder name: `Thumbnails`
- Description: It contains the thumbnails in the PNG format of the currently active Safari pages.

- Domain: `AppDomain/com.apple.mobilesafari/Library/WebKit/WebSiteData`
- Folder name: `LocalStorage`
- Description: The `LocalStorage` database stored by websites. May contain specific information stored by the site and are useful to determine the URL accessed even though it's no longer present in the history.

- Domain: `AppDomain/com.apple.Maps/Library/Preferences`
- Filename: `com.apple.Maps.plist`
- Description: Apple Maps application configuration. It contains a list of recently searched addresses.

- Domain: `AppDomain/com.apple.Maps/Library/`
- Folder name: `Maps`
- Description: It contains several files related to the usage of Apple Maps application. Files are binary type and with the `.mapsdata` extension (for example, `History.mapsdata`), but you can pull strings with the addresses of places.

- Domain: `CameraRollDomain/Media/`
- Folder name: `DCIM`
- Description: It contains videos and pictures taken from the device camera or saved from other third-party applications (for example, WhatsApp, Facebook, and so on). The pictures taken with the device camera are stored in the JPG format while the videos are in the MOV format. Moreover, it contains several subfolders where the files are actually stored, these names contain an increasing number (that is 100APPLE, 101APPLE, 102APPLE, and so on), and each of these folders may contain up to 1000 files.

- Domain: `CameraRollDomain/Media/DCIM`
- Folder name: `.THMB`
- Description: Contains picture thumbnails in the JPG format.

- Domain: `CameraRollDomain/Media/PhotoData/MISC/`
- Filename: `DCIM_APPLE.plist`
- Description: Information about the active folder and the number of files.

- Domain: `CameraRollDomain/Media/PhotoData/`
- Folder name: `Thumbnails`
- Description: Four files in the proprietary ITHMB format, one for each possible thumbnail size.

- Domain: `CameraRollDomain/Media/PhotoData/Thumbnails/V2/`
- Folder name: `DCIM`
- Description: Contains a subfolder for each picture with thumbnail in the JPG format.

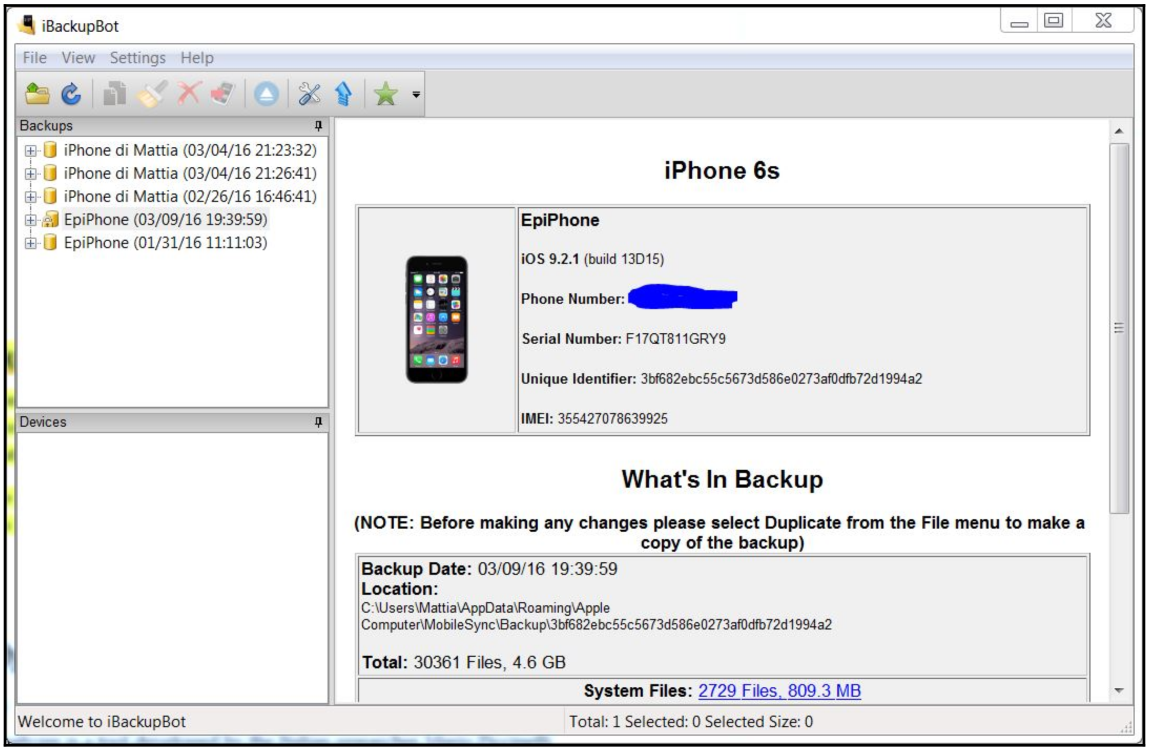
iTunes backup data extraction

There are several tools available to extract data from an iTunes backup-some open source software, as well as some commercial products. These tools allow you to have complete access to the data in case of an unencrypted backup and partial access in case of an encrypted one (in particular the content of the files will not be visible unless you know the backup password or you have been able to crack it). Among the most interesting and powerful tools for accessing and extracting data from a backup, there is forensic software (UFED Physical Analyzer, Oxygen Forensic® Suite, AccessData MPE+, EnCase, Elcomsoft Phone Viewer, and so on), commercial software for data extraction (iBackup Bot, iPhone Backup Extractor, DiskAid, Wondershare Dr. Fone, and so on), and freeware/open source software for data extraction (iPhone Backup Analyzer and iPhone Analyzer). A detailed list is provided in `Appendix B`, *Tools for iOS Forensics*. Another option is to recover the backup content on your own simply with a hex editor. In this case, we suggest you to read the article available at `http://resources.infosecinstitute.com/ios-5-backups-part-1/`.

Case study – iTunes backup parsing with iBackupBot

iBackupBot is a commercial tool for Windows that allows access to iTunes backup stored on a local computer. The trial version allows in any case loading and extracting information from a backup.

Once the software is executed, it automatically loads all the backups stored in the predefined iTunes Backup folder, but another backup can be opened through the **File** menu. In the left-hand pane, all the loaded backups are listed and all the domains can be browsed, while in the central pane general information about the device is shown (device name, iOS version, phone number, serial number, UDID, IMEI, and so on).



iPhone di Mattia (03/04/16 21:23:32)
iPhone di Mattia (03/04/16 21:26:41)
iPhone di Mattia (02/26/16 16:46:41)
EpiPhone (03/09/16 19:39:59)
EpiPhone (01/31/16 11:11:03)

iPhone 6s

EpiPhone
iOS 9.2.1 (build 13D15)
Phone Number: [redacted]
Serial Number: F17QT811GRY9
Unique Identifier: 3bf682ebc55c5673d586e0273af0dfb72d1994a2
IMEI: 355427078639925

What's In Backup

(NOTE: Before making any changes please select Duplicate from the File menu to make a copy of the backup)

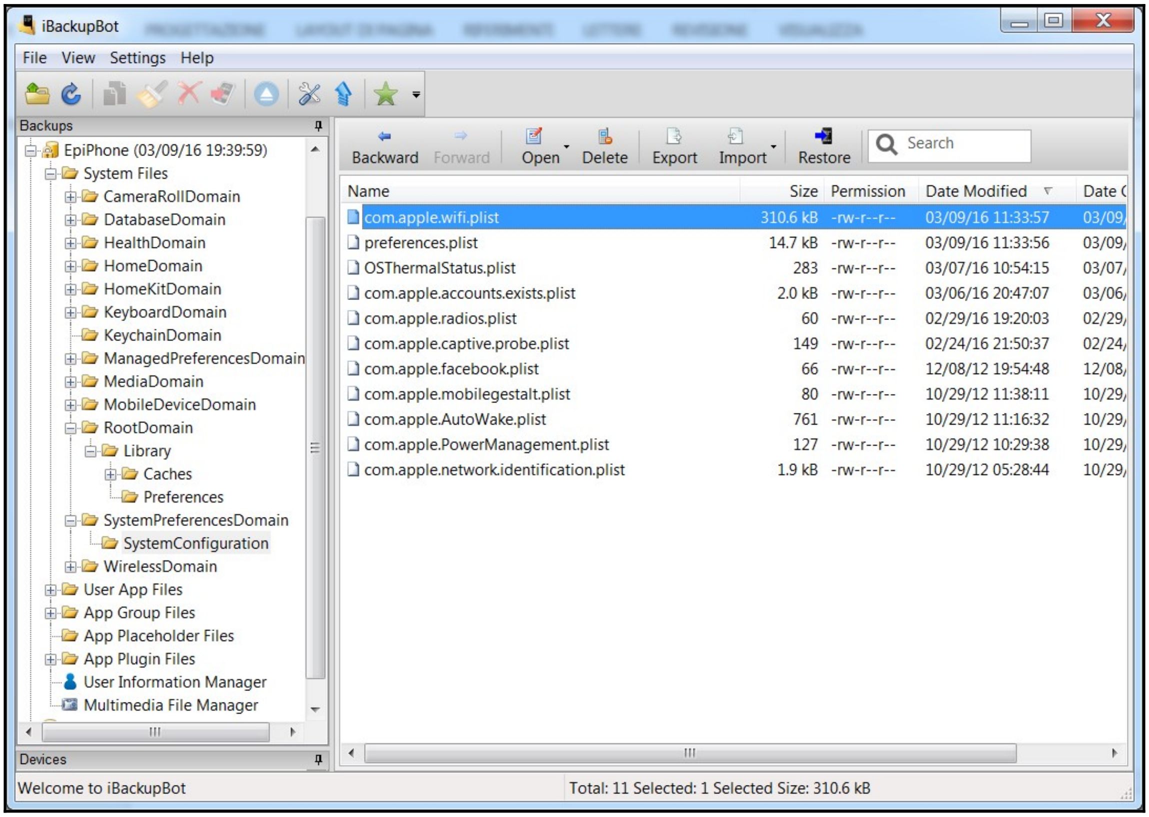
Backup Date: 03/09/16 19:39:59
Location:
C:\Users\Mattia\AppData\Roaming\Apple Computer\MobileSync\Backup\3bf682ebc55c5673d586e0273af0dfb72d1994a2

Total: 30361 Files, 4.6 GB

System Files: 2729 Files, 809.3 MB

Welcome to iBackupBot
Total: 1 Selected: 0 Selected Size: 0

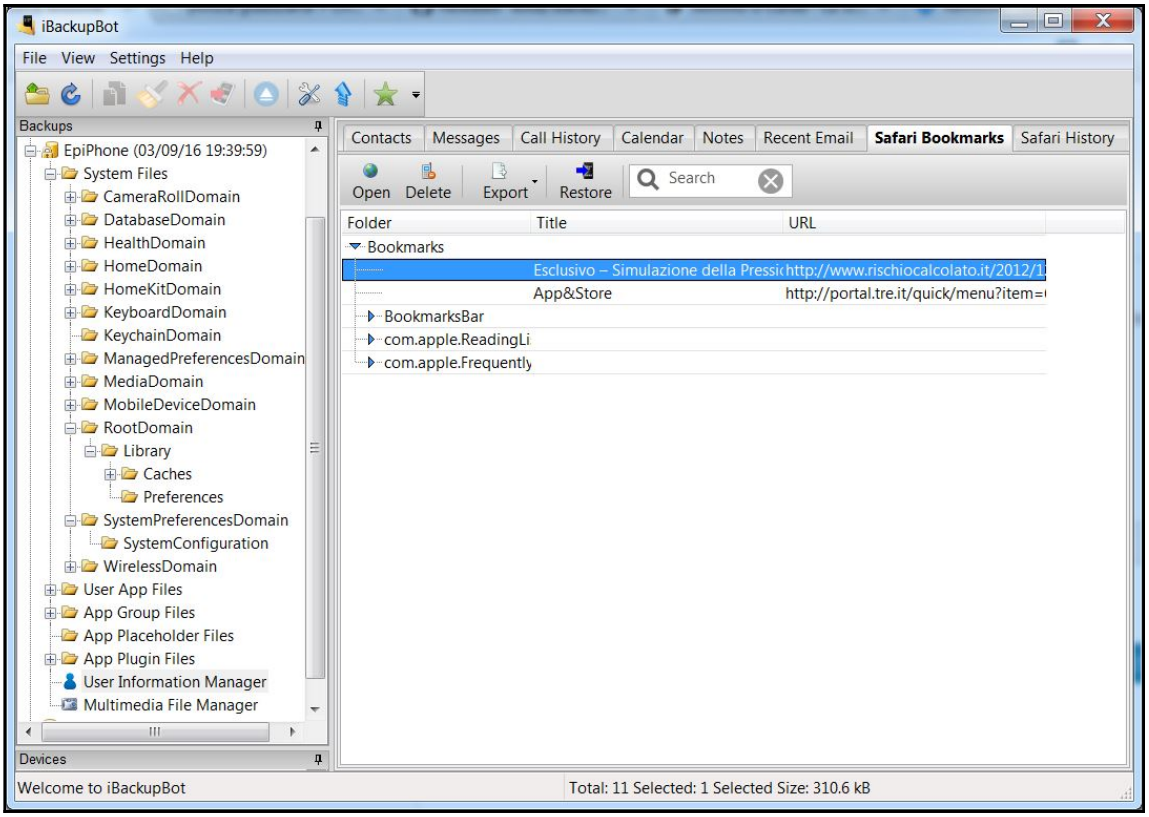
Browsing a backup reveals files that are grouped with respect to their specific domain. **System Files** contains all domains referring to iOS settings and data. Third-party app data is in **User App Files**, **App Group Files**, and **App Plugin Files**. The user can open a file by simply double -clicking on it because the software contains integrated `plist` and SQLite viewers:



Name	Size	Permission	Date Modified	Date (
com.apple.wifi.plist	310.6 kB	-rw-r--r--	03/09/16 11:33:57	03/09/
preferences.plist	14.7 kB	-rw-r--r--	03/09/16 11:33:56	03/09/
OSThermalStatus.plist	283	-rw-r--r--	03/07/16 10:54:15	03/07/
com.apple.accounts.exists.plist	2.0 kB	-rw-r--r--	03/06/16 20:47:07	03/06/
com.apple.radios.plist	60	-rw-r--r--	02/29/16 19:20:03	02/29/
com.apple.captive.probe.plist	149	-rw-r--r--	02/24/16 21:50:37	02/24/
com.apple.facebook.plist	66	-rw-r--r--	12/08/12 19:54:48	12/08/
com.apple.mobilegestalt.plist	80	-rw-r--r--	10/29/12 11:38:11	10/29/
com.apple.AutoWake.plist	761	-rw-r--r--	10/29/12 11:16:32	10/29/
com.apple.PowerManagement.plist	127	-rw-r--r--	10/29/12 10:29:38	10/29/
com.apple.network.identification.plist	1.9 kB	-rw-r--r--	10/29/12 05:28:44	10/29/

Welcome to iBackupBot Total: 11 Selected: 1 Selected Size: 310.6 kB

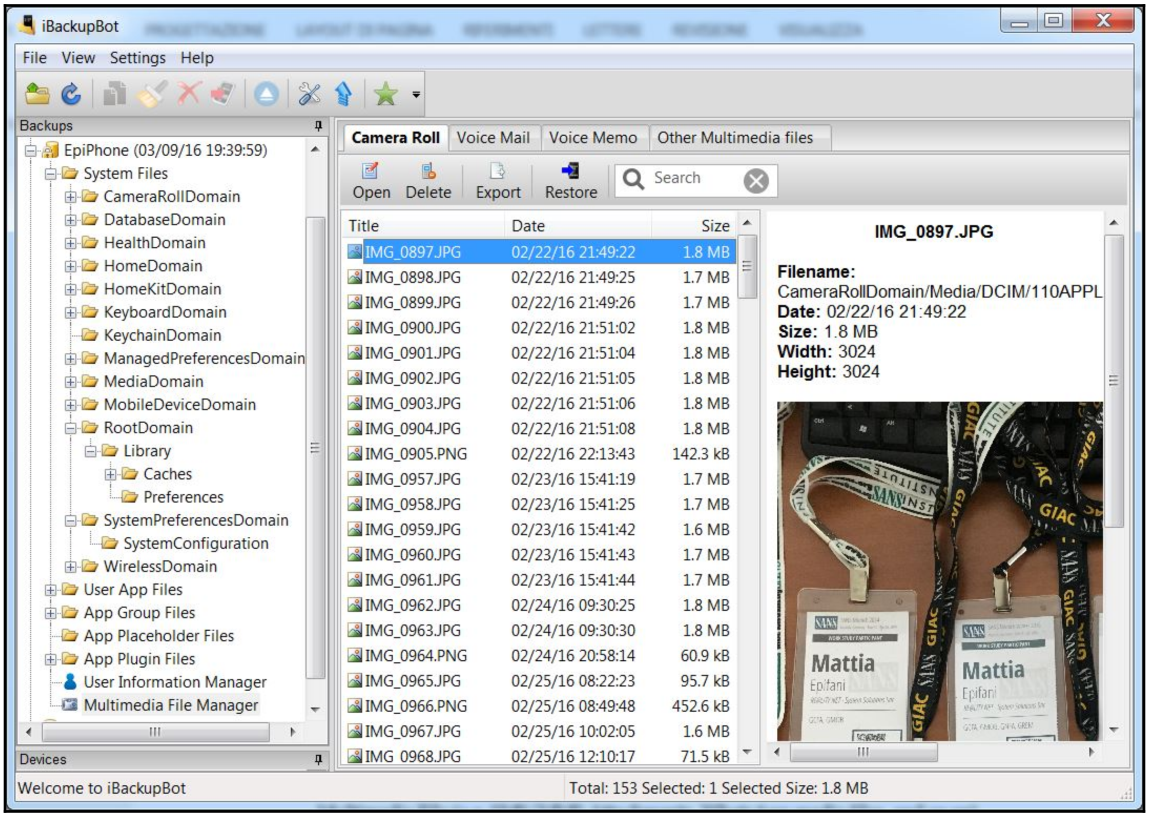
The **User Information Manager** option parses some of the most common data stored in a backup, specifically the following: **Contacts**, SMS/MMS **Messages**, **Call History**, **Calendar**, **Notes**, **Recent Email** (only To: e-mail address, date, and time), **Safari Bookmarks**, and **Safari History**:



Folder	Title	URL
Bookmarks		
	Esclusivo – Simulazione della Pressi	http://www.rischiocalcolato.it/2012/1
	App&Store	http://portal.tre.it/quick/menu?item=
BookmarksBar		
com.apple.ReadingLi		
com.apple.Frequently		

Welcome to iBackupBot Total: 11 Selected: 1 Selected Size: 310.6 kB

Multimedia File Manager provides a view of media files stored in the backup and specifically the **Camera Roll** domain files, **Voice Mail**, **Voice Memo**, and **Other Multimedia files** (for example, SMS/MMS attachments, WhatsApp media files, and so on).



Title	Date	Size
IMG_0897.JPG	02/22/16 21:49:22	1.8 MB
IMG_0898.JPG	02/22/16 21:49:25	1.7 MB
IMG_0899.JPG	02/22/16 21:49:26	1.7 MB
IMG_0900.JPG	02/22/16 21:51:02	1.8 MB
IMG_0901.JPG	02/22/16 21:51:04	1.8 MB
IMG_0902.JPG	02/22/16 21:51:05	1.8 MB
IMG_0903.JPG	02/22/16 21:51:06	1.8 MB
IMG_0904.JPG	02/22/16 21:51:08	1.8 MB
IMG_0905.PNG	02/22/16 22:13:43	142.3 kB
IMG_0957.JPG	02/23/16 15:41:19	1.7 MB
IMG_0958.JPG	02/23/16 15:41:25	1.7 MB
IMG_0959.JPG	02/23/16 15:41:42	1.6 MB
IMG_0960.JPG	02/23/16 15:41:43	1.7 MB
IMG_0961.JPG	02/23/16 15:41:44	1.7 MB
IMG_0962.JPG	02/24/16 09:30:25	1.8 MB
IMG_0963.JPG	02/24/16 09:30:30	1.8 MB
IMG_0964.PNG	02/24/16 20:58:14	60.9 kB
IMG_0965.JPG	02/25/16 08:22:23	95.7 kB
IMG_0966.PNG	02/25/16 08:49:48	452.6 kB
IMG_0967.JPG	02/25/16 10:02:05	1.6 MB
IMG_0968.JPG	02/25/16 12:10:17	71.5 kB

IMG_0897.JPG

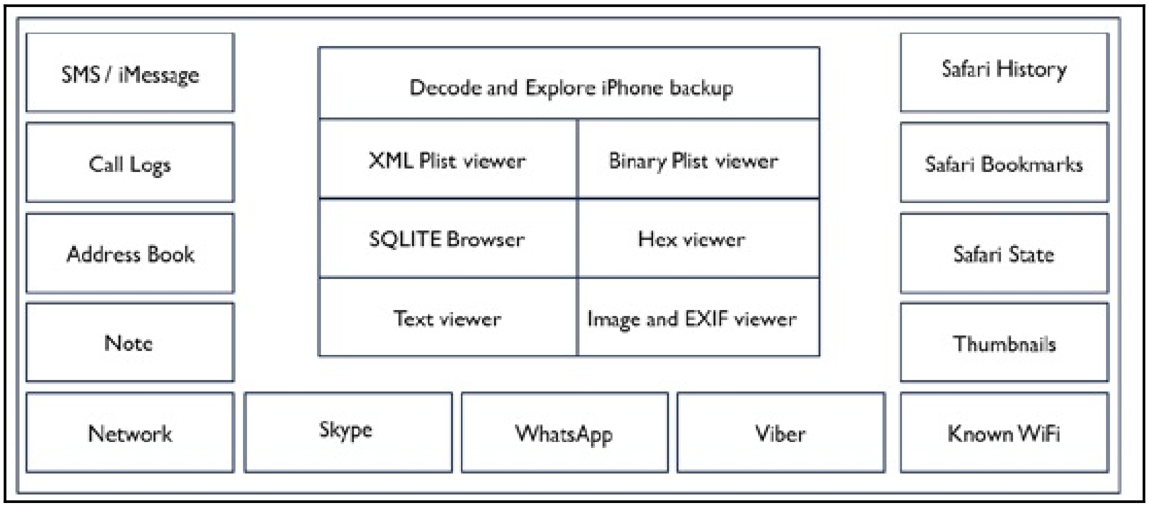
Filename: CameraRollDomain/Media/DCIM/110APPL
Date: 02/22/16 21:49:22
Size: 1.8 MB
Width: 3024
Height: 3024

Welcome to iBackupBot
Total: 153 Selected: 1 Selected Size: 1.8 MB

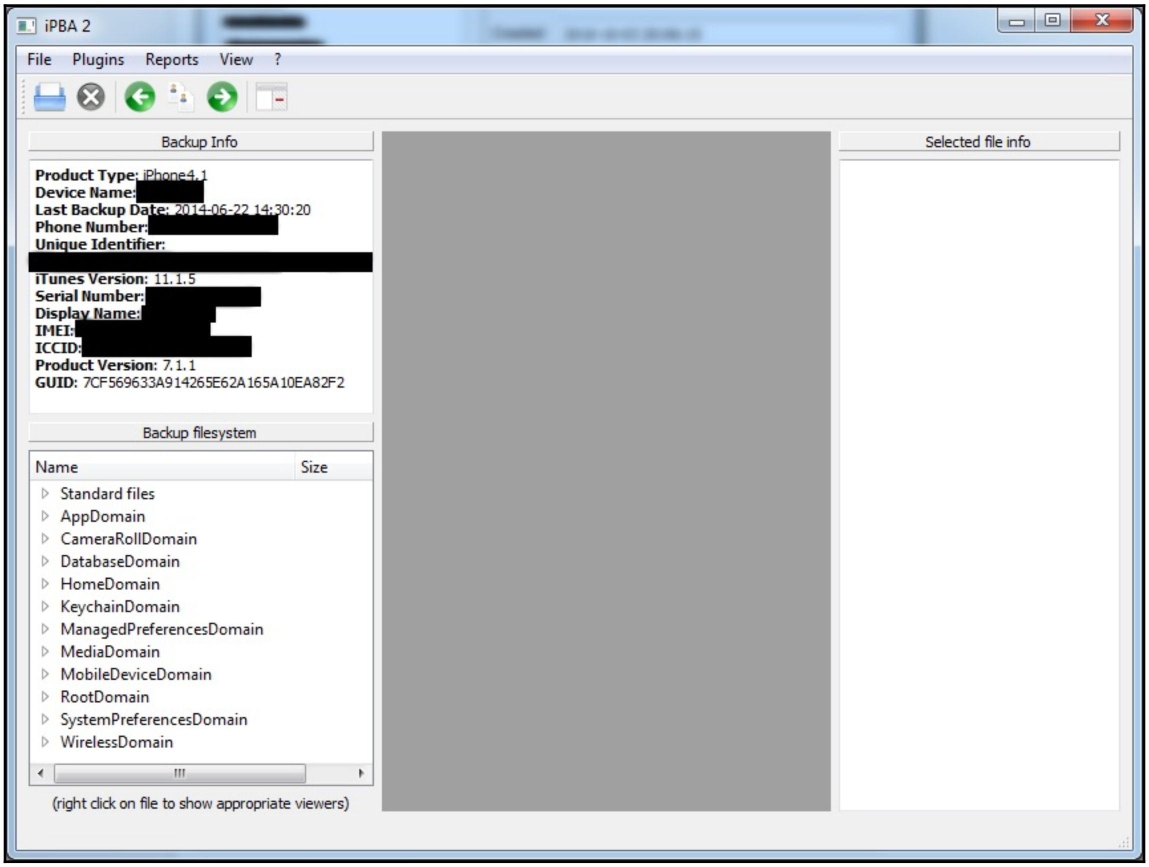
Case study – iTunes backup analysis with iPBA

iPhone Backup Analyzer is a tool developed by the Italian researcher Mario Piccinelli, and it provides a simple way to browse through the backup folder and perform a forensic analysis of an iDevice backup. It was released as open source software under the MIT license, and since it is written in Python, it should be cross-platform (Mac, Linux, and Windows).

The main goal behind the development is to provide a way to analyze the contents of the iPhone backup. It is meant to be used by anyone who wants to easily study what the backup contains, be they a forensics expert, an iOS developer, or just an interested iPhone user. The software is also packed with utilities to easily browse through the content formatted in a ready-to-use way, such as messages, contacts, Safari bookmarks, and so on. Its complete feature set can be summarized in the following diagram:



In a Windows environment, after downloading the tool, you need to unzip it to a folder and launch the executable `iPBA2.exe` file. By navigating to **File** | **Open Archive**, you can choose the folder containing the backup. The software parses and analyzes the backup and provides a graphical way to browse through it:

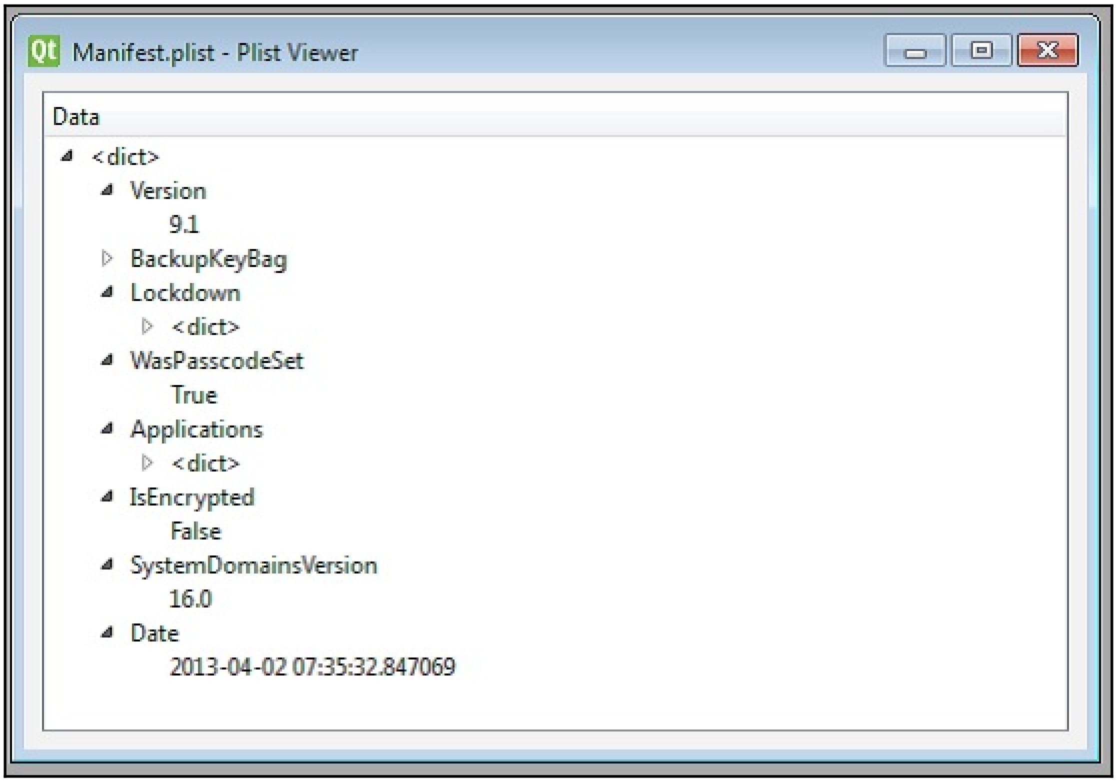


Product Type: iPhone4,1
Device Name: [redacted]
Last Backup Date: 2014-06-22 14:30:20
Phone Number: [redacted]
Unique Identifier: [redacted]
iTunes Version: 11.1.5
Serial Number: [redacted]
Display Name: [redacted]
IMEI: [redacted]
ICCID: [redacted]
Product Version: 7.1.1
GUID: 7CF569633A914265E62A165A10EA82F2

Name	Size
Standard files	
AppDomain	
CameraRollDomain	
DatabaseDomain	
HomeDomain	
KeychainDomain	
ManagedPreferencesDomain	
MediaDomain	
MobileDeviceDomain	
RootDomain	
SystemPreferencesDomain	
WirelessDomain	

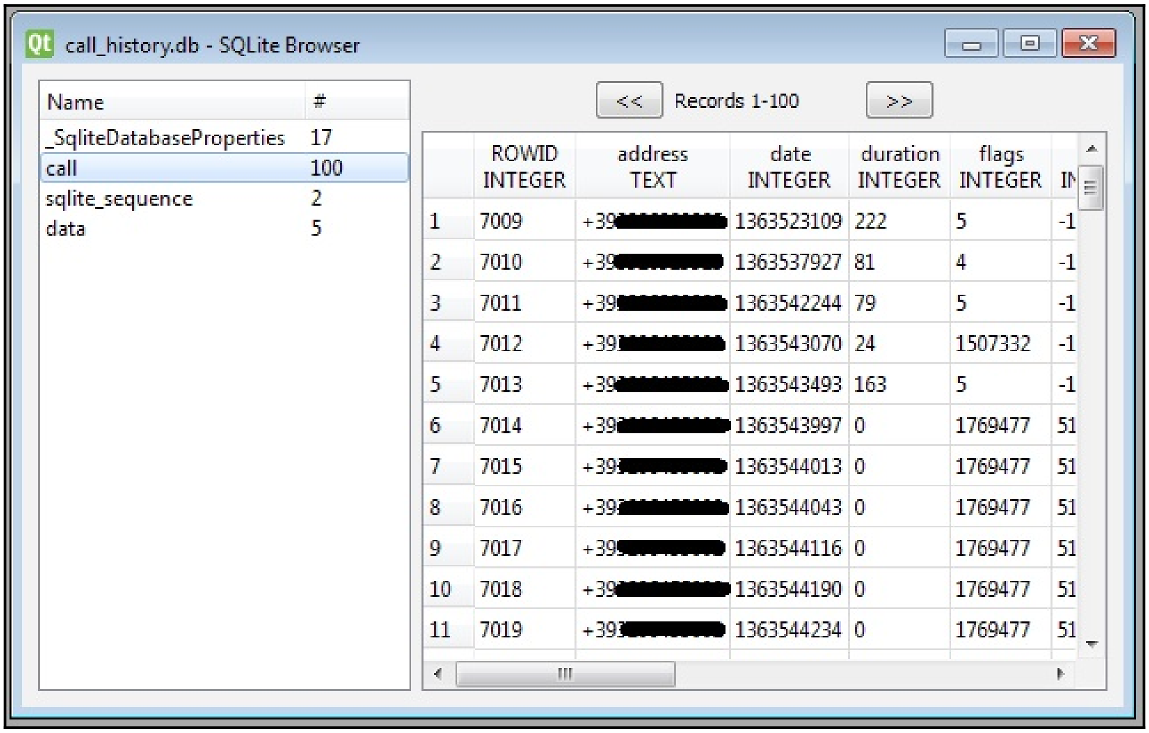
(right click on file to show appropriate viewers)

By right-clicking on a `plist` or SQLite file, the analyst can view the file content. For example, in the following screenshot, you can see the content of the `Manifest.plist` file:



```
<dict>
    Version
        9.1
    BackupKeyBag
    Lockdown
        <dict>
    WasPasscodeSet
        True
    Applications
        <dict>
    IsEncrypted
        False
    SystemDomainsVersion
        16.0
    Date
        2013-04-02 07:35:32.847069
```

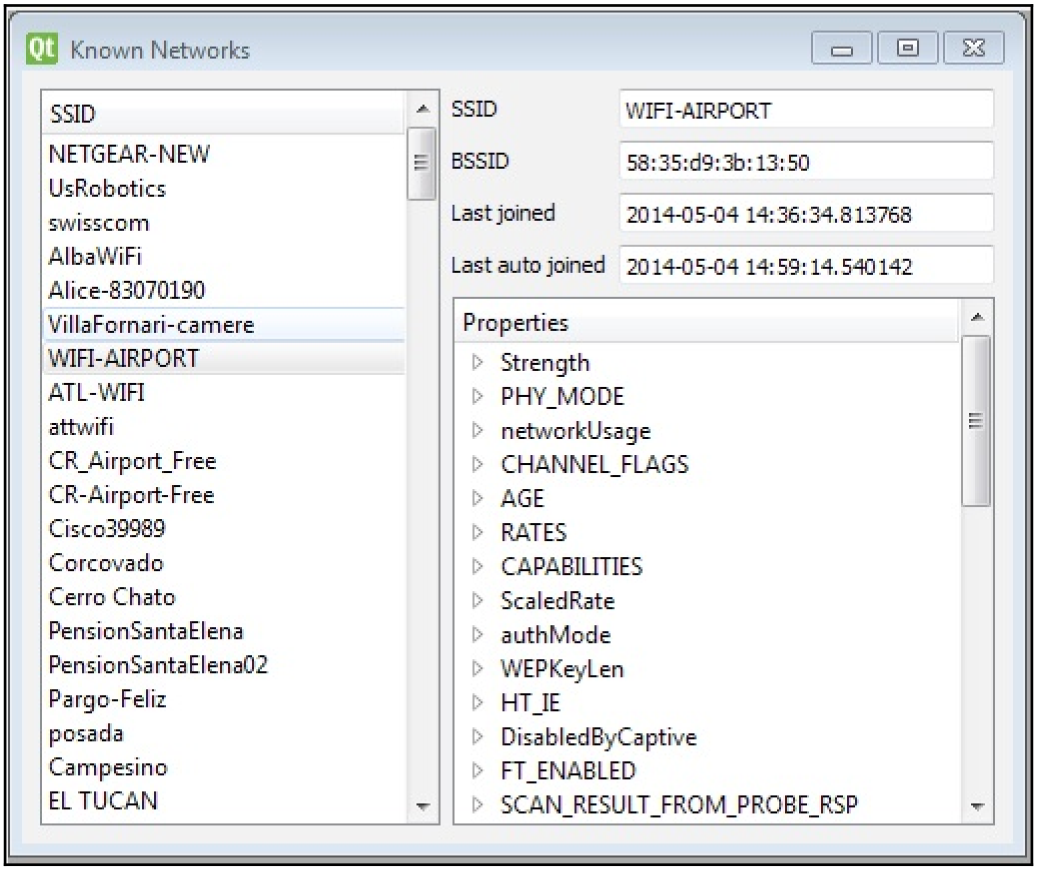
In the following screenshot, you can see the contents of a Call History SQLite database:



Name	#
_SqliteDatabaseProperties	17
call	100
sqlite_sequence	2
data	5

	ROWID INTEGER	address TEXT	date INTEGER	duration INTEGER	flags INTEGER	IN
1	7009	+39[redacted]	1363523109	222	5	-1
2	7010	+39[redacted]	1363537927	81	4	-1
3	7011	+39[redacted]	1363542244	79	5	-1
4	7012	+39[redacted]	1363543070	24	1507332	-1
5	7013	+39[redacted]	1363543493	163	5	-1
6	7014	+39[redacted]	1363543997	0	1769477	51
7	7015	+39[redacted]	1363544013	0	1769477	51
8	7016	+39[redacted]	1363544043	0	1769477	51
9	7017	+39[redacted]	1363544116	0	1769477	51
10	7018	+39[redacted]	1363544190	0	1769477	51
11	7019	+39[redacted]	1363544234	0	1769477	51

By choosing an item from the **Plugins** menu, you can also analyze useful information from the backup. Currently, the software offers 14 plugins: **Address Book Browser**, **Call History**, **Phone Info Browser**, **Known Networks**, **Network Identification**, **Note Browser**, **Safari History Browser**, **Safari State Explorer**, **Safari Bookmarks**, **Skype Browser**, **Messages Browser**, **Thumbnails Browser**, **Viber Browser**, and **WhatsApp Browser**. In the following screenshot, you can see, for example, the known Wi-Fi networks plugin:



SSID
NETGEAR-NEW
UsRobotics
swisscom
AlbaWiFi
Alice-83070190
VillaFornari-camere
WIFI-AIRPORT
ATL-WIFI
attwifi
CR_Airport_Free
CR-Airport-Free
Cisco39989
Corcovado
Cerro Chato
PensionSantaElena
PensionSantaElena02
Pargo-Feliz
posada
Campesino
EL TUCAN

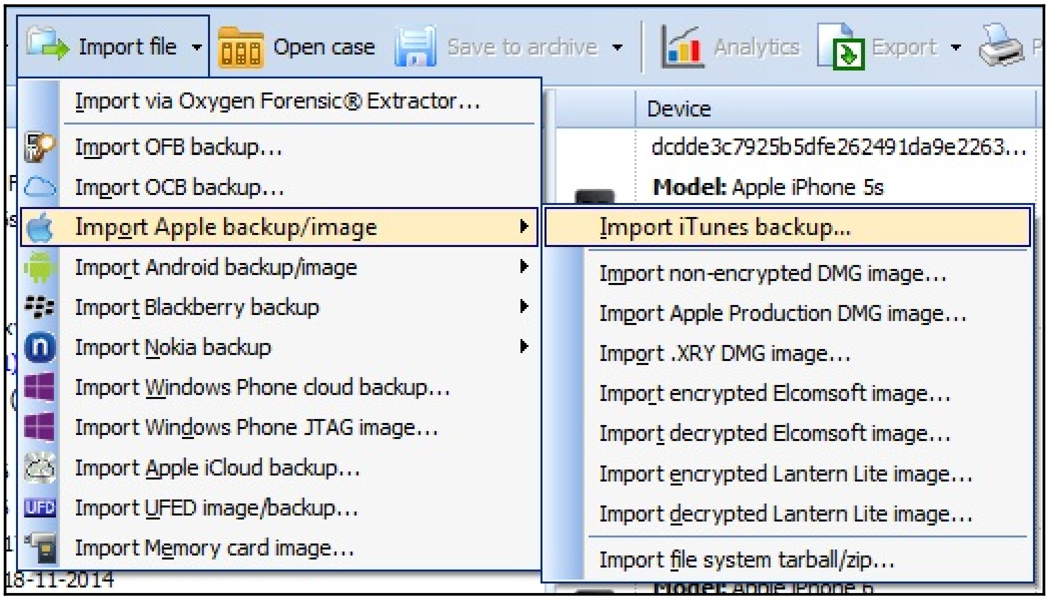
SSID	WIFI-AIRPORT
BSSID	58:35:d9:3b:13:50
Last joined	2014-05-04 14:36:34.813768
Last auto joined	2014-05-04 14:59:14.540142

Properties
Strength
PHY_MODE
networkUsage
CHANNEL_FLAGS
AGE
RATES
CAPABILITIES
ScaledRate
authMode
WEPKeyLen
HT_IE
DisabledByCaptive
FT_ENABLED
SCAN_RESULT_FROM_PROBE_RSP

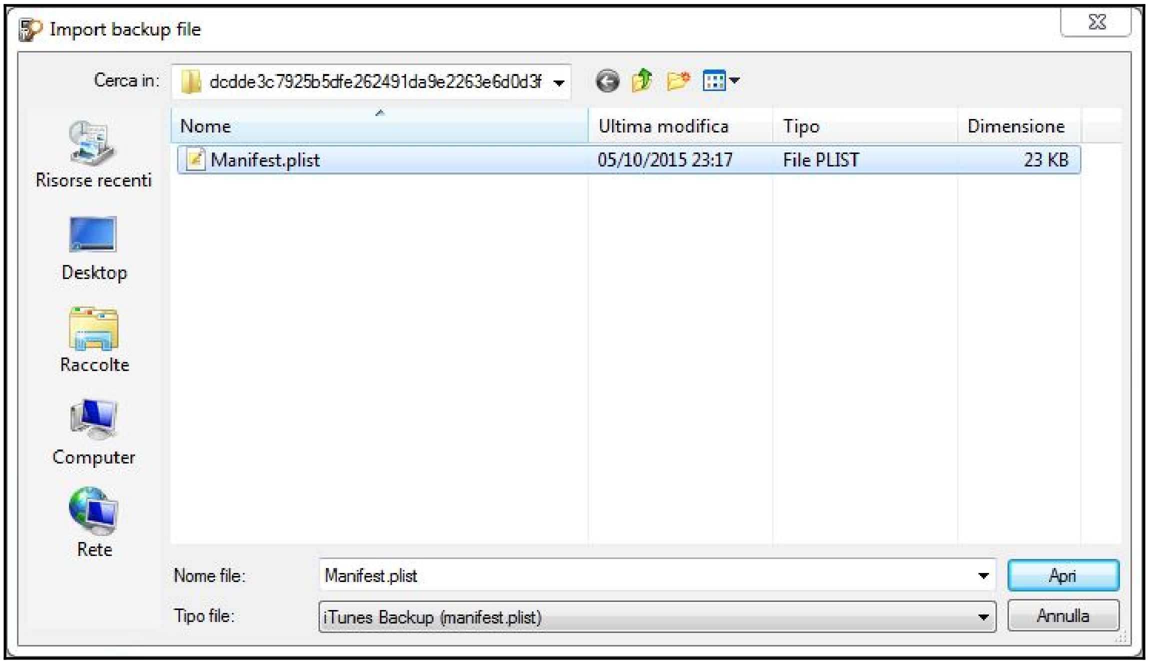
Case study – iTunes backup analysis with Oxygen Forensic Analyst

Oxygen Forensic Analyst is a commercial tool that has already been introduced in `Chapter 3`, *Evidence Acquisition from iDevices*. This tool enables you to acquire data from an iDevice and also to import a previously created iTunes backup.

In order to import a backup, it is enough to click on **Import File** from the main window and navigate to **Import Apple backup/image** | **Import iTunes backup...**:


Import file
Open case
Save to archive
Analytics
Export
Device
dcdde3c7925b5dfe262491da9e2263...
Model: Apple iPhone 5s
Import via Oxygen Forensic® Extractor...
Import OFB backup...
Import OCB backup...
Import Apple backup/image
Import Android backup/image
Import Blackberry backup
Import Nokia backup
Import Windows Phone cloud backup...
Import Windows Phone JTAG image...
Import Apple iCloud backup...
Import UFED image/backup...
Import Memory card image...
Import iTunes backup...
Import non-encrypted DMG image...
Import Apple Production DMG image...
Import .XRY DMG image...
Import encrypted Elcomsoft image...
Import decrypted Elcomsoft image...
Import encrypted Lantern Lite image...
Import decrypted Lantern Lite image...
Import file system tarball/zip...
18-11-2014

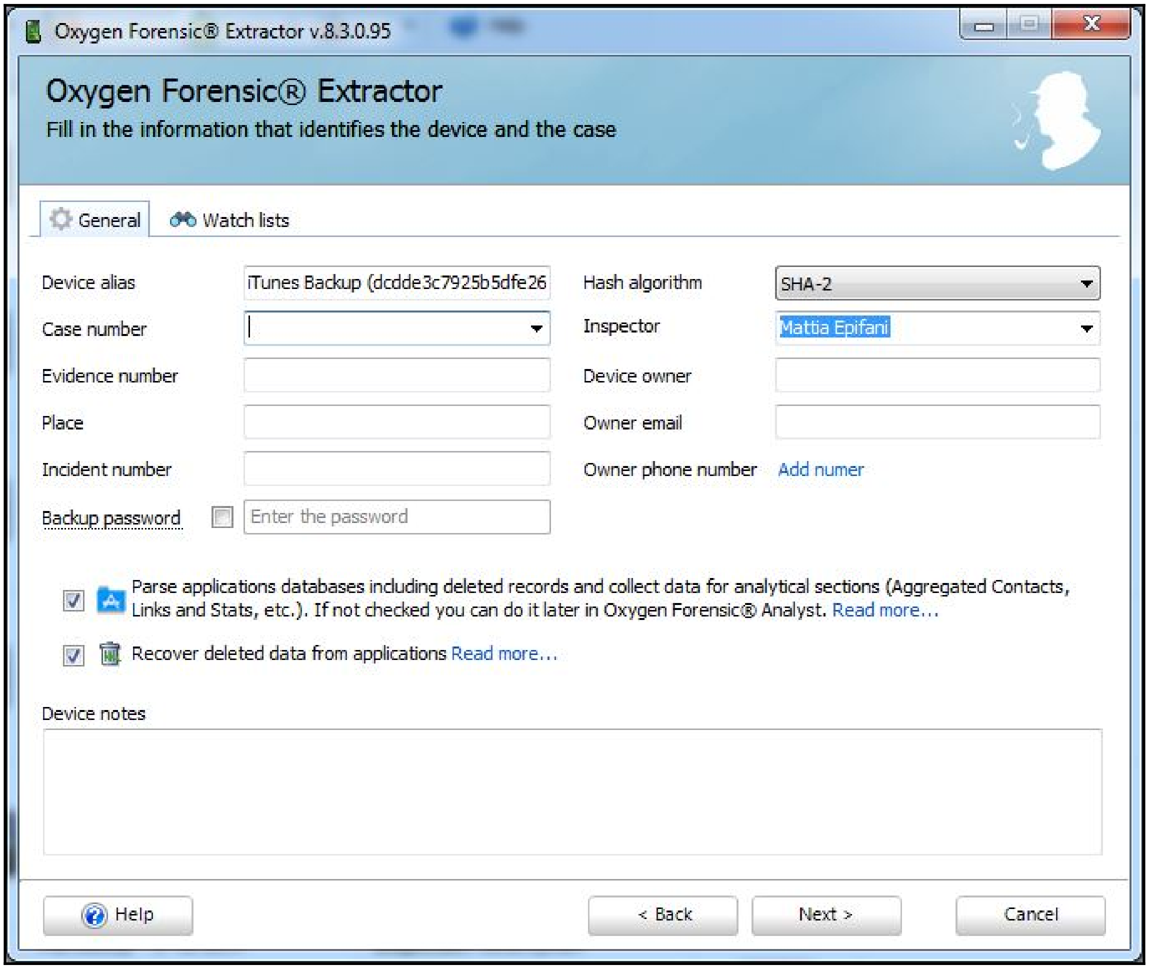
When selecting a folder containing a backup, the tool shows the `Manifest.plist` file, which has to be chosen:



Nome	Ultima modifica	Tipo	Dimensione
Manifest.plist	05/10/2015 23:17	File PLIST	23 KB

Nome file: Manifest.plist
Tipo file: iTunes Backup (manifest.plist)

The software recognizes the backup structure and proposes an options screen where the investigator can decide whether they want to only do the backup conversion, or if they also want the application to do the parsing of applications' databases. Similarly, it is also possible to ask the software to recover deleted records within the database:


Oxygen Forensic® Extractor v.8.3.0.95

Oxygen Forensic® Extractor

Fill in the information that identifies the device and the case

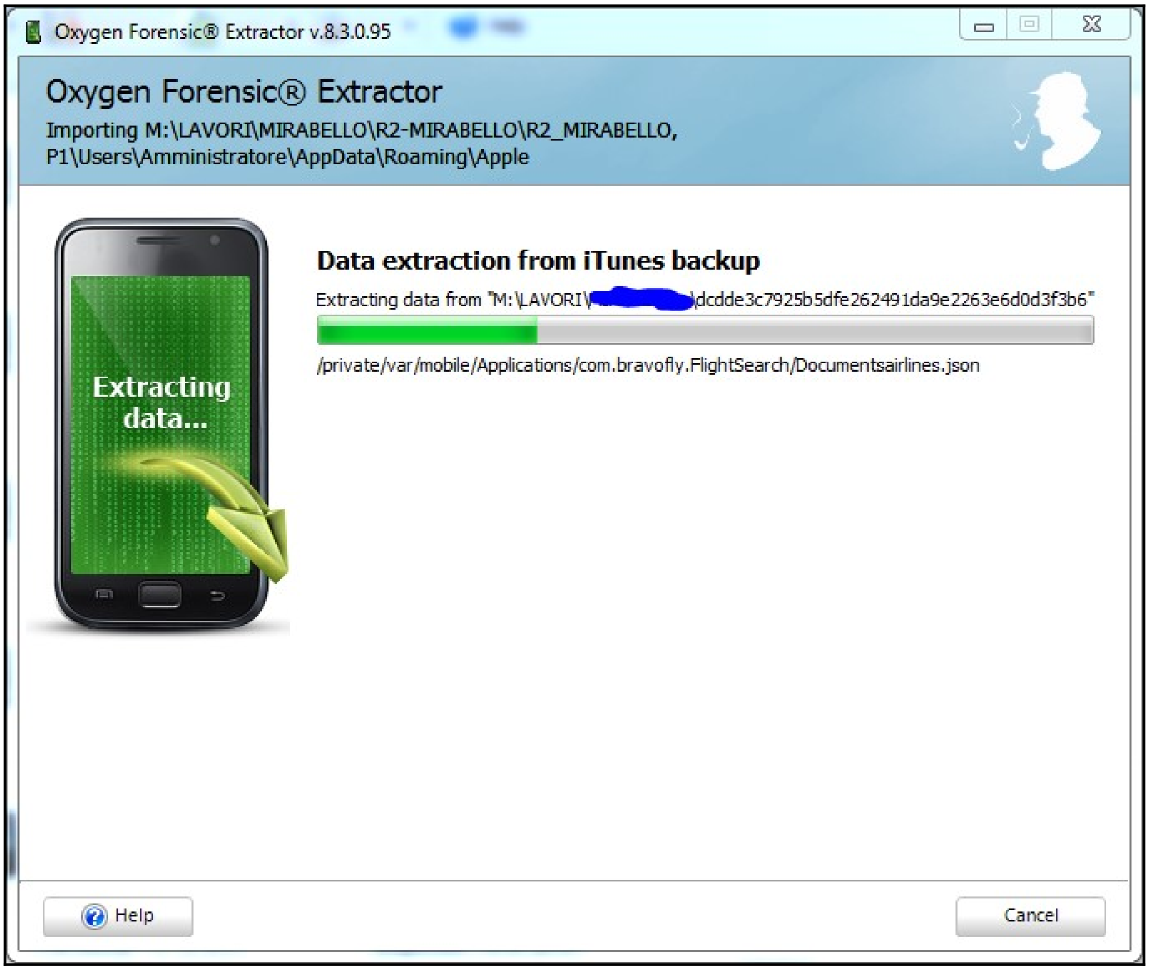
Device alias	iTunes Backup (dcdde3c7925b5dfe26	Hash algorithm	SHA-2
Case number		Inspector	Mattia Epifani
Evidence number		Device owner	
Place		Owner email	
Incident number		Owner phone number	Add numer
Backup password	☐ Enter the password		

☑ Parse applications databases including deleted records and collect data for analytical sections (Aggregated Contacts, Links and Stats, etc.). If not checked you can do it later in Oxygen Forensic® Analyst. Read more...

☑ Recover deleted data from applications Read more...

Device notes

The software then starts the analysis process and allows users to analyze the backup content. In `Chapter 6`, *Analyzing iOS Devices*, we will see a further example of the analysis performed with Oxygen Forensic Analyst:


Oxygen Forensic® Extractor v.8.3.0.95

Oxygen Forensic® Extractor

Importing M:\LAVORI\MIRABELLO\R2-MIRABELLO\R2_MIRABELLO, P1\Users\Amministratore\AppData\Roaming\Apple

Data extraction from iTunes backup

Extracting data from "M:\LAVORI\[redacted]\dcdde3c7925b5dfe262491da9e2263e6d0d3f3b6"

/private/var/mobile/Applications/com.bravofly.FlightSearch/Documentsairlines.json

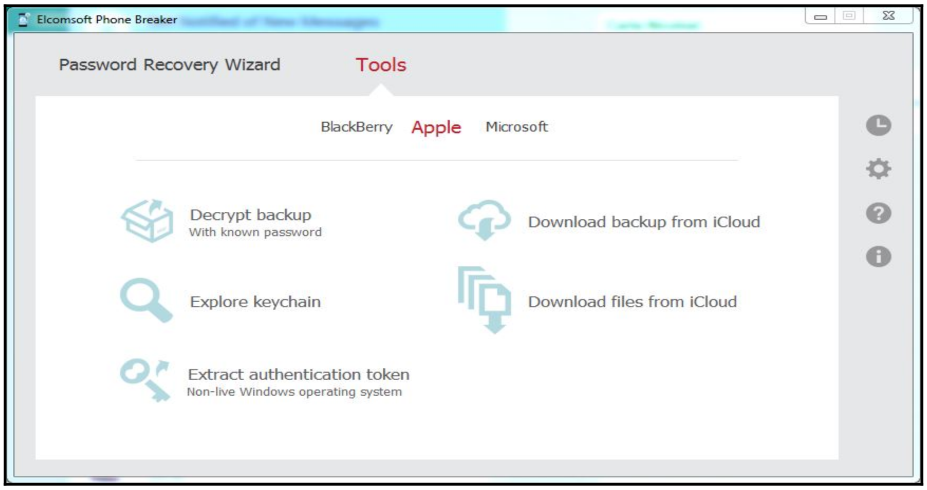
Encrypted iTunes backup cracking

As we explained in `Chapter 3`, *Evidence Acquisition from iDevices*, and in the first part of this chapter, an iTunes backup can be encrypted with a password chosen by the iDevice user. When you seize an iDevice with a backup password already set, or if you have a computer with a previously created encrypted backup, you can try to crack the backup using a dedicated tool. Currently, we were able to find only three software packages that can be used to crack an encrypted backup: EPB, Passware Forensic, and iPhone Backup Unlocker.

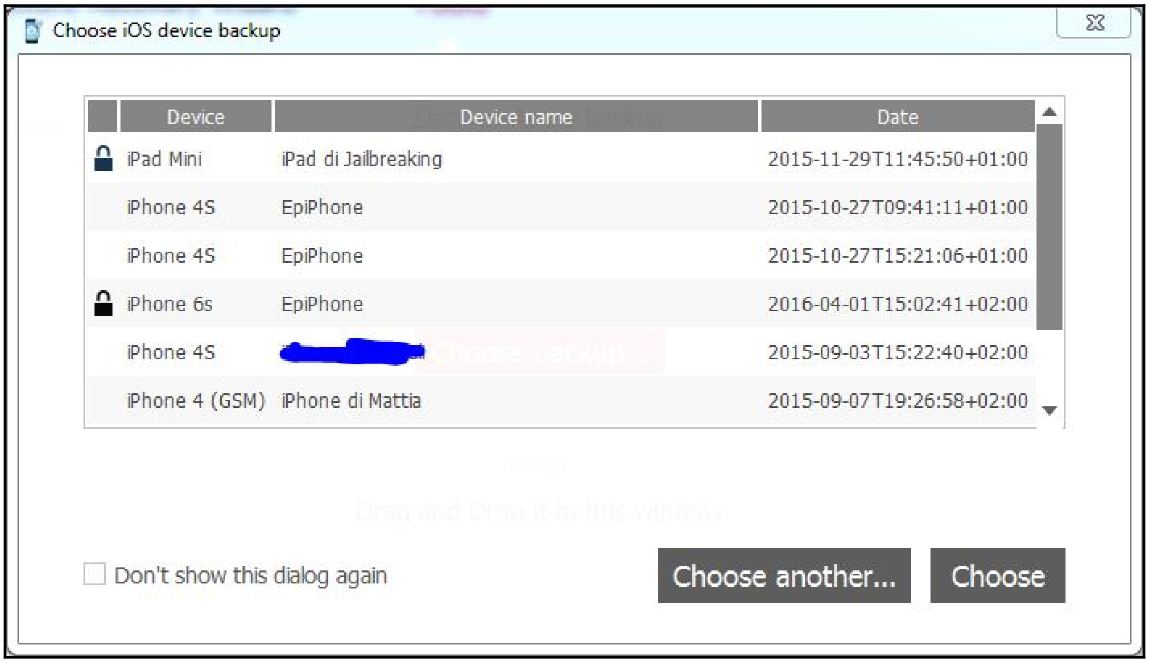
Case study – iTunes encrypted backup cracking with EPB

As mentioned on the product's website, Elcomsoft Phone Breaker enables forensic access to password-protected backups for smartphones and portable devices based on the Apple iOS platforms. The password recovery tool supports Apple devices running iOS, including iPhone, iPad, and iPod touch devices of all generations released to date.

After launching the tool, the first step is to load the encrypted backup by clicking on the **Decrypt backup** option from the main window (**Tools**) and selecting **Decrypt backup**, as shown in the following screenshot:



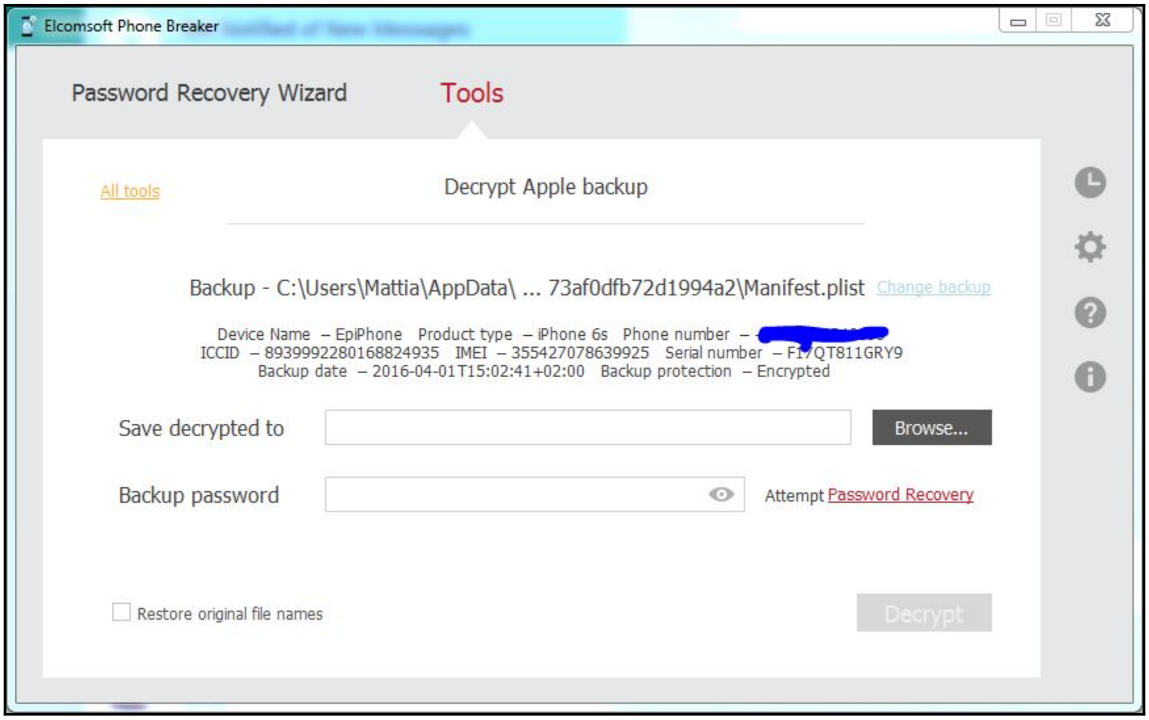
The software automatically provides a list of encrypted backups, saved in the default backup folder of the user who is executing the tool:



Choose iOS device backup

	Device	Device name	Date
🔒	iPad Mini	iPad di Jailbreaking	2015-11-29T11:45:50+01:00
	iPhone 4S	EpiPhone	2015-10-27T09:41:11+01:00
	iPhone 4S	EpiPhone	2015-10-27T15:21:06+01:00
🔒	iPhone 6s	EpiPhone	2016-04-01T15:02:41+02:00
	iPhone 4S	[redacted]	2015-09-03T15:22:40+02:00
	iPhone 4 (GSM)	iPhone di Mattia	2015-09-07T19:26:58+02:00

The analyst can choose one of the proposed encrypted backups or choose another folder containing other encrypted backups. After selecting the backup, the tool asks the analyst where they want to save the decrypted backup and, if known, to provide the backup password:



Backup - C:\Users\Mattia\AppData\ ... 73af0dfb72d1994a2\Manifest.plist

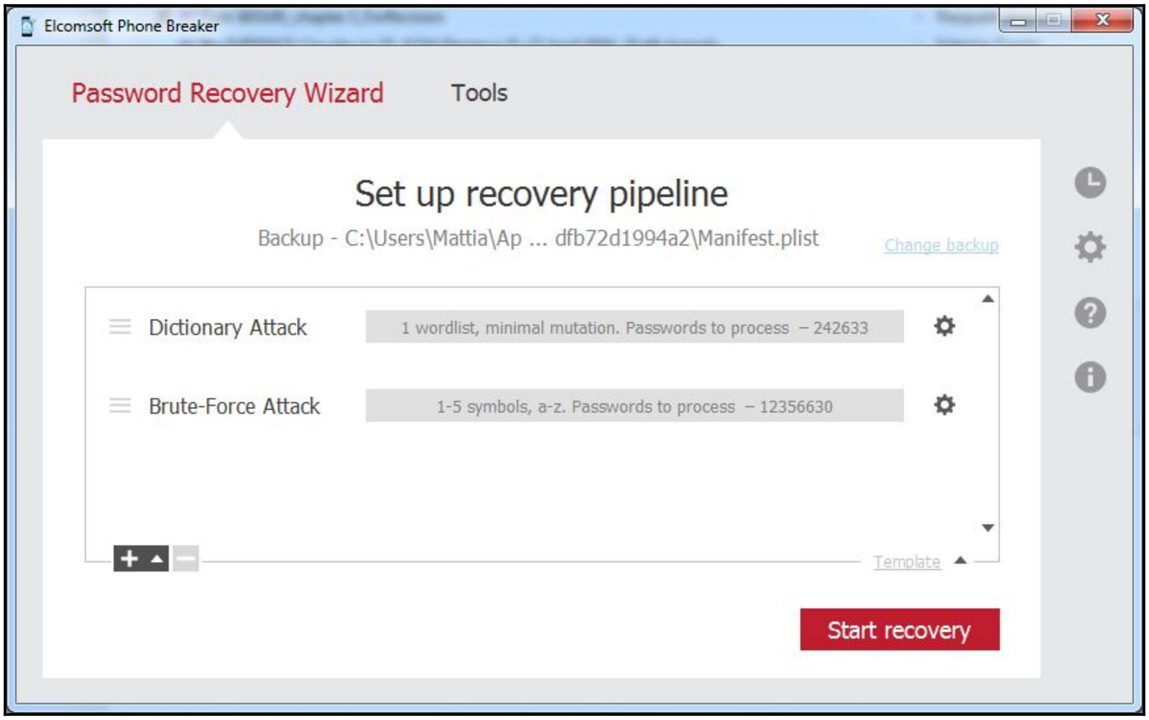
Device Name – EpiPhone Product type – iPhone 6s Phone number – [redacted]
ICCID – 8939992280168824935 IMEI – 355427078639925 Serial number – F17QT811GRY9
Backup date – 2016-04-01T15:02:41+02:00 Backup protection – Encrypted

Save decrypted to

Backup password

☐ Restore original file names

By clicking on **Attempt Password Recovery**, the user can select the type of cracking they want to perform. You can choose between two options: **Dictionary Attack** or **Brute-Force Attack**, as shown in the following screenshot:



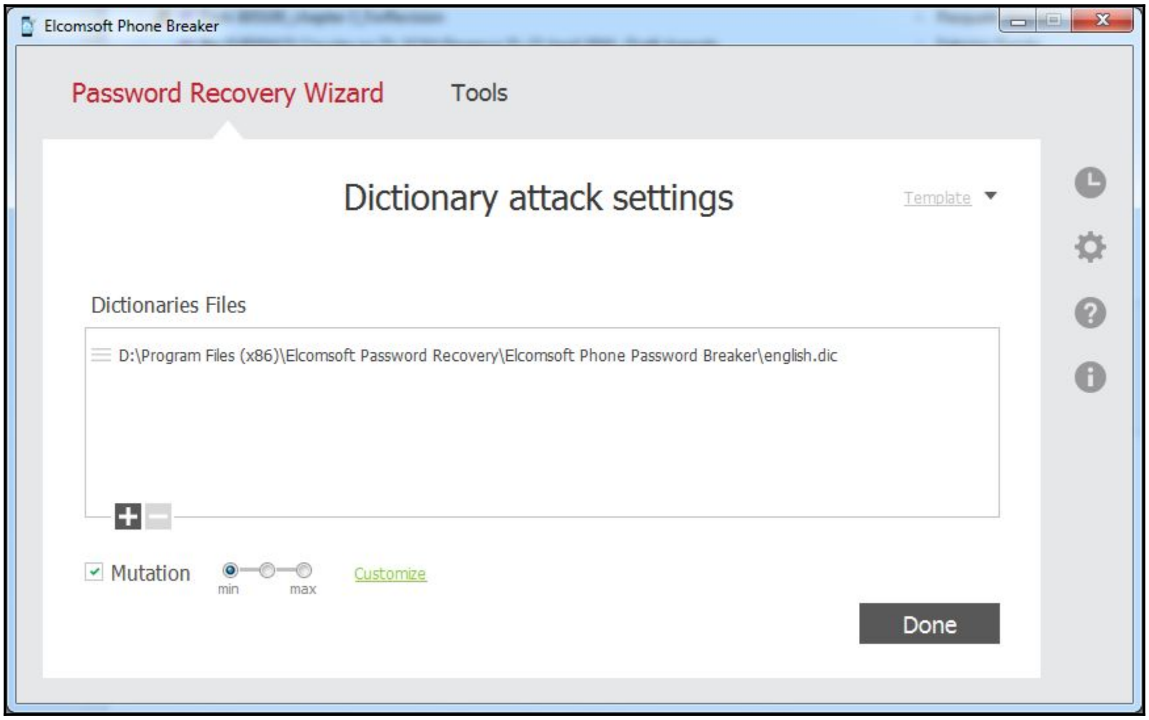
Set up recovery pipeline

Backup - C:\Users\Mattia\Ap ... dfb72d1994a2\Manifest.plist

Dictionary Attack — 1 wordlist, minimal mutation. Passwords to process – 242633

Brute-Force Attack — 1-5 symbols, a-z. Passwords to process – 12356630

In the first case, the analyst can provide a custom dictionary file, as shown in the following screenshot:



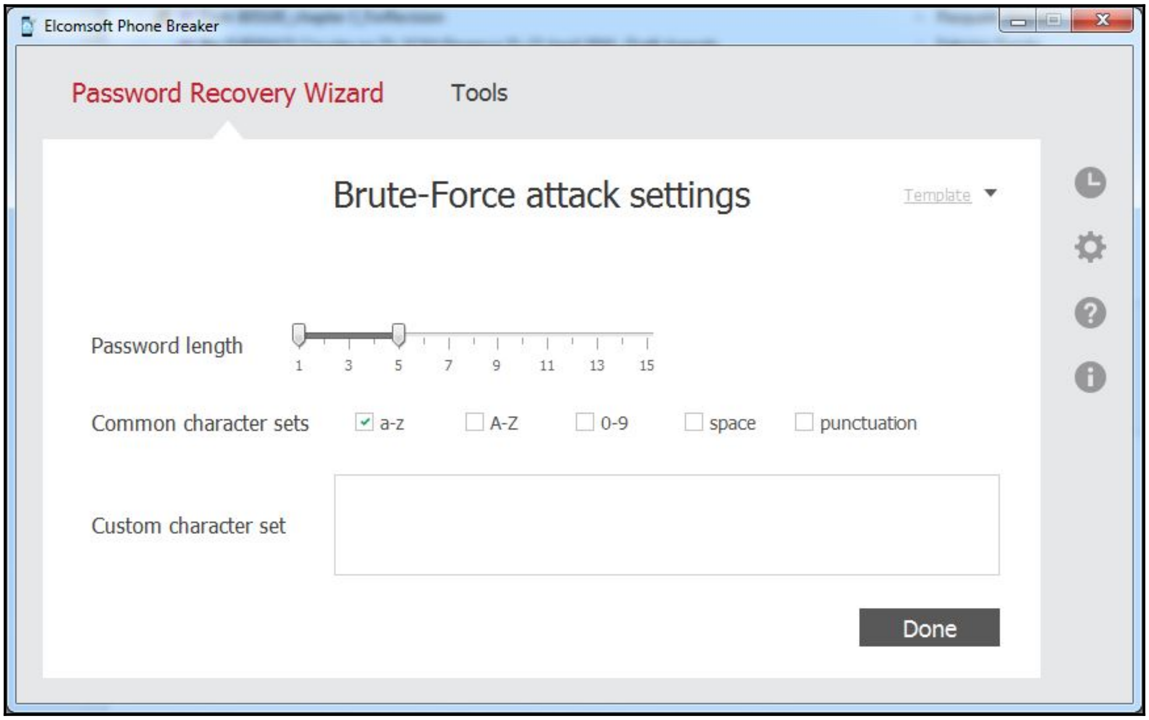
Dictionary attack settings

Dictionaries Files

D:\Program Files (x86)\Elcomsoft Password Recovery\Elcomsoft Phone Password Breaker\english.dic

✓ Mutation ● min max Customize

In the second case, the analyst can decide the parameters for the brute-force attack, as follows:



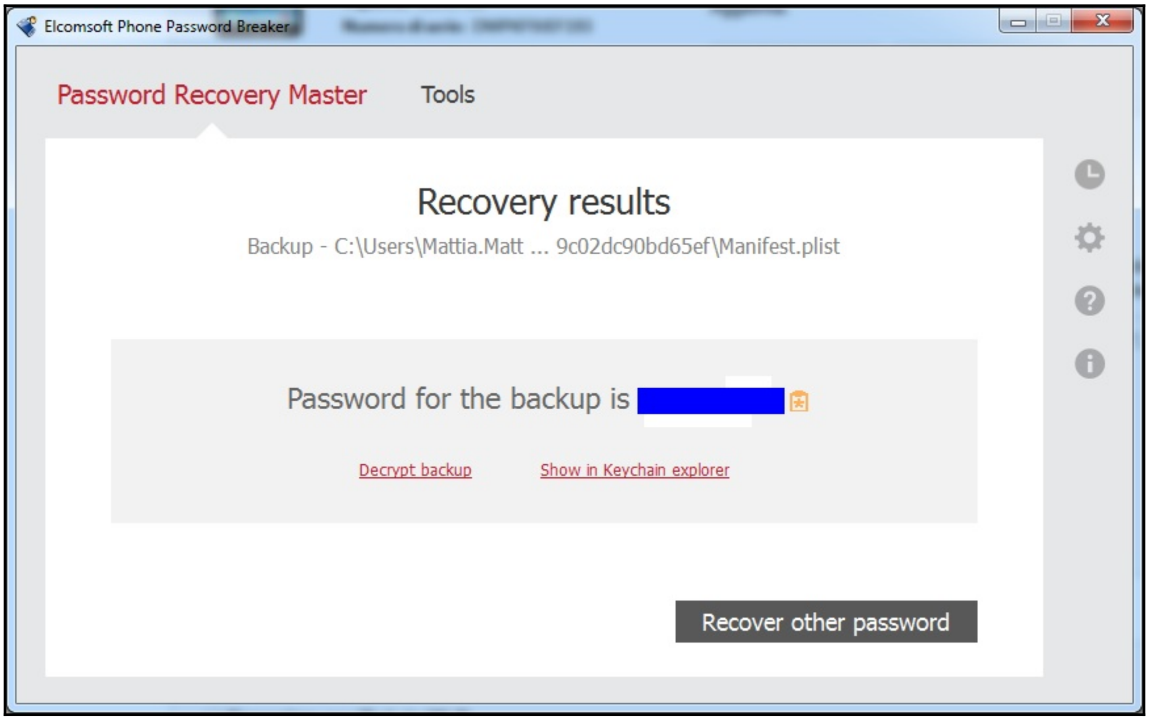
Brute-Force attack settings

Password length: 1 – 5

Common character sets: [✓ a-z] A-Z 0-9 space punctuation

Custom character set

If the cracking procedure is successful, the tool provides the password to the analyst and gives the options to decrypt the backup (so that it can be analyzed with one of the tools previously mentioned):

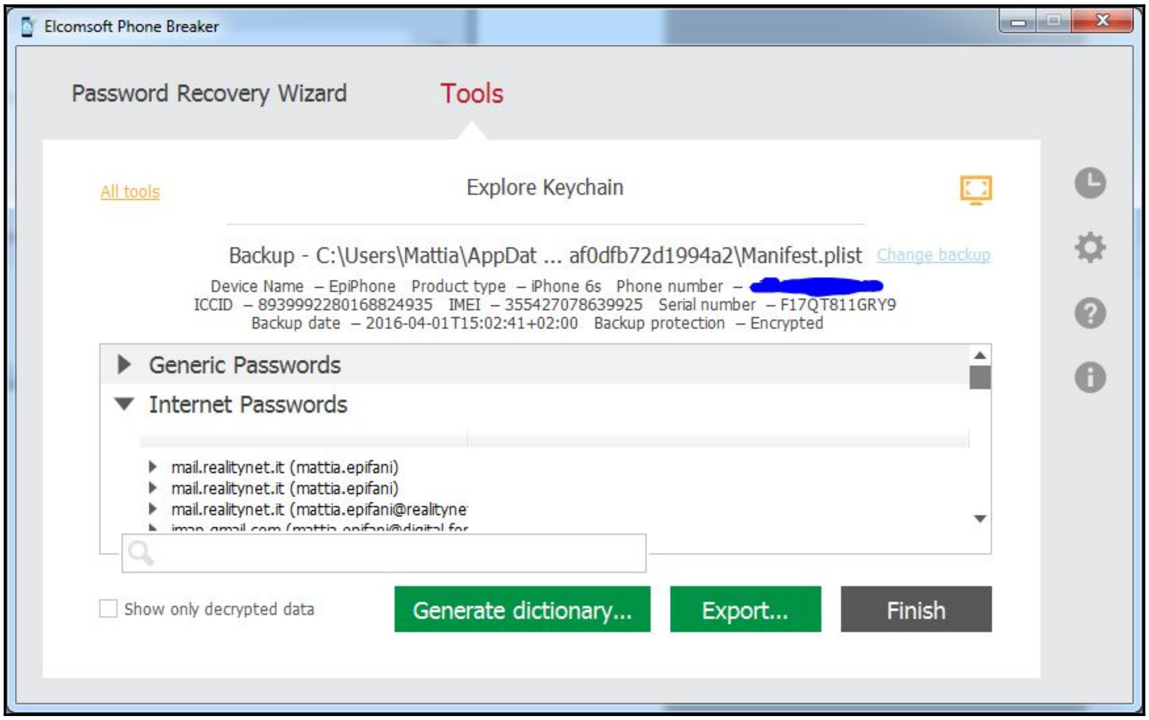


Recovery results

Backup - C:\Users\Mattia.Matt ... 9c02dc90bd65ef\Manifest.plist

Password for the backup is [redacted]

Otherwise, it is possible to show the keychain content with the username and password for the Wi-Fi network connection, e-mail accounts configured in the Mail app, stored Internet passwords, and stored passwords from other apps:



Explore Keychain

Backup - C:\Users\Mattia\AppDat ... af0dfb72d1994a2\Manifest.plist Change backup

Device Name – EpiPhone Product type – iPhone 6s Phone number – [redacted]
ICCID – 8939992280168824935 IMEI – 355427078639925 Serial number – F17QT811GRY9
Backup date – 2016-04-01T15:02:41+02:00 Backup protection – Encrypted

Generic Passwords
Internet Passwords
mail.realitynet.it (mattia.epifani)
mail.realitynet.it (mattia.epifani)
mail.realitynet.it (mattia.epifani@realityne
imap.gmail.com (mattia.epifani@digital-for

Summary

In this chapter, we explained the most useful information about an iTunes backup related to the forensic analysis of an iOS device. In particular, we illustrated how the backup is structured and how to parse it with commercial and open source tools. We also explained the differences between an unencrypted and encrypted backup and suggested some ways to try to crack the backup password. A really interesting point about the iTunes backup is that if the device does not have a backup password already set by its owner, when preforming the acquisition, you can create an encrypted backup, choosing a known password in order to be able to access the password saved in the `Keychain` file without the need for cracking. Instead, if you happen to have an encrypted backup for which you are not able to crack the password, it is anyway possible to analyze the `plist` files and the content of the `Manifest.mbdb` file, recovering in this way the list of all files present inside that backup. In the next chapter, how to recover data from the user iCloud account with either its credentials or authentication token will be explained.

Self-test questions

Q1. In which folder are the iOS devices backup stored in Windows 7?

1. `C:\Users\[username]\AppData\Roaming\Apple Computer\MobileSync\Backup`
2. `C:\Users\[username]\AppData\Local\Apple Computer\MobileSync\Backup`
3. `C:\Users\[username]\AppData\Apple Computer\MobileSync\Backup`
4. `C:\Program Data\Apple Computer\MobileSync\Backup`

Q2. Which file contains information about the backup (such as backup date, device name, and so on)?

1. `Manifest.plist`
2. `Info.plist`
3. `Status.plist`
4. `Manifest.mbdb`

Q3. Which file contains the description of all the files in the backup folder?

1. `Manifest.plist`
2. `Info.plist`
3. `Status.plist`
4. `Manifest.mbdb`

Q4. Which backup domain contains multimedia elements related to the camera?

1. App domain
2. Camera Roll domain
3. Media domain
4. Keychain domain

5
Evidence Acquisition and Analysis from iCloud

The goal of this chapter is to introduce the cloud system provided by Apple to all its users through which they can save the backups of their devices and other files on remote servers. In the first part of the chapter, we will show you the main characteristics of this service, and then the techniques to create and recover a backup and other files from iCloud.

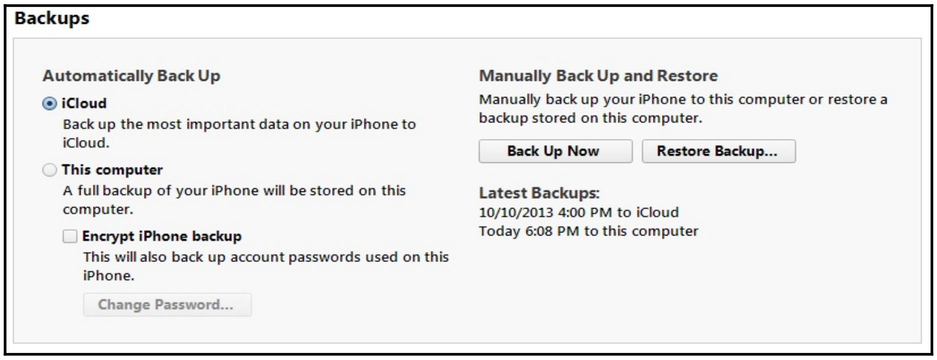
The iCloud service

iCloud is a cloud storage and cloud computing service designed by Apple and offered to iOS users since iOS 5. The iCloud service was announced on June 6, 2011 during the Apple Worldwide Developers Conference, and became available to the public from October 12, 2011. In February 2016, iCloud had more than 780 million users. The service allows users to store data on iCloud Drive, share photos and videos between various devices and people, perform geolocation for their Apple device with Find My iPhone, and save personal passwords with iCloud Keychain on remote servers. The iCloud Drive storage space can also be used by individual applications to store information and specific backup data; for example, WhatsApp, at the time of the first configuration or following a version upgrade, asks the user whether they wants to use iCloud to store backups of conversations and WhatsApp media files, and if so, how often (daily, weekly, or monthly). iCloud also allows users to synchronize data between devices (e-mail, contacts, calendars, bookmarks, notes, reminders, iWork documents, and so on) or to make a backup of an iOS device (iPhone, iPad, or iPod touch) on remote servers rather than using iTunes and your local computer. The data uploaded to iCloud can also be shared with Apple computers running Mac OS X (native support) or a personal computer running Windows Vista or later, by installing the iCloud Control Panel software, which can be downloaded for free from the Apple website.

Each iCloud account has 5 GB of free storage. Purchases made through iTunes (music, apps, videos, movies, and so on) are not calculated in the count of the occupied space and can be stored in iCloud and downloaded on all devices associated with the Apple ID of the user. Moreover, the user has the option to purchase additional storage in denominations of 50 GB ($0.99 a month), 200 GB ($2.99 a month), or 1,000 GB ($9.99 a month).

iDevice backup on iCloud

iCloud allows users to make online backups of iDevices so that they will be able to restore their data even on a different iDevice (for example, in case of replacement of devices). The choice of which backup mode to use can be done directly in the settings of the device or through iTunes when the device is connected to the PC or Mac, as shown in the following screenshot:



Backups

Automatically Back Up

(●) **iCloud**
Back up the most important data on your iPhone to iCloud.

( ) **This computer**
A full backup of your iPhone will be stored on this computer.

☐ **Encrypt iPhone backup**
This will also back up account passwords used on this iPhone.

Manually Back Up and Restore
Manually back up your iPhone to this computer or restore a backup stored on this computer.

Latest Backups:
10/10/2013 4:00 PM to iCloud
Today 6:08 PM to this computer

Once the user has activated the service, the device automatically backs up, at least once a day, and each time all the following conditions are met:

- It is connected to the power cable
- It is connected to a Wi-Fi network
- The screen is locked
- Enough space on iCloud is available

iCloud online backups are incremental through subsequent snapshots, and each snapshot is the current status of the device at the time of its creation. The structure of the backup stored on iCloud is entirely analogous to that of the backup made with iTunes.

iDevice backup acquisition

Backups that are made online are, to all intents and purposes, not encrypted. Technically, they are encrypted, but the encryption key is stored with the encrypted files. Apple made this choice in order for users to be able to restore the backup on a different device from the one that created it. Currently, the acquisition of iCloud backup is supported by various commercial software (**Elcomsoft Phone Breaker** (**EPPB**), Passware, Oxygen Forensics Detective, Cellebrite Cloud Analyzer, iPhone Backup Extractor, and Wondershare Dr.Fone) and two open source tools (iLoot, which is available at `https://github.com/hackappcom/iloot` and works up to iOS 8 and InflatableDonkey, which is available at `https://github.com/horrorho/InflatableDonkey` and works with iOS 9). The interesting aspect is that this same technique was used in the iCloud hack performed in 2014, when personal photos and videos were hacked from the respective iCloud services and released over the Internet (more information is available at `http://en.wikipedia.org/wiki/214_celebrity_photo_hack`). Although there is no strong evidence yet that describes how the hack was made, it is believed that Apple's *Find My iPhone* service was responsible for this and Apple did not implement any security measure to lock down the account after a particular number of wrong login attempts, which directly raises the possibility of exploitation (brute force, in this case). The tool used to brute force the iCloud password, named iBrute, is still available at `https://github.com/hackappcom/ibrute`, but it has not been working since January 2015. A new tool called iBrutr`https://github.com/Prx13/iBrutr`, which uses a different iCloud service vulnerability, was released in September 2015.

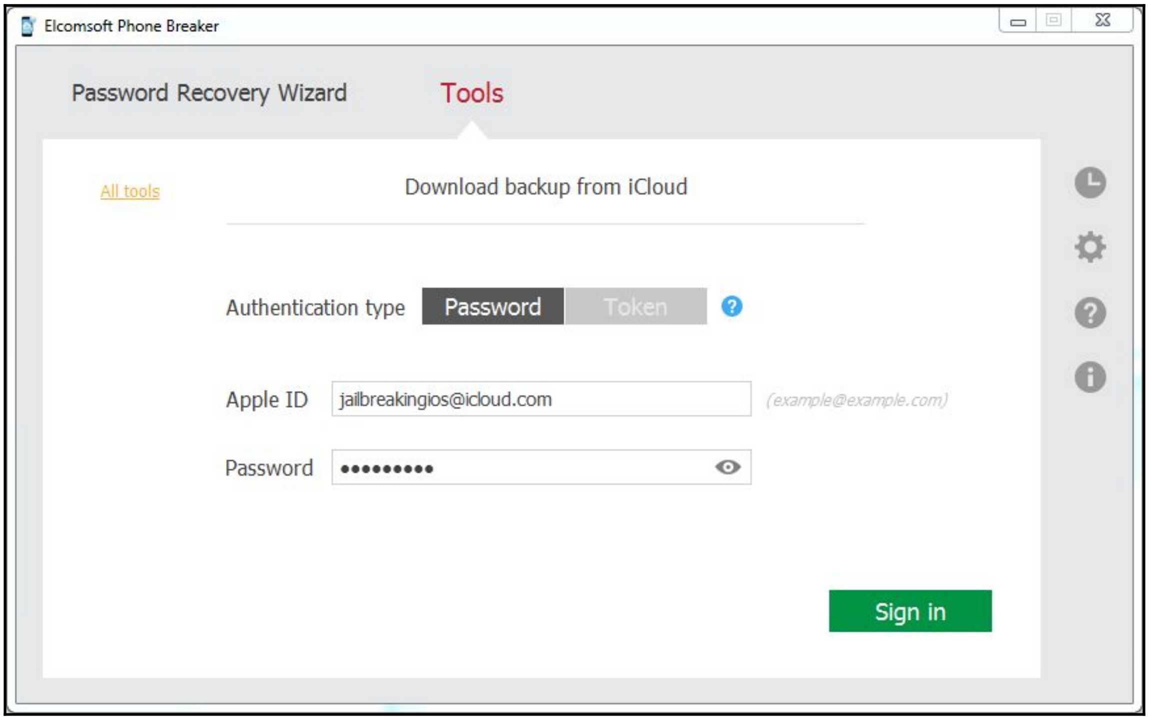
Case study - iDevice backup acquisition and EPPB with username and password

As reported on the software manufacturer's website, EPPB allows the acquisition of data stored on an online backup. Moreover, online backups can be acquired without having the original iOS device to hand. All that's needed to access online backups stored in the cloud service are the original user's credentials, including their Apple ID, accompanied with the corresponding password or an authentication token extracted from a computer (detailed later in this chapter).

iCloud login credentials can be retrieved as follows:

- Using social engineering techniques
- From a PC (or a Mac) on which they are stored
- Using iTunes Password Decryptor (`http://securityxploded.com/`)
- Using WebBrowserPassView (`http://www.nirsoft.net/`)
- Directly from the device (iPhone/iPad/iPod touch) by extracting the credentials stored in the Keychain, as explained in `Chapter 4`, *Evidence Acquisition and Analysis from iTunes Backup*

Once the credentials have been extracted, the download of the backup is very simple. Follow the step-by-step instructions provided in the program by entering the username and password in the **Download backup from iCloud** dialog box by navigating to **Tools** | **Apple** | **Download backup from iCloud** | **Password** and clicking on **Sign in**, as shown in the following screenshot:

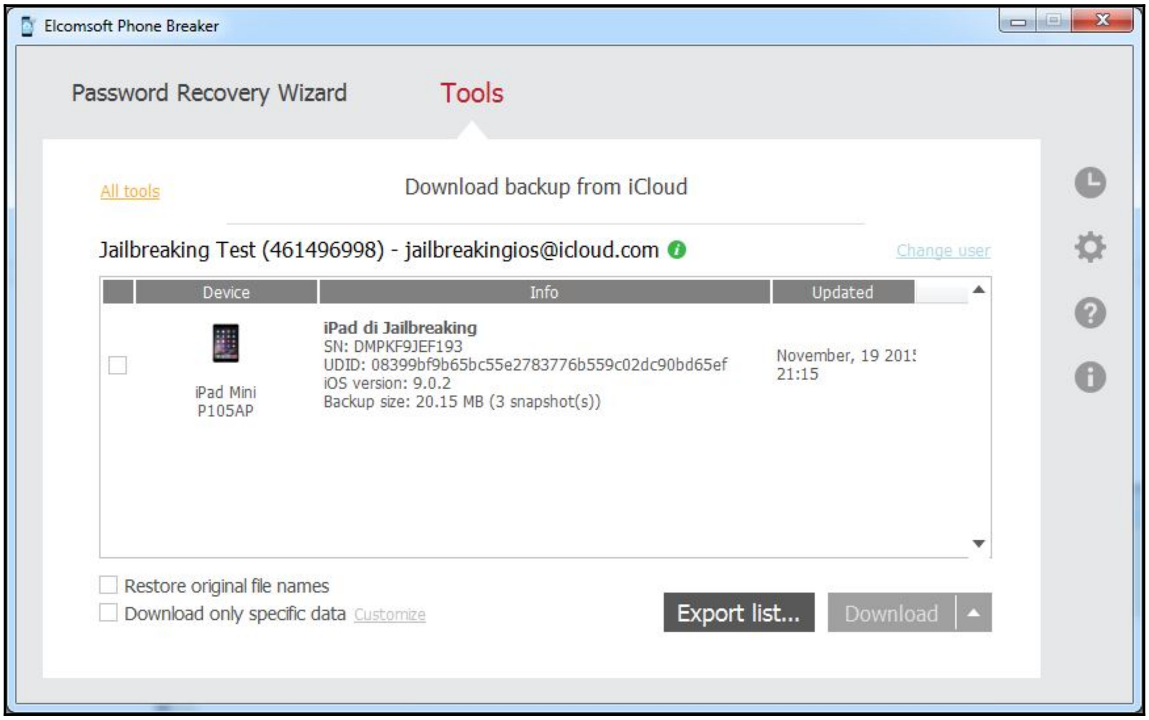


Authentication type: Password | Token

Apple ID: jailbreakingios@icloud.com (example@example.com)

Password: ••••••••••

At this point, the software displays a screen that shows all the backups present in the user account and allows you to download data, as shown here:



Download backup from iCloud

Jailbreaking Test (461496998) - jailbreakingios@icloud.com

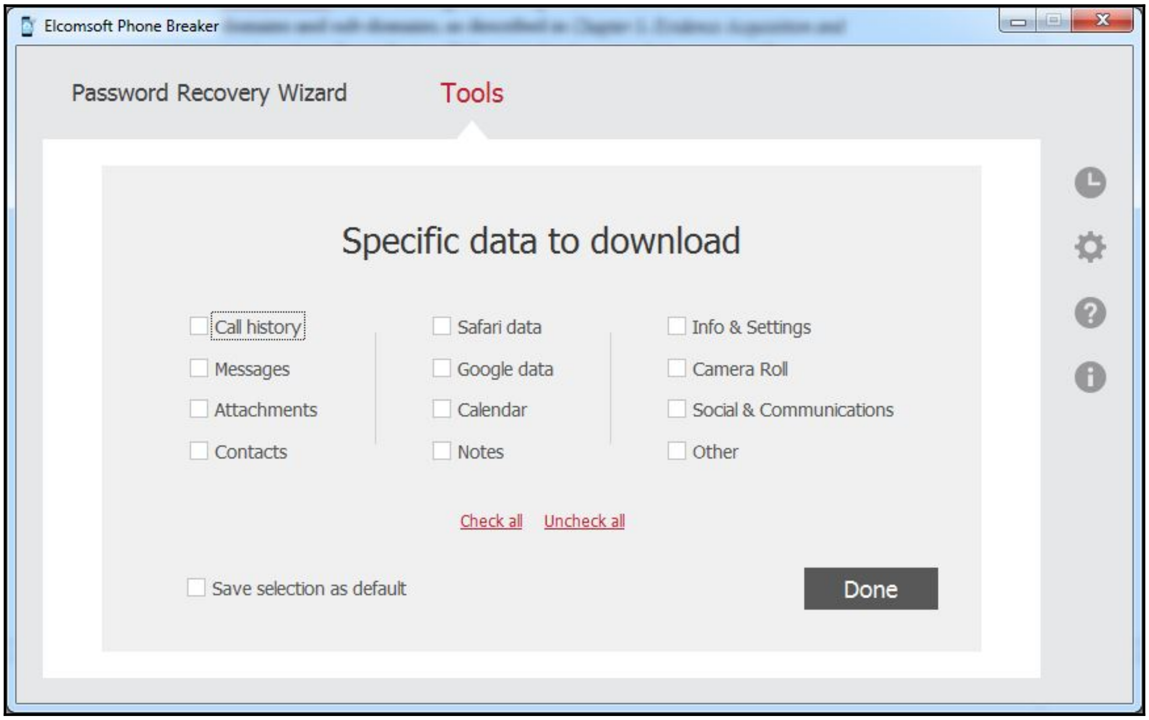
	Device	Info	Updated
☐	iPad Mini P105AP	**iPad di Jailbreaking** SN: DMPKF9JEF193 UDID: 08399bf9b65bc55e2783776b559c02dc90bd65ef iOS version: 9.0.2 Backup size: 20.15 MB (3 snapshot(s))	November, 19 201! 21:15

☐ Restore original file names
☐ Download only specific data Customize

It is important to notice the possibility of using the following two options:

- **Restore original file names**: If enabled, this option interprets the contents of the `Manifest.mbdb` file and rebuilds the backup with the same tree structure into domains and subdomains as described in `Chapter 4`, *Evidence Acquisition and Analysis from iTunes Backup*. If the investigator intends to carry out the analysis with traditional software for data extraction from backups, it is recommended that you disable this option, because if enabled, that software will no longer be able to parse the backup.

- **Download only specific data**: This option is very useful when the investigator needs to download only some specific information. Currently, the software supports **Call history**, **Messages**, **Attachments**, **Contacts**, **Safari data**, **Google data**, **Calendar**, **Notes**, **Info & Settings**, **Camera Roll**, **Social & Communications**, and so on. In this case, the **Restore original file names** option is automatically activated and it cannot be disabled:

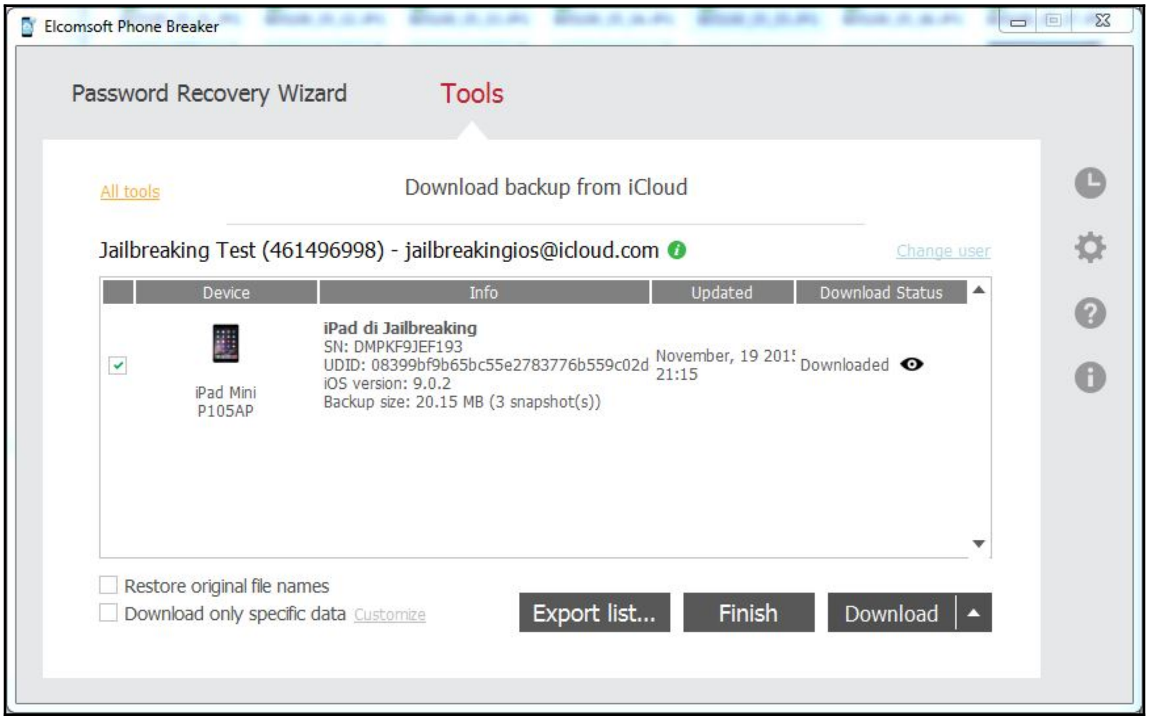


Specific data to download

☐ Call history	☐ Safari data	☐ Info & Settings
☐ Messages	☐ Google data	☐ Camera Roll
☐ Attachments	☐ Calendar	☐ Social & Communications
☐ Contacts	☐ Notes	☐ Other

☐ Save selection as default

Once you have chosen the destination folder for the download, the backup starts. The time required to download depends on the size of the storage space available to the user and the number of snapshots stored within that space:



Jailbreaking Test (461496998) - jailbreakingios@icloud.com

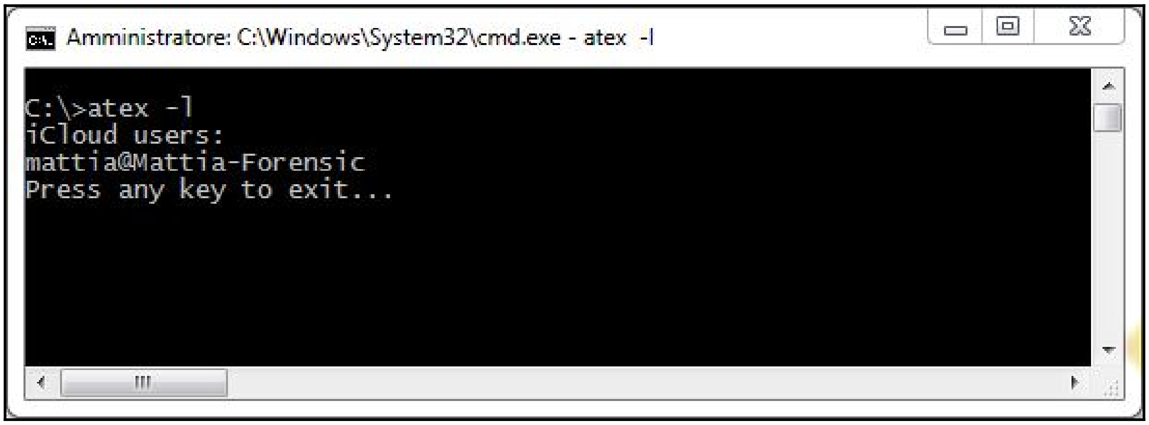
	Device	Info	Updated	Download Status
✓	iPad Mini P105AP	iPad di Jailbreaking SN: DMPKF9JEF193 UDID: 08399bf9b65bc55e2783776b559c02d iOS version: 9.0.2 Backup size: 20.15 MB (3 snapshot(s))	November, 19 201! 21:15	Downloaded

Case study - iDevice backup acquisition and EPPB with authentication token

The Forensic edition of Phone Breaker from Elcomsoft is a tool that gives a digital forensics examiner the power to obtain iCloud data without having the original Apple ID and password. This kind of access is made possible via the use of an authentication token extracted from the user's computer. These tokens can be obtained from any suspect's computer where iCloud Control Panel is installed. Tokens can be extracted from such computers both live while it's running and afterwards from its hard drive, by extracting specific files and folders.

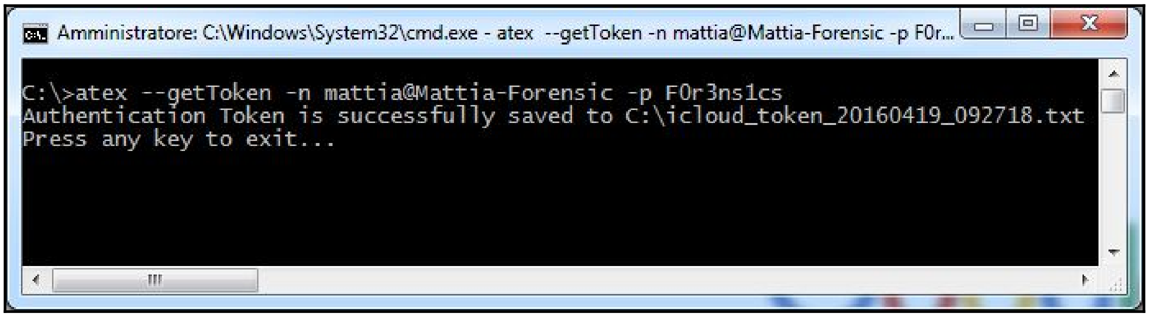
In the first case, on a live system, the analyst needs to use a small executable file called `atex.exe` on the machine. The executable file can be launched from an external pen drive during a live forensics activity.

Open Command Prompt and launch the `atex -l` command to list all the local iCloud users as follows:



```
C:\>atex -l
iCloud users:
mattia@Mattia-Forensic
Press any key to exit...
```

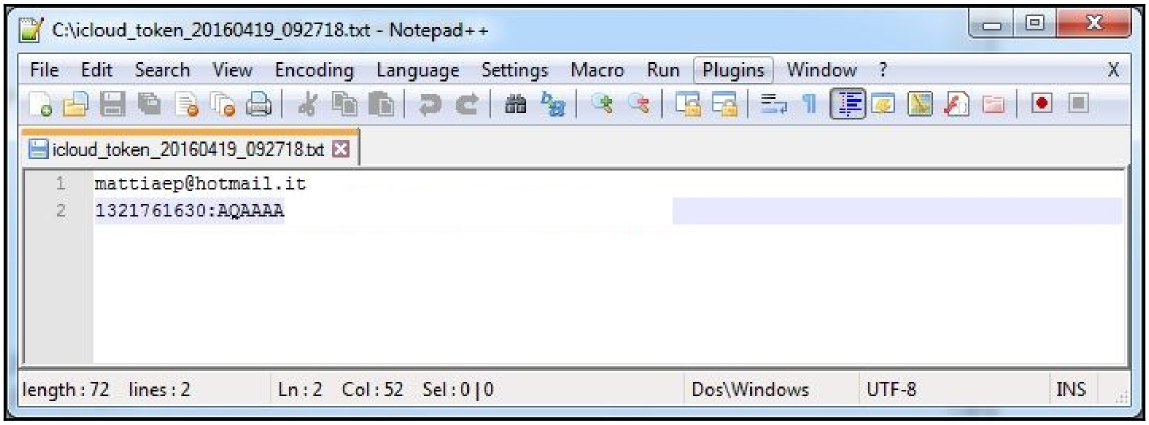
Then, launch `atex.exe` again with the `--getToken` parameter and enter the username of the specific local Windows user (`-n username`) and the password for this user's Windows account (`-p password`):



Amministratore: C:\Windows\System32\cmd.exe - atex --getToken -n mattia@Mattia-Forensic -p F0r...

```
C:\>atex --getToken -n mattia@Mattia-Forensic -p F0r3ns1cs
Authentication Token is successfully saved to C:\icloud_token_20160419_092718.txt
Press any key to exit...
```

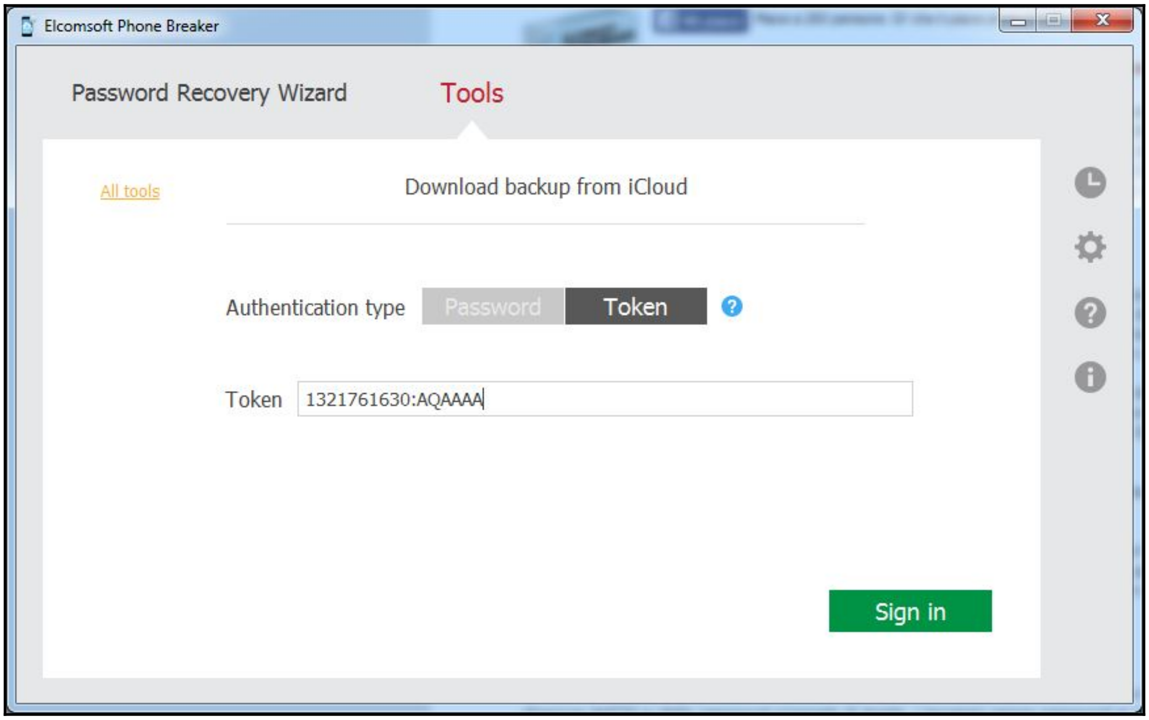
A file called `icloud_token_<timestamp>.txt` will be created in the folder from which `atex.exe` was launched. This file contains the Apple ID of the current iCloud Control Panel user and its authentication token:



C:\icloud_token_20160419_092718.txt - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
icloud_token_20160419_092718.txt
length : 72 lines : 2 Ln : 2 Col : 52 Sel : 0 | 0 Dos\Windows UTF-8 INS

```
mattiaep@hotmail.it
1321761630:AQAAAA
```

Now that the analyst has the authentication token, they can start the EPPB software, navigate to **Tools** | **Apple** | **Download backup from iCloud** | **Token**, copy and paste the token (be careful to copy the entire second row from the `.txt` file created by the `atex.exe` tool) into the software, and click on **Sign in**, as shown in the following screenshot. At this point, the software shows the screen for downloading the iCloud backups stored within the iCloud space of the user, in a similar way as you provide a username and password:



Authentication type: Password [✓ Token]

Token: 1321761630:AQAAAA

The procedure for the Mac OS X version is exactly the same. Just launch the `atex` Mac version from a shell and follow the steps as shown previously in the Windows environment:

- `sudo atex -l`: This command is used to get the list of all iCloud users.
- `sudo atex --getTokenOnline -u <username>`: This command is used to get the authentication token for a specific user. You will need to enter the user's system password when prompted.

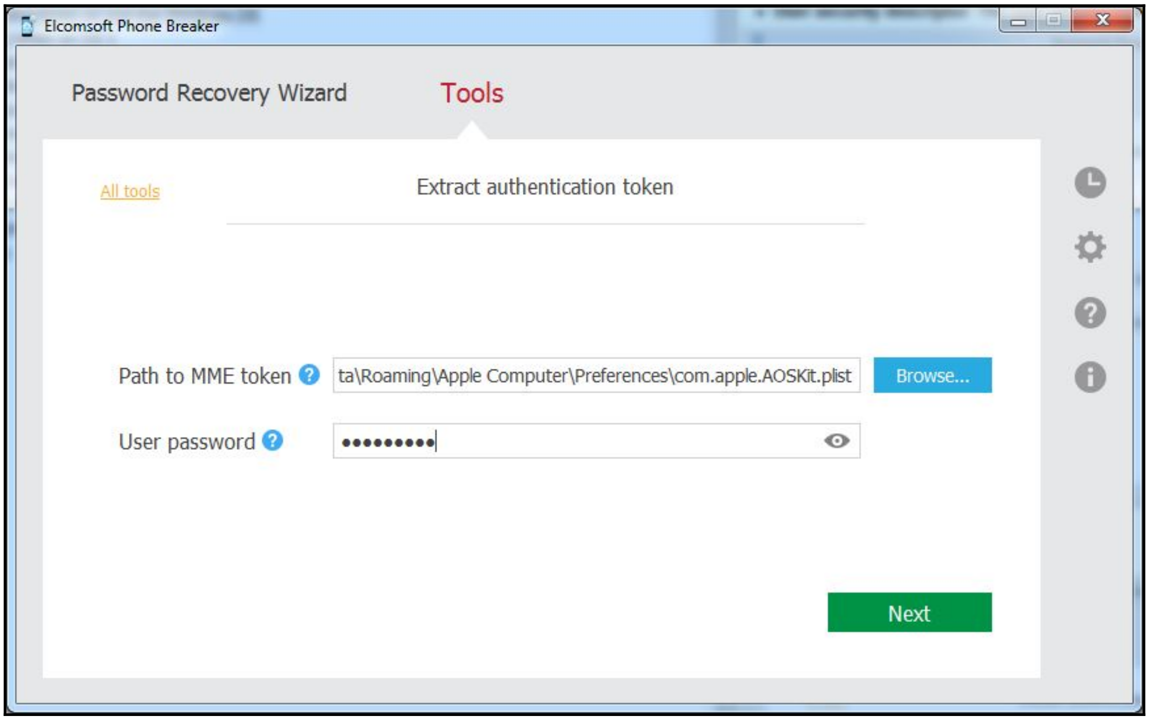
Tokens can also be extracted from a switched off computer, running either Mac OS X or MS Windows, where iCloud Control Panel is installed and configured.

In a MS Windows environment, in order to obtain the token, you need to extract the following file and folder (or its forensics image) from the computer:

- The `com.apple.AOSKit.plist` file, typically located in `\Users\<username>\AppData\Roaming\Apple Computer\Preferences\`
- The `\Users\<username>\AppData\Roaming\Microsoft\Protect\<SID>\` folder

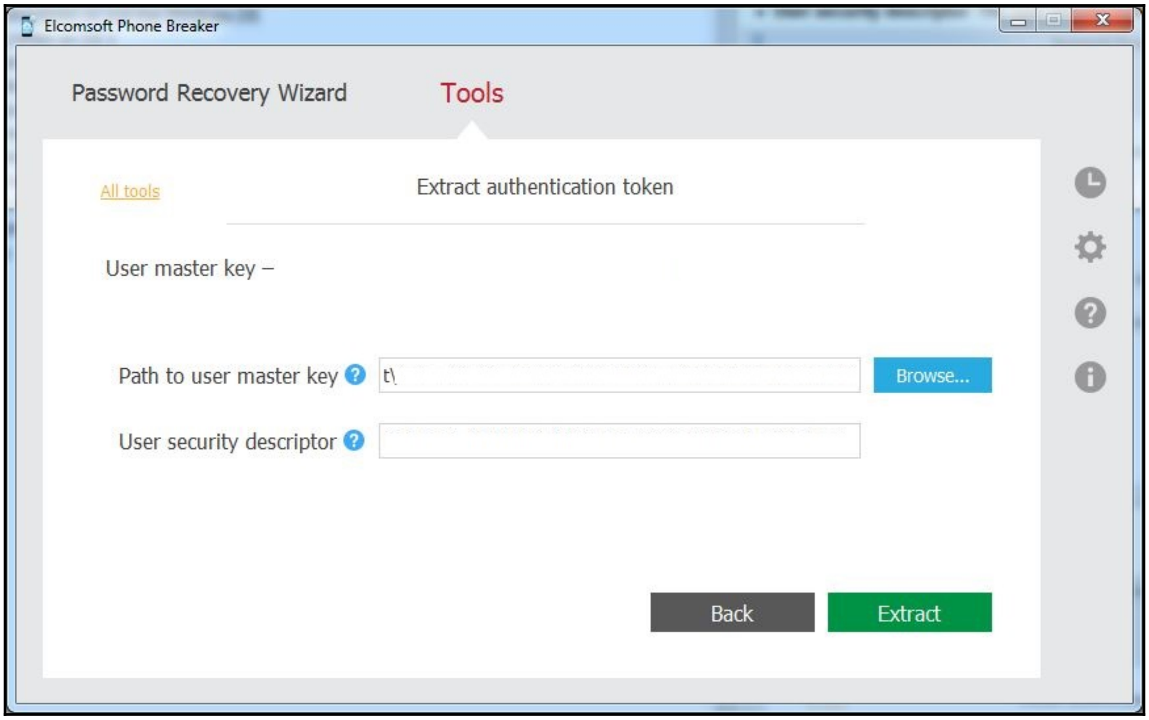
It is also necessary to know the account password for the local user on the target computer.

Once these files have been extracted, you can start the EPPB software and navigate to **Tools | Apple | Extract authentication token**. The software prompts you to provide the `com.apple.AOSKit.plist` file and the local user password:



Path to MME token: ta\Roaming\Apple Computer\Preferences\com.apple.AOSKit.plist
User password: ●●●●●●●●●

The next step is you need to provide the path to the folder containing the user's master key:

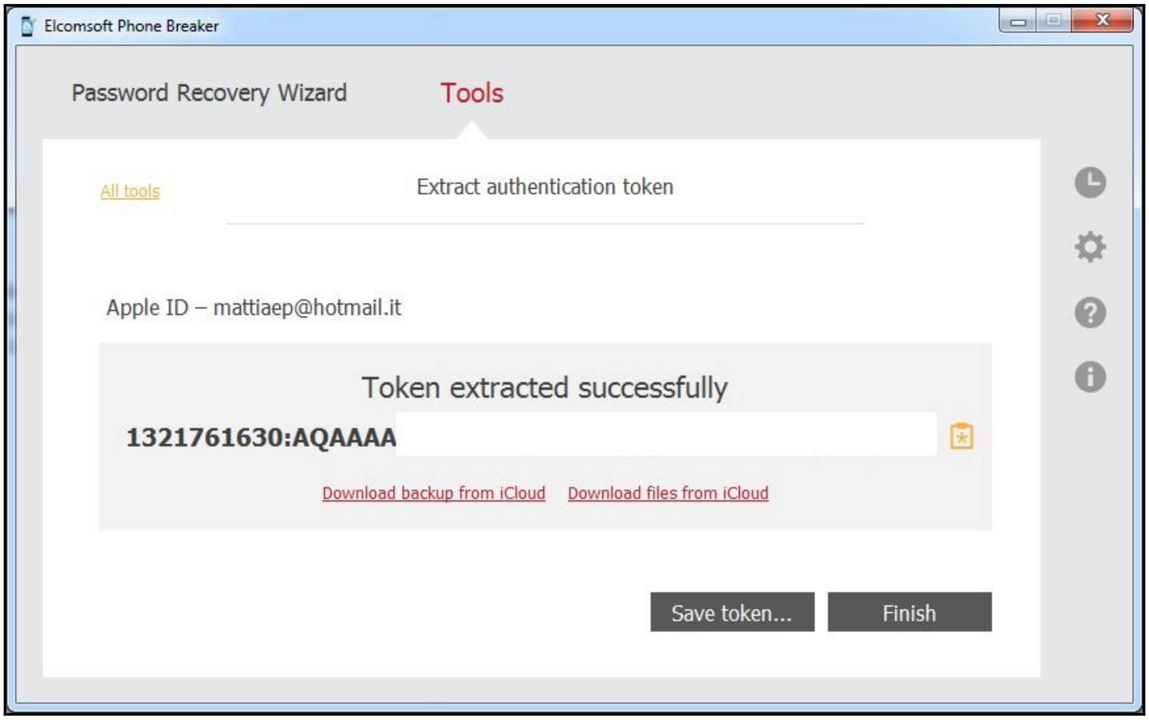


User master key –

Path to user master key t\

User security descriptor

If the information provided is correct, the software will show the authentication token, allowing direct access to the download features of backup from iCloud and files from the iCloud Drive:



Elcomsoft Phone Breaker

Password Recovery Wizard   Tools

All tools

Extract authentication token

Apple ID – mattiaep@hotmail.it

Token extracted successfully

1321761630:AQAAAA[redacted]

Download backup from iCloud   Download files from iCloud

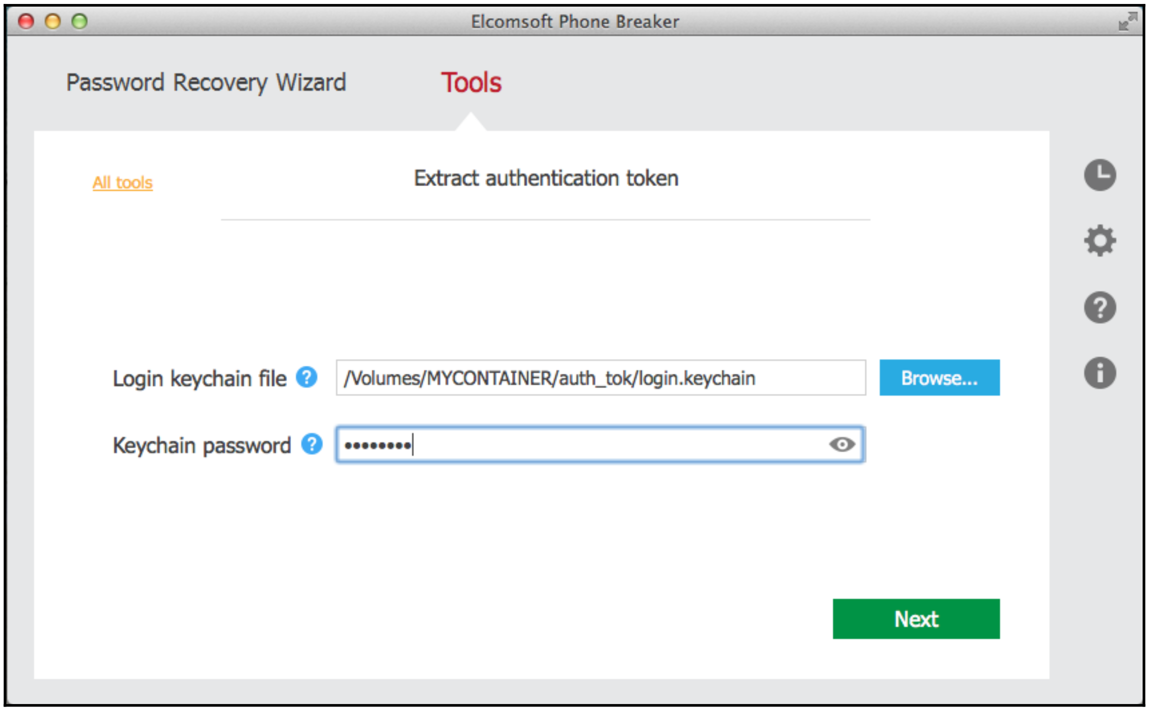
Save token...   Finish

In the Mac OS X environment, in order to obtain the token, you need to extract the following file and folder (or its forensics image) from the computer:

- The `login.keychain` file, stored by default in `/Users/<username>/Library/Keychains/`
- The folder containing the authentication token, stored by default in `/Users/<username>/Library/Application Support/iCloud/Accounts/`

It is also necessary to know the account password for the local user on the target computer.

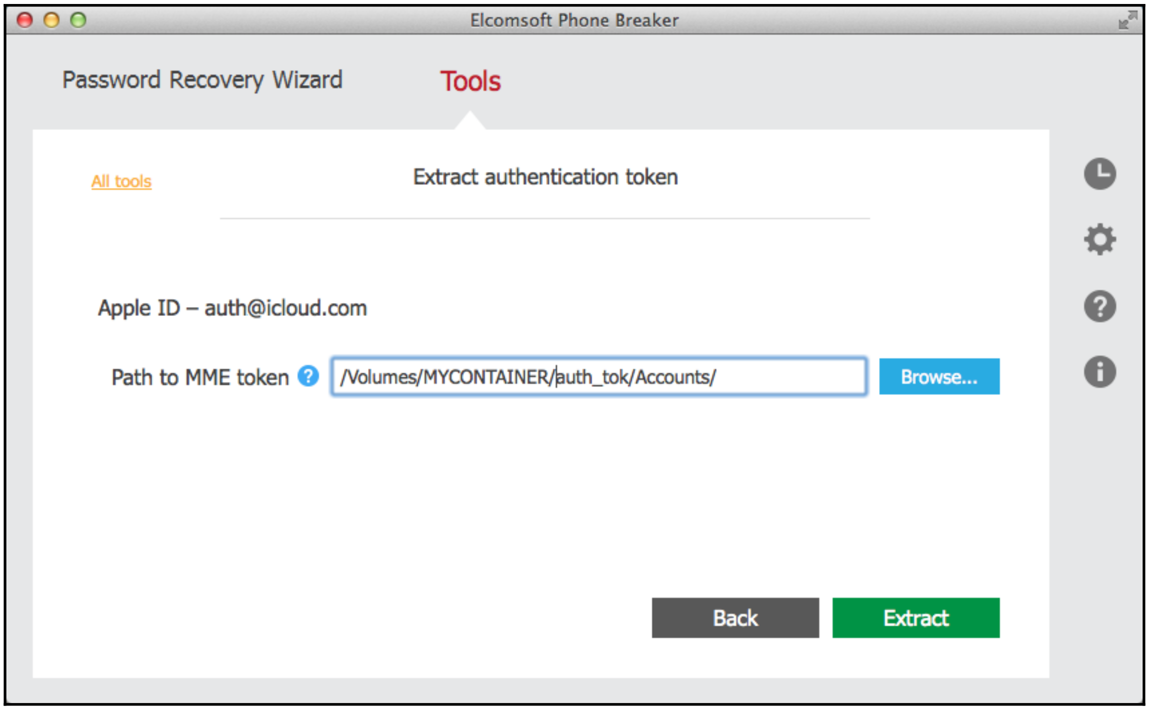
Once these files have been extracted, you can start the EPPB software and navigate to **Tools** | **Apple** | **Extract authentication token**. The software prompts you to provide the `login.keychain` file and the local user password:



Login keychain file: /Volumes/MYCONTAINER/auth_tok/login.keychain

Keychain password: •••••••••

The next step you need is to provide the path to the folder containing the authentication token:



Extract authentication token

Apple ID – auth@icloud.com

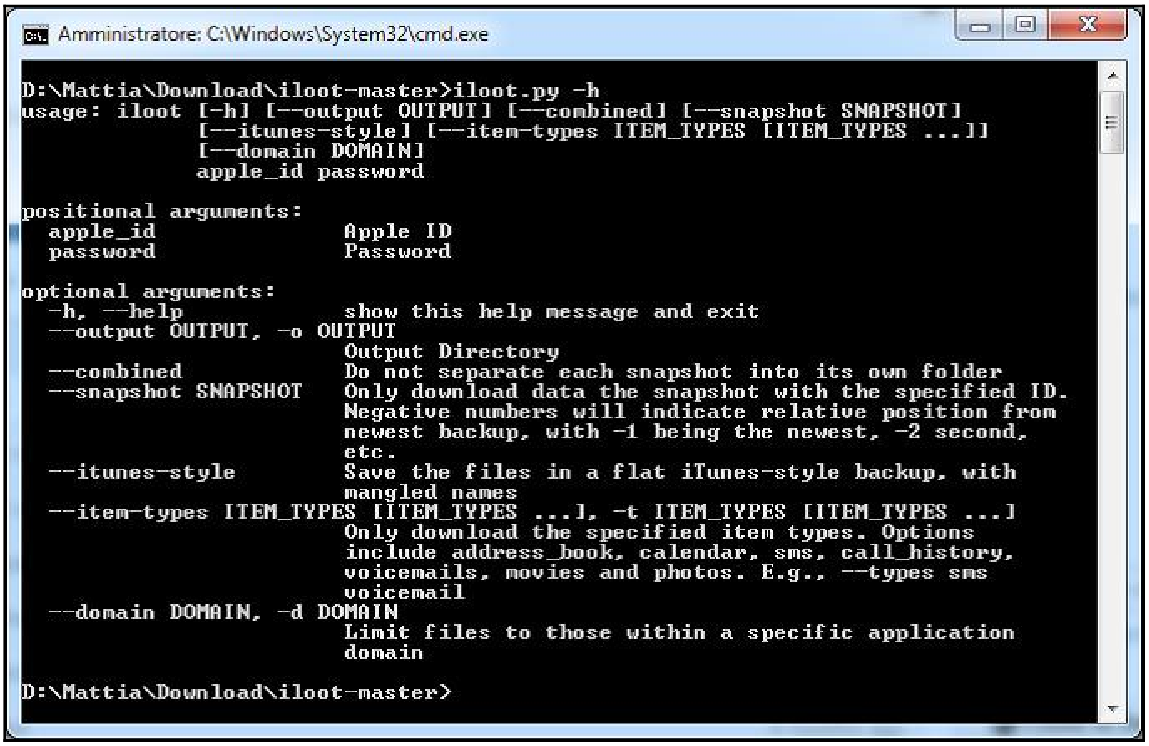
Path to MME token: /Volumes/MYCONTAINER/auth_tok/Accounts/

By clicking on **Extract**, the token is extracted in a `plist` file and can be used to access and backup files stored in iCloud.

Case study - iDevice backup acquisition with iLoot

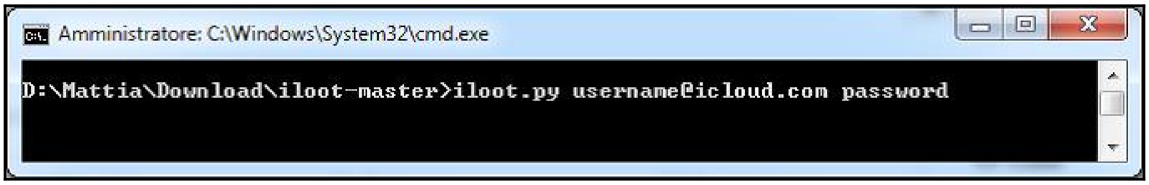
The same activity can be performed using the open source tool called iLoot (available at `https://github.com/hackappcom/iloot`), which works up to iOS 8. It requires Python and some dependencies. We suggest that you check out the website for the latest version and requirements.

By accessing the help (`iloot.py -h`), we can see the various available options. We can choose the output folder, if we want to download one specific snapshot, if we want the backup to be downloaded in the original iTunes format or with the domain-style folders, or if we want to download only specific information (for example, call history, SMS, photos, and so on) or only a specific domain, as shown here:



```
D:\Mattia\Download\iloot-master>iloot.py -h
usage: iloot [-h] [--output OUTPUT] [--combined] [--snapshot SNAPSHOT]
             [--itunes-style] [--item-types ITEM_TYPES [ITEM_TYPES ...]]
             [--domain DOMAIN]
             apple_id password

positional arguments:
  apple_id              Apple ID
  password              Password

optional arguments:
  -h, --help            show this help message and exit
  --output OUTPUT, -o OUTPUT
                        Output Directory
  --combined            Do not separate each snapshot into its own folder
  --snapshot SNAPSHOT   Only download data the snapshot with the specified ID.
                        Negative numbers will indicate relative position from
                        newest backup, with -1 being the newest, -2 second,
                        etc.
  --itunes-style        Save the files in a flat iTunes-style backup, with
                        mangled names
  --item-types ITEM_TYPES [ITEM_TYPES ...], -t ITEM_TYPES [ITEM_TYPES ...]
                        Only download the specified item types. Options
                        include address_book, calendar, sms, call_history,
                        voicemails, movies and photos. E.g., --types sms
                        voicemail
  --domain DOMAIN, -d DOMAIN
                        Limit files to those within a specific application
                        domain

D:\Mattia\Download\iloot-master>
```

To download a backup, you only need to insert the account credentials, as shown in the following screenshot:



```
D:\Mattia\Download\iloot-master>iloot.py username@icloud.com password
```

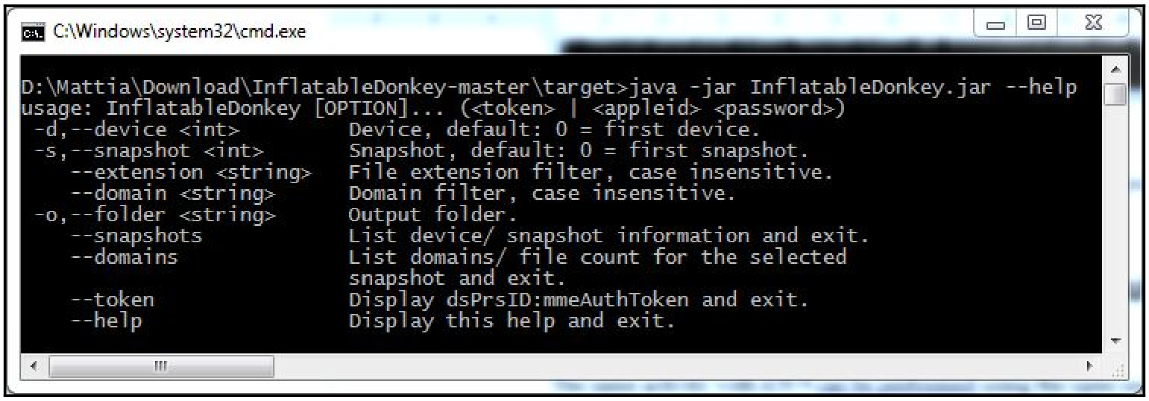
At the end of the process, you will find the backup in the output folder (the default folder's name is `/output`).

Case study - iDevice backup acquisition with InflatableDonkey

The same activity on iOS 9 can be performed using an open source tool called InflatableDonkey (available at `https://github.com/horrorho/InflatableDonkey`). It requires Java SDK and Maven. We suggest that you check out the website for the latest version and requirements.

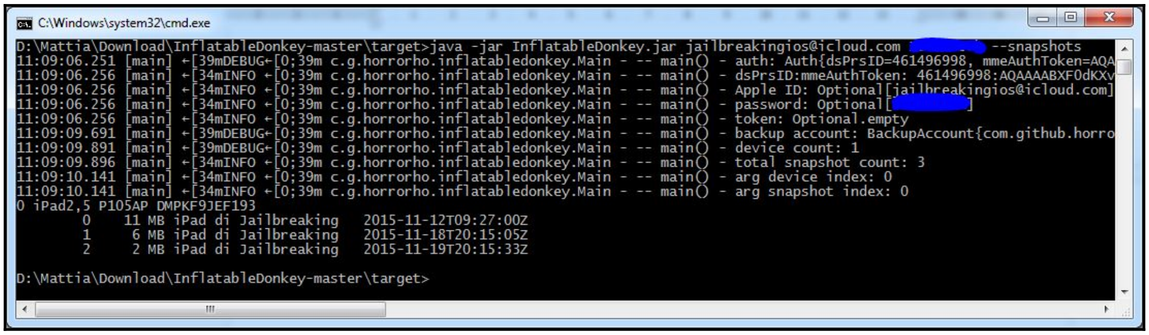
At the time of writing this book, the tool is still a PoC, so test it before using in real investigations.

By accessing the help (`java -jar InflatableDonkey.jar --help`), we can see the various available options. We can choose the device that we want to access and the specific backup snapshot we want to download, apply a file extension or domain-based filter, request the list of available devices and snapshots, and define the output folder, as shown here:


C:\Windows\system32\cmd.exe

```
D:\Mattia\Download\InflatableDonkey-master\target>java -jar InflatableDonkey.jar --help
usage: InflatableDonkey [OPTION]... (<token> | <appleid> <password>)
 -d,--device <int>          Device, default: 0 = first device.
 -s,--snapshot <int>        Snapshot, default: 0 = first snapshot.
    --extension <string>    File extension filter, case insensitive.
    --domain <string>       Domain filter, case insensitive.
 -o,--folder <string>       Output folder.
    --snapshots             List device/ snapshot information and exit.
    --domains               List domains/ file count for the selected
                            snapshot and exit.
    --token                 Display dsPrsID:mmeAuthToken and exit.
    --help                  Display this help and exit.
```

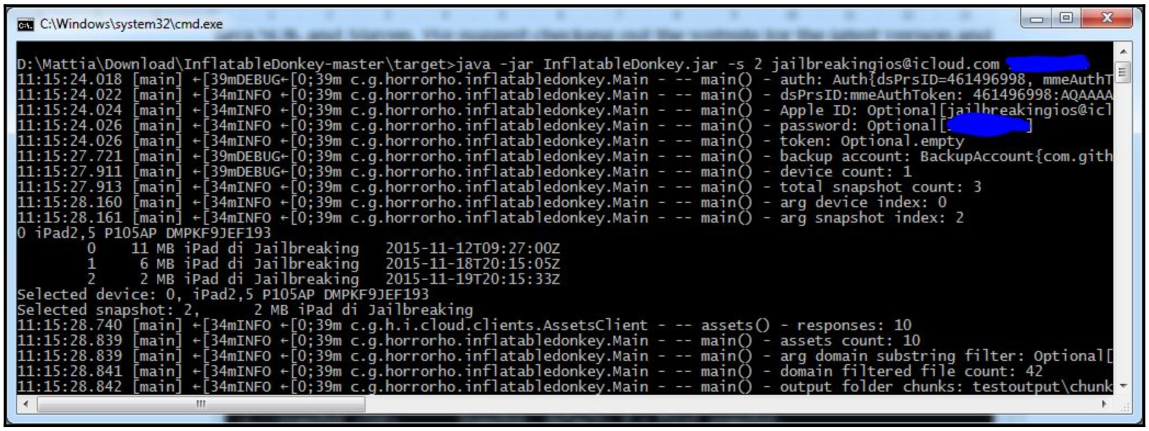
We can list the available devices and snapshots by providing the valid iCloud username and password, and the `--snapshots` option:



C:\Windows\system32\cmd.exe

```
D:\Mattia\Download\InflatableDonkey-master\target>java -jar InflatableDonkey.jar jailbreakingios@icloud.com [redacted] --snapshots
11:09:06.251 [main] ←[39mDEBUG←[0;39m c.g.horrorho.inflatabledonkey.Main - -- main() - auth: Auth{dsPrsID=461496998, mmeAuthToken=AQA
11:09:06.256 [main] ←[34mINFO ←[0;39m c.g.horrorho.inflatabledonkey.Main - -- main() - dsPrsID:mmeAuthToken: 461496998:AQAAAABXF0dKXv
11:09:06.256 [main] ←[34mINFO ←[0;39m c.g.horrorho.inflatabledonkey.Main - -- main() - Apple ID: Optional[jailbreakingios@icloud.com]
11:09:06.256 [main] ←[34mINFO ←[0;39m c.g.horrorho.inflatabledonkey.Main - -- main() - password: Optional[[redacted]]
11:09:06.256 [main] ←[34mINFO ←[0;39m c.g.horrorho.inflatabledonkey.Main - -- main() - token: Optional.empty
11:09:09.691 [main] ←[39mDEBUG←[0;39m c.g.horrorho.inflatabledonkey.Main - -- main() - backup account: BackupAccount{com.github.horro
11:09:09.891 [main] ←[39mDEBUG←[0;39m c.g.horrorho.inflatabledonkey.Main - -- main() - device count: 1
11:09:09.896 [main] ←[34mINFO ←[0;39m c.g.horrorho.inflatabledonkey.Main - -- main() - total snapshot count: 3
11:09:10.141 [main] ←[34mINFO ←[0;39m c.g.horrorho.inflatabledonkey.Main - -- main() - arg device index: 0
11:09:10.141 [main] ←[34mINFO ←[0;39m c.g.horrorho.inflatabledonkey.Main - -- main() - arg snapshot index: 0
0 iPad2,5 P105AP DMPKF9JEF193
        0    11 MB iPad di Jailbreaking   2015-11-12T09:27:00Z
        1     6 MB iPad di Jailbreaking   2015-11-18T20:15:05Z
        2     2 MB iPad di Jailbreaking   2015-11-19T20:15:33Z

D:\Mattia\Download\InflatableDonkey-master\target>
```

To download the backup, you need to insert the account credentials and eventually choose the specific device and snapshots. In the example shown in the following screenshot, the second snapshot from the available device has been requested:



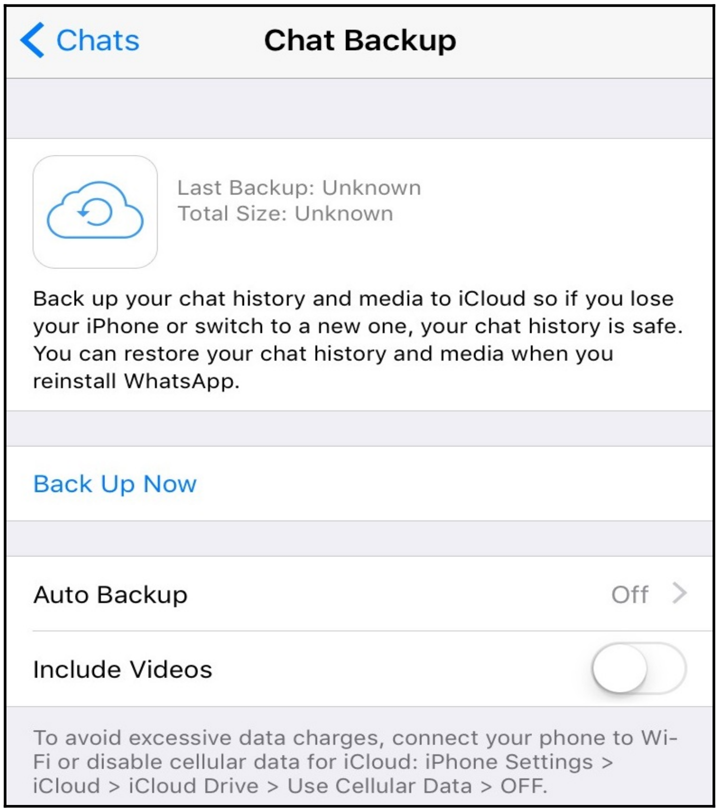
C:\Windows\system32\cmd.exe

```
D:\Mattia\Download\InflatableDonkey-master\target>java -jar InflatableDonkey.jar -s 2 jailbreakingios@icloud.com [redacted]
11:15:24.018 [main] ←[39mDEBUG←[0;39m c.g.horrorho.inflatabledonkey.Main - -- main() - auth: Auth{dsPrsID=461496998, mmeAuthT
11:15:24.022 [main] ←[34mINFO ←[0;39m c.g.horrorho.inflatabledonkey.Main - -- main() - dsPrsID:mmeAuthToken: 461496998:AQAAAA
11:15:24.024 [main] ←[34mINFO ←[0;39m c.g.horrorho.inflatabledonkey.Main - -- main() - Apple ID: Optional[jailbreakingios@icl
11:15:24.026 [main] ←[34mINFO ←[0;39m c.g.horrorho.inflatabledonkey.Main - -- main() - password: Optional[[redacted]]
11:15:24.026 [main] ←[34mINFO ←[0;39m c.g.horrorho.inflatabledonkey.Main - -- main() - token: Optional.empty
11:15:27.721 [main] ←[39mDEBUG←[0;39m c.g.horrorho.inflatabledonkey.Main - -- main() - backup account: BackupAccount{com.gith
11:15:27.911 [main] ←[39mDEBUG←[0;39m c.g.horrorho.inflatabledonkey.Main - -- main() - device count: 1
11:15:27.913 [main] ←[34mINFO ←[0;39m c.g.horrorho.inflatabledonkey.Main - -- main() - total snapshot count: 3
11:15:28.160 [main] ←[34mINFO ←[0;39m c.g.horrorho.inflatabledonkey.Main - -- main() - arg device index: 0
11:15:28.161 [main] ←[34mINFO ←[0;39m c.g.horrorho.inflatabledonkey.Main - -- main() - arg snapshot index: 2
0 iPad2,5 P105AP DMPKF9JEF193
        0    11 MB iPad di Jailbreaking   2015-11-12T09:27:00Z
        1     6 MB iPad di Jailbreaking   2015-11-18T20:15:05Z
        2     2 MB iPad di Jailbreaking   2015-11-19T20:15:33Z
Selected device: 0, iPad2,5 P105AP DMPKF9JEF193
Selected snapshot: 2,      2 MB iPad di Jailbreaking
11:15:28.740 [main] ←[34mINFO ←[0;39m c.g.h.i.cloud.clients.AssetsClient - -- assets() - responses: 10
11:15:28.839 [main] ←[34mINFO ←[0;39m c.g.horrorho.inflatabledonkey.Main - -- main() - assets count: 10
11:15:28.839 [main] ←[34mINFO ←[0;39m c.g.horrorho.inflatabledonkey.Main - -- main() - arg domain substring filter: Optional[
11:15:28.841 [main] ←[34mINFO ←[0;39m c.g.horrorho.inflatabledonkey.Main - -- main() - domain filtered file count: 42
11:15:28.842 [main] ←[34mINFO ←[0;39m c.g.horrorho.inflatabledonkey.Main - -- main() - output folder chunks: testoutput\chunk
```

At the end of the process, you will find the backup in the output folder (the default folder's name is `/testoutput`). It is important to highlight that it is not possible to download the backup in the original iTunes format, but only with the domain-style folders.

Case study - WhatsApp backup acquisition with Elcomsoft Explorer for WhatsApp

As already mentioned, the applications installed on your iOS device have the opportunity to backup their data to iCloud Drive. One of the applications that enables this feature is WhatsApp (navigate to **Settings** | **Chats** | **Chat Backup** within the application menu). You can manually make the backup (**Back Up Now**) or you can set the **Auto Backup** feature on a **Daily**, **Weekly**, or **Monthly** basis:



Last Backup: Unknown
Total Size: Unknown

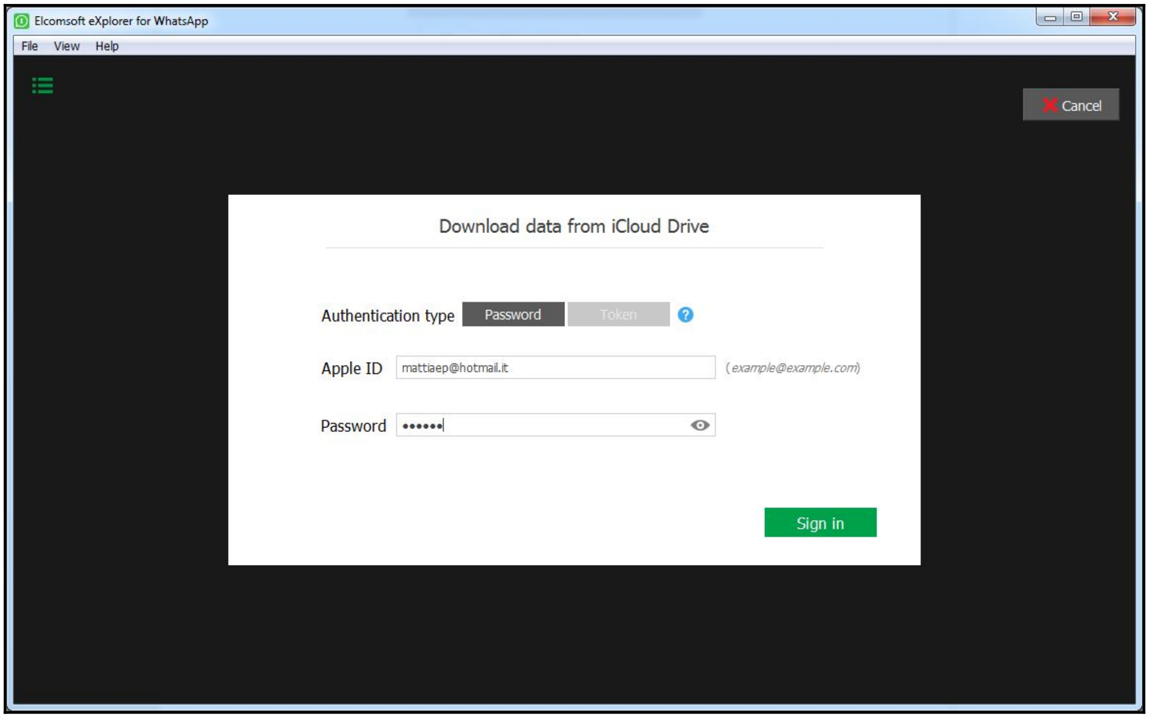
Back up your chat history and media to iCloud so if you lose your iPhone or switch to a new one, your chat history is safe. You can restore your chat history and media when you reinstall WhatsApp.

Auto Backup Off

Include Videos

To avoid excessive data charges, connect your phone to Wi-Fi or disable cellular data for iCloud: iPhone Settings > iCloud > iCloud Drive > Use Cellular Data > OFF.

Elcomsoft Explorer for WhatsApp allows you to extract WhatsApp backup data stored within a backup on iCloud or iCloud Drive. Like the EPPB software, it can access the data on iCloud through the authentication credentials (username and password) or through an authentication token:



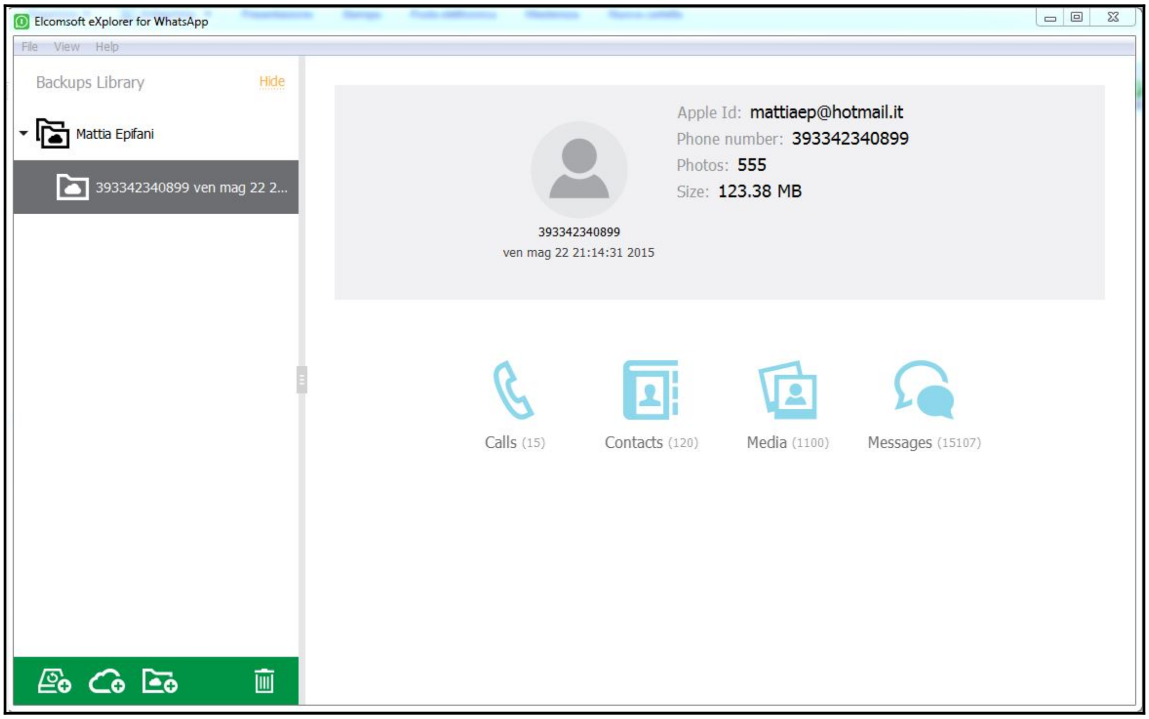
Download data from iCloud Drive

Authentication type [✓ Password] Token

Apple ID mattiaep@hotmail.it (example@example.com)

Password ••••••

Once downloaded, the data can be viewed directly within the software divided by type, **Calls**, **Contacts**, **Media**, and **Messages:**



Apple Id: mattiaep@hotmail.it
Phone number: 393342340899
Photos: 555
Size: 123.38 MB

393342340899
ven mag 22 21:14:31 2015

iCloud Control Panel artifacts on the computer

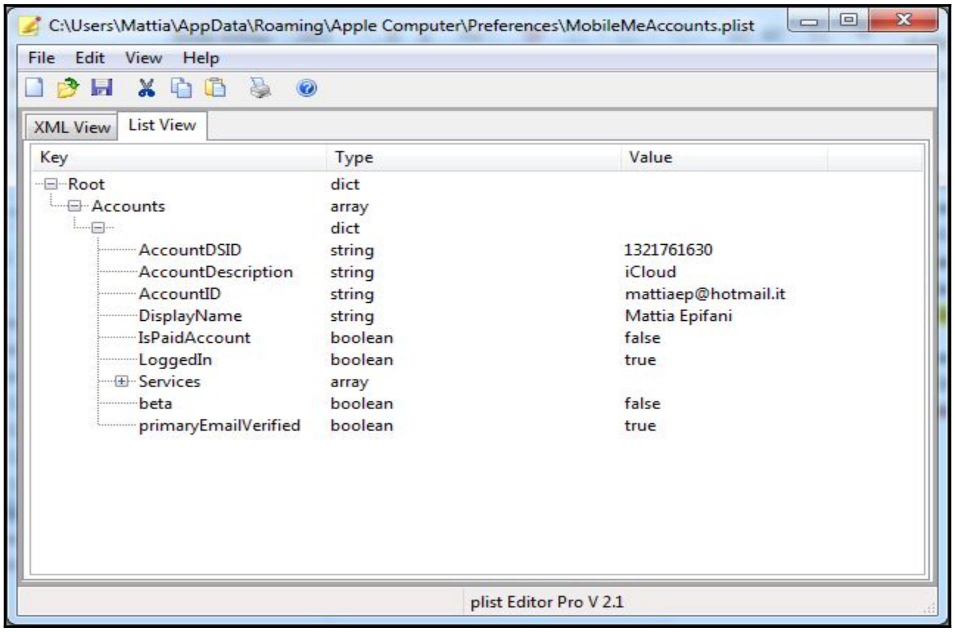
The installation of iCloud Control Panel, other than allowing the recovery of the user's authentication token, as shown previously, leaves logs of interest within the disk of the computer.

On a Windows Vista/7/8 system, the logs of the connections to the iCloud service are stored in `C:\Users\<username>\AppData\Roaming\Apple Computer\Logs`. To locate the logs of interest, it is necessary to search logs related to the executable `iCloud.exe` file within the text file. The files are named according to a standard format that includes the date and time at which the service started (for example, `asl.113048_18Apr16.log`), thus letting the analyst create a timeline of iCloud usage.

The user information configured in the iCloud Control Panel software is stored in the `C:\Users\<username>\AppData\Roaming\Apple Computer\Preferences\mobilemeaccounts.plist` file.

In particular, the following user information is there in the file:

- `AccountDSID`: This key denotes user identification
- `AccountID`: This key denotes the iCloud account username
- `DisplayName`: This key denotes the displayed name set by the account owner
- `IsPaidAccount`: This key is set to `True` if the user has purchased additional services from Apple (more storage on iCloud)
- `LoggedIn`: This key denotes whether the user is automatically logged in or not to the service


C:\Users\Mattia\AppData\Roaming\Apple Computer\Preferences\MobileMeAccounts.plist
File
Edit
View
Help
XML View
List View
plist Editor Pro V 2.1

Key	Type	Value
Root	dict	
Accounts	array	
	dict	
AccountDSID	string	1321761630
AccountDescription	string	iCloud
AccountID	string	mattiaep@hotmail.it
DisplayName	string	Mattia Epifani
IsPaidAccount	boolean	false
LoggedIn	boolean	true
Services	array	
beta	boolean	false
primaryEmailVerified	boolean	true

On a Mac OS X system, the user information is stored instead in the `/Users/<user>/Library/Preferences/MobileMeAccounts.plist` file.

You will also find plenty of the `asl` logs (the Apple system logs), so, in order to check a user's iCloud activity, you will have to parse the following log files:

- `/private/var/log/asl/YYYY.MM.DD.UID.asl`
- `/private/var/log/system.log`

For a detailed analysis of iCloud artifacts on a Mac OS X computer we suggest you to read the presentation *Ubiquity Forensics – Your iCloud and You* available at `http://www.mac4n6.com/resources/`.

Acquiring data from Cloud with stored tokens

In `Chapter 4`, *Evidence Acquisition and Analysis from iTunes Backup*, we saw how an encrypted backup contains an application's login credentials stored in the Keychain. Depending on the application, it is possible to directly find the password, as in the case of the native mail, or an authentication token, as in the case of many third-party applications such as Facebook and Twitter.

Therefore, knowing the password of the backup (either because it was specifically set in order to extract the credentials, or because it was cracked) you can extract the credentials of an application and use them to access the user cloud services directly on various service portals. In this way, for applications that store the token, it is not necessary to know the user's password for the specific service, but the access is guaranteed by the same token.

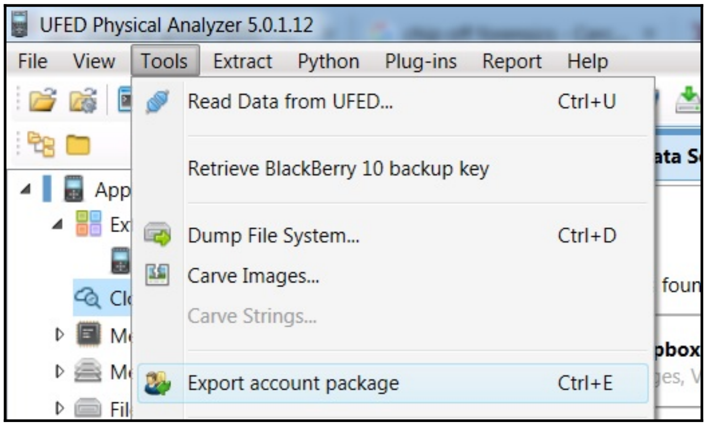
Several commercial mobile forensics solutions are integrating this functionality within their software or additional software to be used in combination with their forensic tool. Among the best known, those worth mentioning are UFED Cloud Analyzer, Oxygen Forensic Detective, and Paraben Device Seizure.

It should be stressed that this type of software makes remote access to various servers and that data is not physically stored within the analyzed device. For this, we need to consider at least two aspects: to be sure to have adequate legal authority to do this activity and, in any case, to bear in mind that this type of acquisition will leave traces and may generate notifications for the owner. For these reasons, we must take particular care when using it.

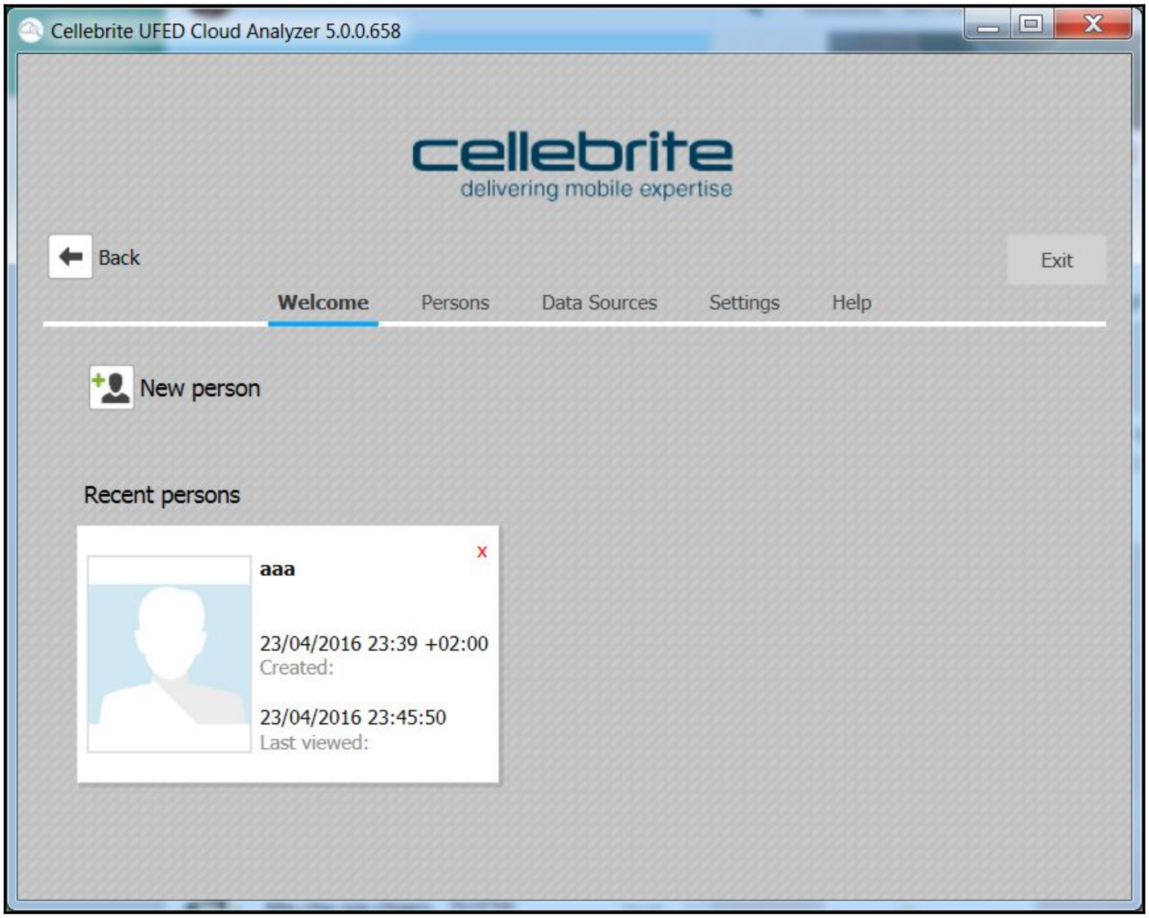
Case study - Cloud data acquisition with UFED Cloud Analyzer

UFED Cloud Analyzer is a commercial software developed by Cellebrite as completion of forensic suites for the acquisition and analysis of mobile devices. This software allows you to access your data on different platforms: Google (Gmail, Contacts, Drive, Location History, and Search History), Dropbox, Facebook, Twitter, Instagram, Kik, and Vk.

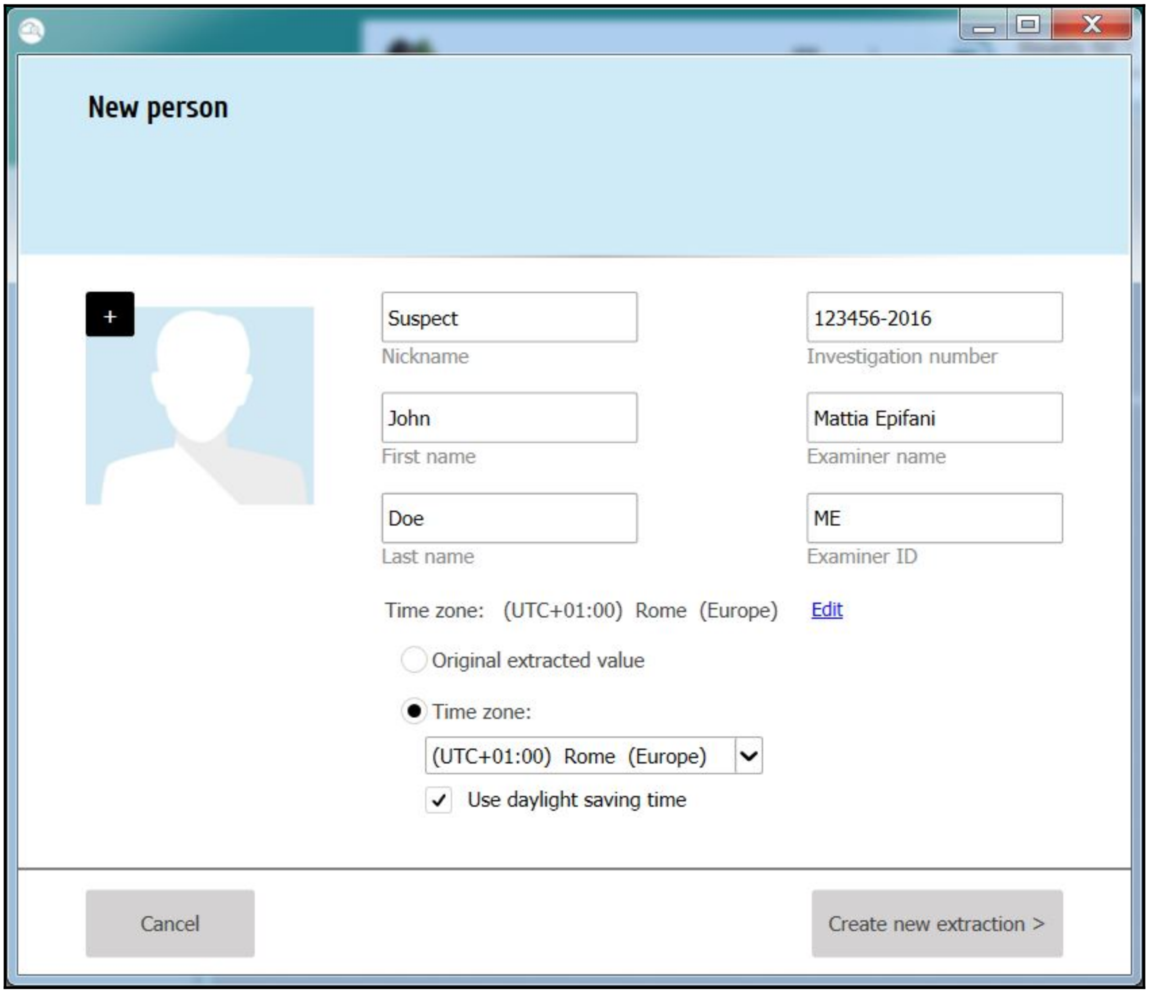
The software can take login credentials as manual input (username and password) or by importing an account package exported using the UFED Physical Analyzer software. The following screenshot shows the export procedure:


UFED Physical Analyzer 5.0.1.12
File
View
Tools
Extract
Python
Plug-ins
Report
Help
Read Data from UFED... Ctrl+U
Retrieve BlackBerry 10 backup key
Dump File System... Ctrl+D
Carve Images...
Carve Strings...
Export account package Ctrl+E

The starting screen of the software allows you to view the already existing persons or create a new one. It refers to it as person, since we are talking about users' personal accounts:



When inserting a new person, you can specify their identification information and the time zone of reference, as shown here:



New person

Nickname: Suspect	Investigation number: 123456-2016
First name: John	Examiner name: Mattia Epifani
Last name: Doe	Examiner ID: ME

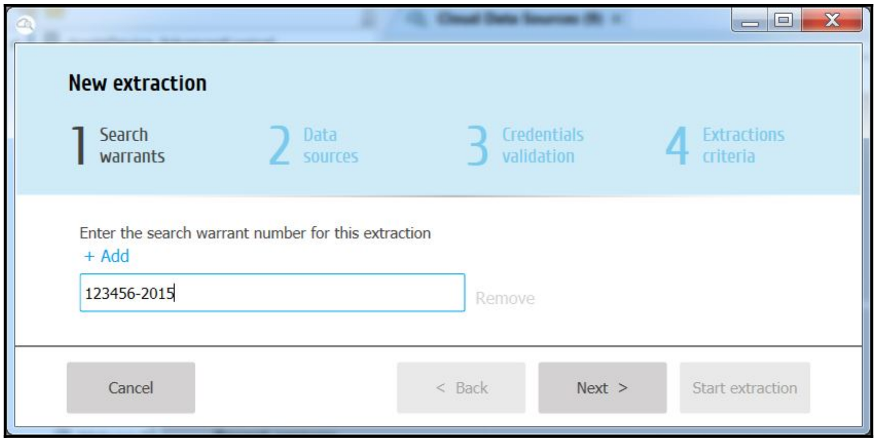
Time zone: (UTC+01:00) Rome (Europe) Edit

( ) Original extracted value

(●) Time zone: (UTC+01:00) Rome (Europe)

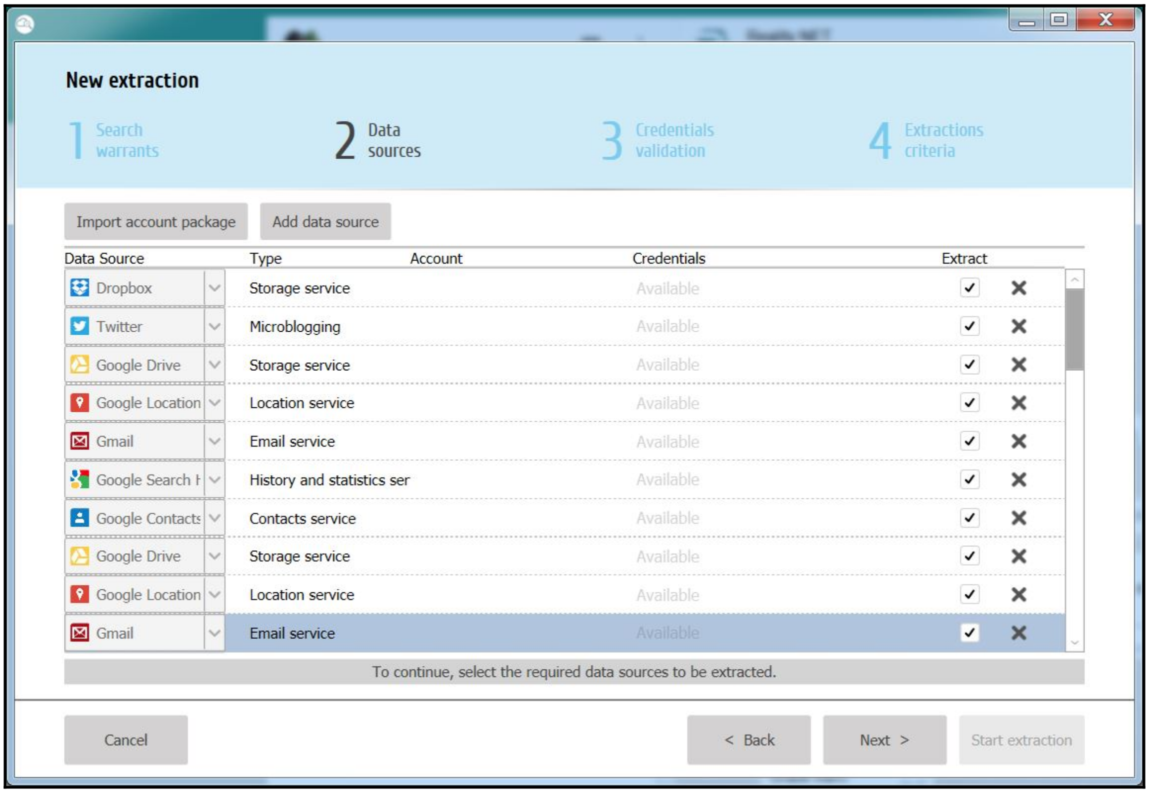
✓ Use daylight saving time

It is then possible to proceed with the extraction, providing even the related legal information (for example, Search warrant):



Enter the search warrant number for this extraction
+ Add
123456-2015 Remove

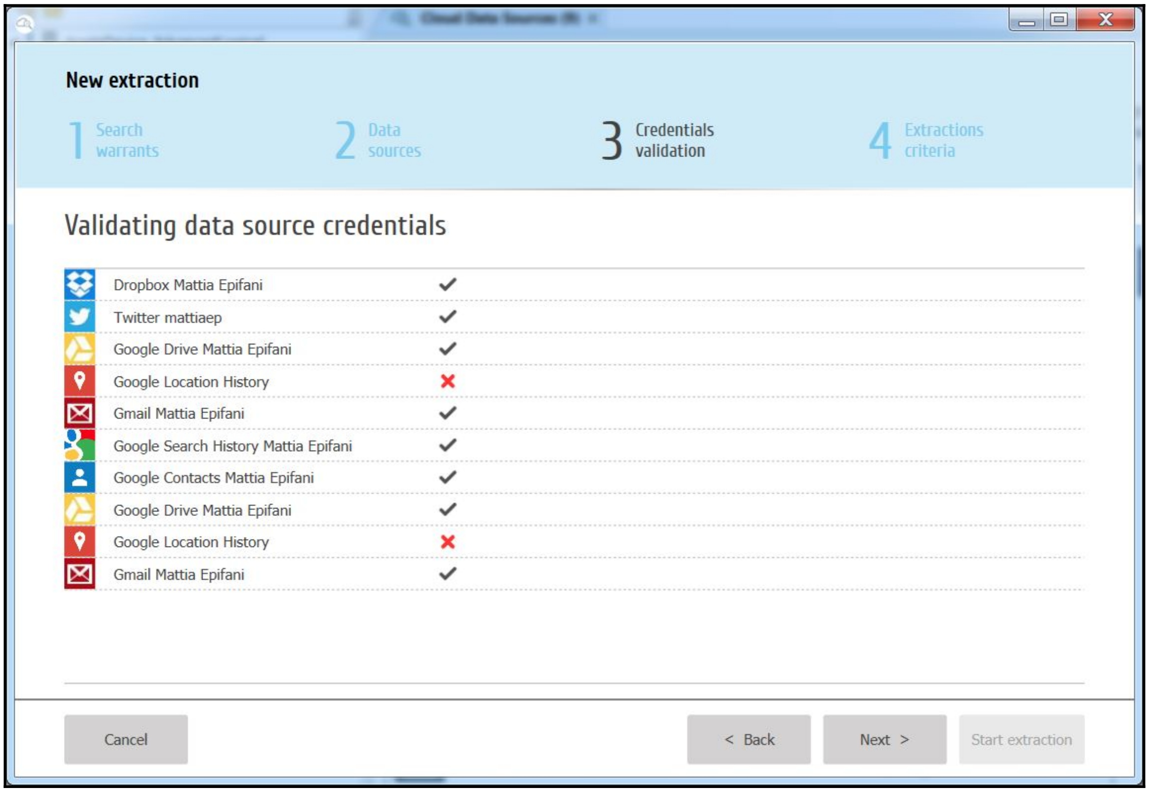
It is then necessary to select which account, among those available within the account package, you want to acquire. In the following example, credentials for Dropbox, Twitter, and Google are available:



Data Source	Type	Account	Credentials	Extract	
Dropbox	Storage service		Available	✓	✕
Twitter	Microblogging		Available	✓	✕
Google Drive	Storage service		Available	✓	✕
Google Location	Location service		Available	✓	✕
Gmail	Email service		Available	✓	✕
Google Search H	History and statistics ser		Available	✓	✕
Google Contacts	Contacts service		Available	✓	✕
Google Drive	Storage service		Available	✓	✕
Google Location	Location service		Available	✓	✕
Gmail	Email service		Available	✓	✕

To continue, select the required data sources to be extracted.

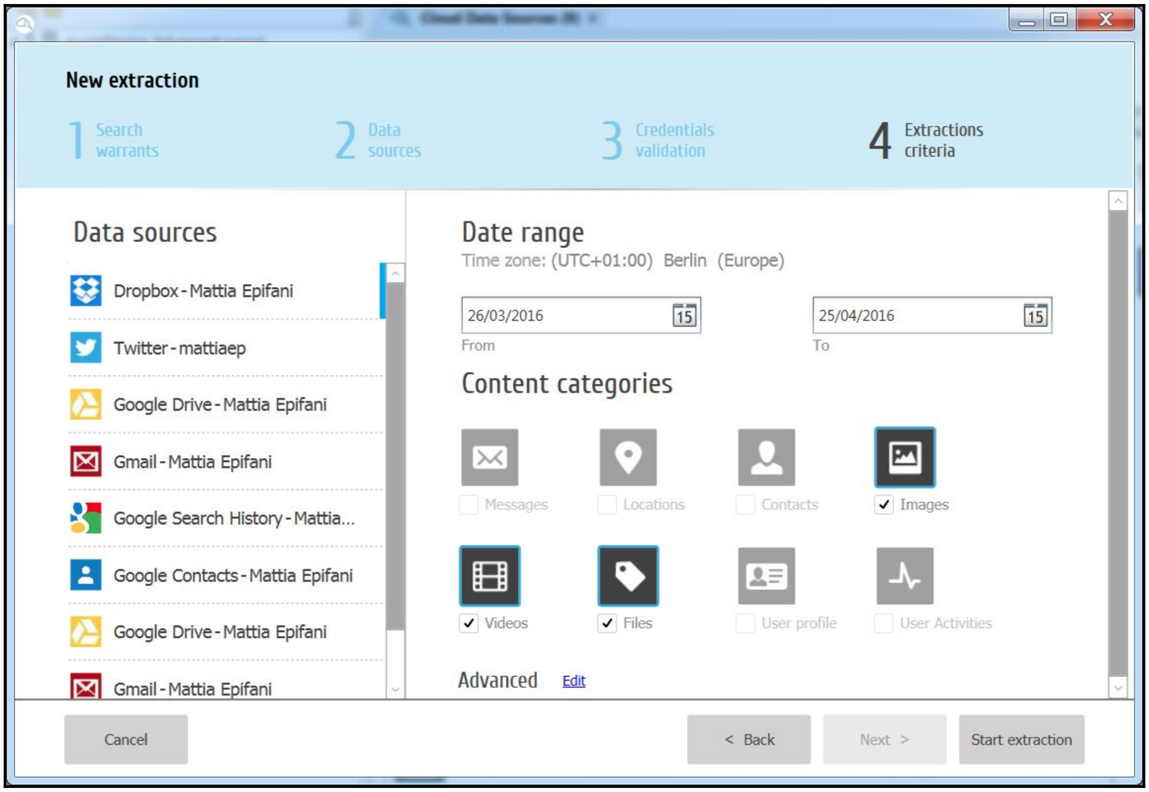
The user's credentials are then authenticated to verify the correctness and validity:



Validating data source credentials

Dropbox Mattia Epifani	✓
Twitter mattiaep	✓
Google Drive Mattia Epifani	✓
Google Location History	✗
Gmail Mattia Epifani	✓
Google Search History Mattia Epifani	✓
Google Contacts Mattia Epifani	✓
Google Drive Mattia Epifani	✓
Google Location History	✗
Gmail Mattia Epifani	✓

Finally, it is also possible to specify the parameters for each account to be acquired, such as the type of content (for example, **Messages**, **Contacts**, **Images**, **Videos**, and so on) and the timeframe of interest. Once such options have been selected, it is possible to start extracting data:


New extraction

1 Search warrants — 2 Data sources — 3 Credentials validation — 4 Extractions criteria

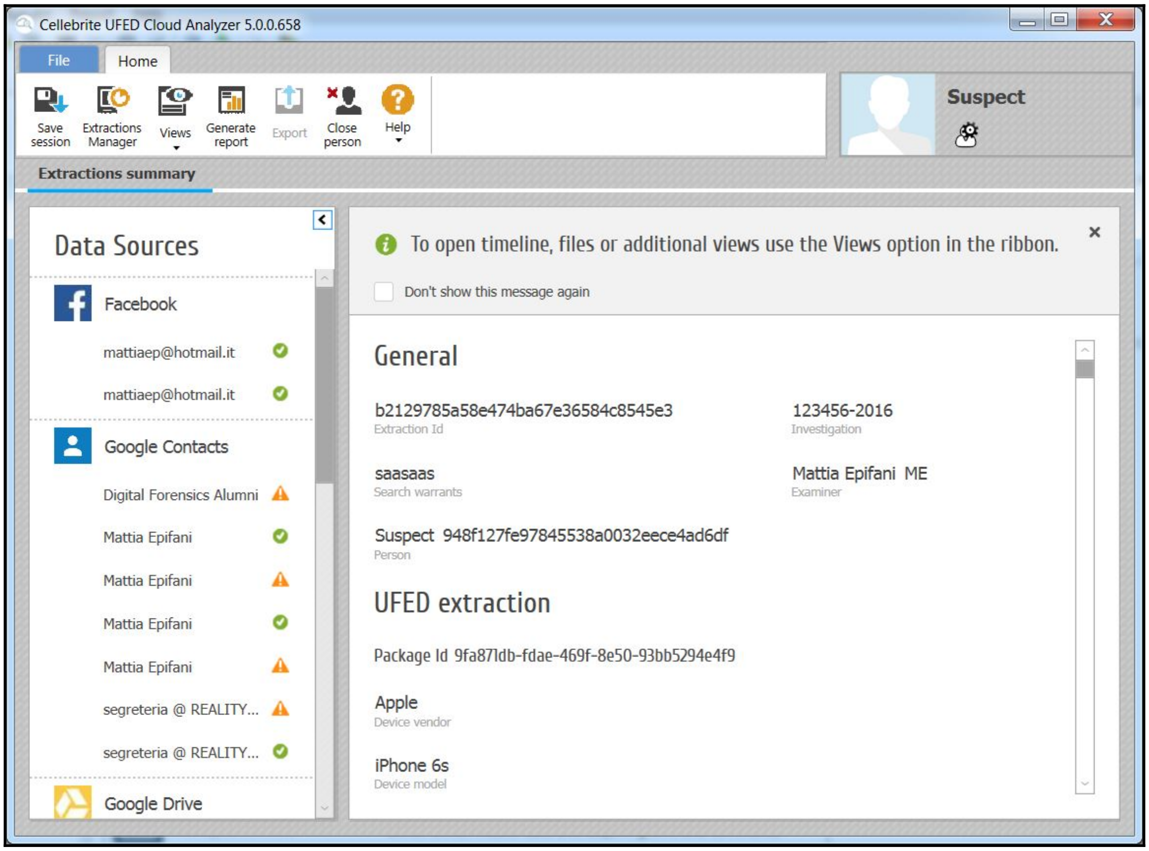
Data sources
- Dropbox - Mattia Epifani
- Twitter - mattiaep
- Google Drive - Mattia Epifani
- Gmail - Mattia Epifani
- Google Search History - Mattia...
- Google Contacts - Mattia Epifani
- Google Drive - Mattia Epifani
- Gmail - Mattia Epifani

Date range
Time zone: (UTC+01:00) Berlin (Europe)
From: 26/03/2016
To: 25/04/2016

Content categories
- [ ] Messages
- [ ] Locations
- [ ] Contacts
- [✓] Images
- [✓] Videos
- [✓] Files
- [ ] User profile
- [ ] User Activities

Advanced Edit

Once the extraction is completed, it shows a summary of the acquired data, broken down for each account:



mattiaep@hotmail.it ✓
mattiaep@hotmail.it ✓
Digital Forensics Alumni ⚠
Mattia Epifani ✓
Mattia Epifani ⚠
Mattia Epifani ✓
Mattia Epifani ⚠
segreteria @ REALITY... ⚠
segreteria @ REALITY... ✓

To open timeline, files or additional views use the Views option in the ribbon.

Don't show this message again

General

b2129785a58e474ba67e36584c8545e3 Extraction Id	123456-2016 Investigation
saasaas Search warrants	Mattia Epifani ME Examiner
Suspect 948f127fe97845538a0032eece4ad6df Person	

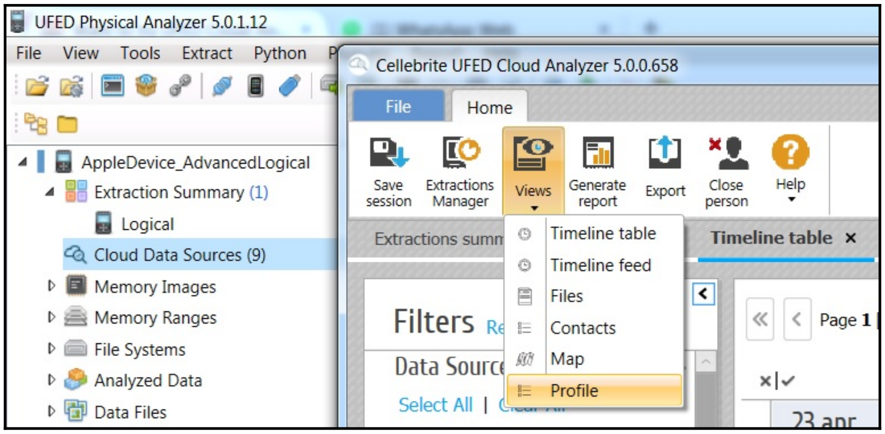
UFED extraction

Package Id 9fa871db-fdae-469f-8e50-93bb5294e4f9

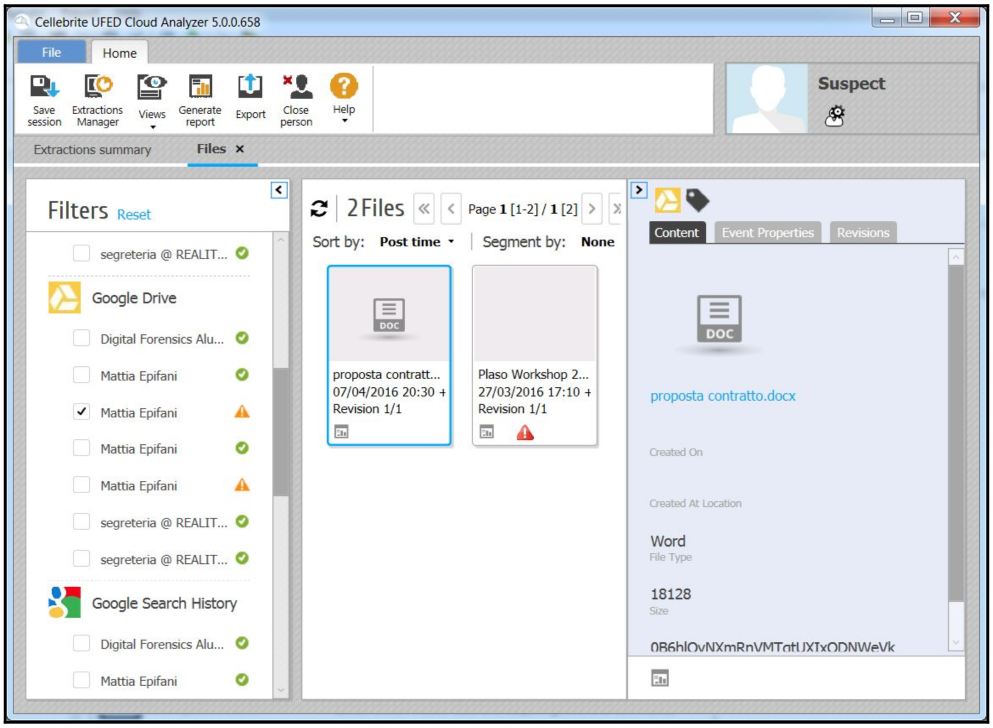
Apple
Device vendor

iPhone 6s
Device model

The software also allows filtering and searching the data by offering different types of visualization, such as **Timeline table**, **Timeline feed**, **Files**, **Contacts**, **Map**, and **Profile**:



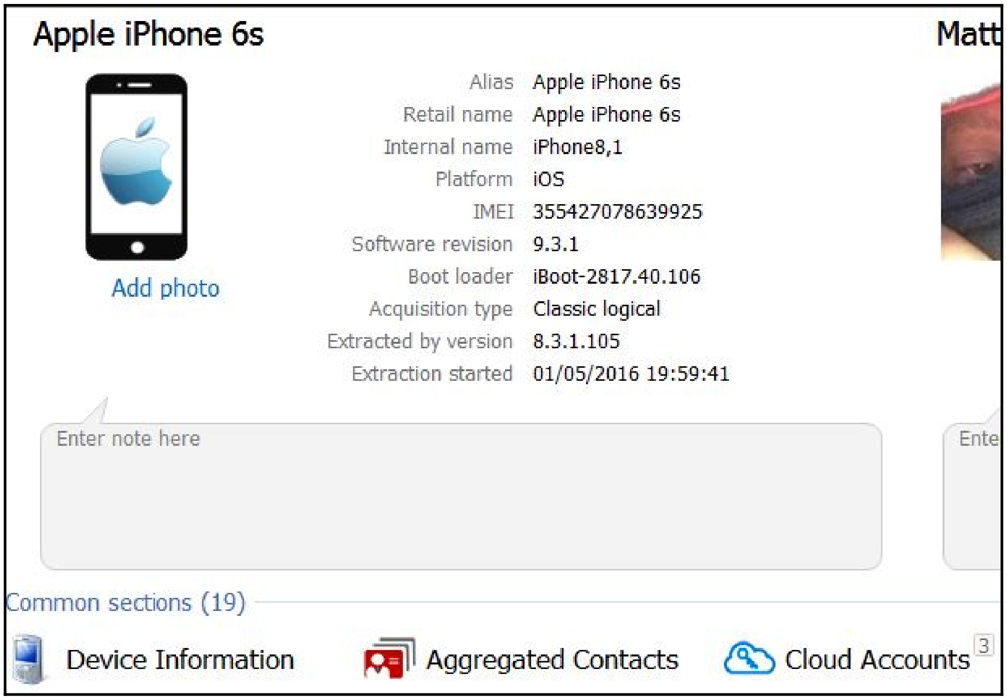
The following screenshot shows an example of the file visualization of a **Google Drive** account:



Case study - cloud data acquisition with Oxygen Forensic Detective

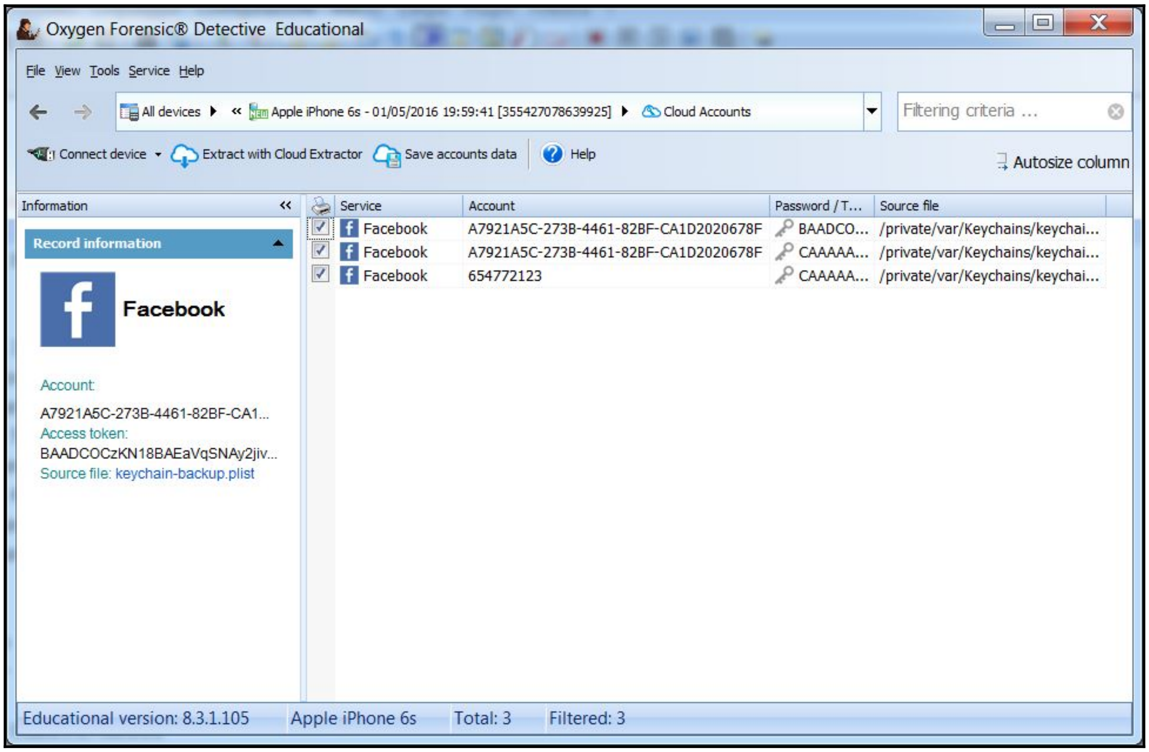
Oxygen Forensic Detective is a commercial software developed by Oxygen Forensic and is the most complete version among those available from this vendor. In particular, in addition to extraction and analysis capabilities present in the Analyst version, this version includes specific features to enable access of the data stored on the Cloud. The software allows you to access your data on different platforms: Google (Calendar Bookmarks, History, Contacts, Drive, Location History, Mail, Photos, Tasks, and Backup WhatsApp), iCloud (Calendars, Contacts, and iOS Backup), Live (Calendars, Contacts, and OneDrive), Box, Dropbox, Facebook, Twitter, Instagram, and Swarm (Foursquare).

The software can take login credentials as manual input (username and password) or by importing an Oxygen Credentials Package exported using the Oxygen Forensic Detective software itself. The following screenshot shows an example of the acquisition of an **Apple iPhone 6S** with iOS **9.3.1**. It's important to remember that for the extraction of credentials, encrypting the backup is necessary:


Apple iPhone 6s

Alias	Apple iPhone 6s
Retail name	Apple iPhone 6s
Internal name	iPhone8,1
Platform	iOS
IMEI	355427078639925
Software revision	9.3.1
Boot loader	iBoot-2817.40.106
Acquisition type	Classic logical
Extracted by version	8.3.1.105
Extraction started	01/05/2016 19:59:41

The authentication credentials can be viewed by accessing the specific account section. For each account, the password or authentication token and the input file of the credentials are given. In the specific example here, the credentials were related to a Facebook account:


Oxygen Forensic® Detective Educational
File View Tools Service Help
All devices « Apple iPhone 6s - 01/05/2016 19:59:41 [355427078639925] Cloud Accounts
Filtering criteria ...
Connect device
Extract with Cloud Extractor
Save accounts data
Help
Autosize column
Information
Record information

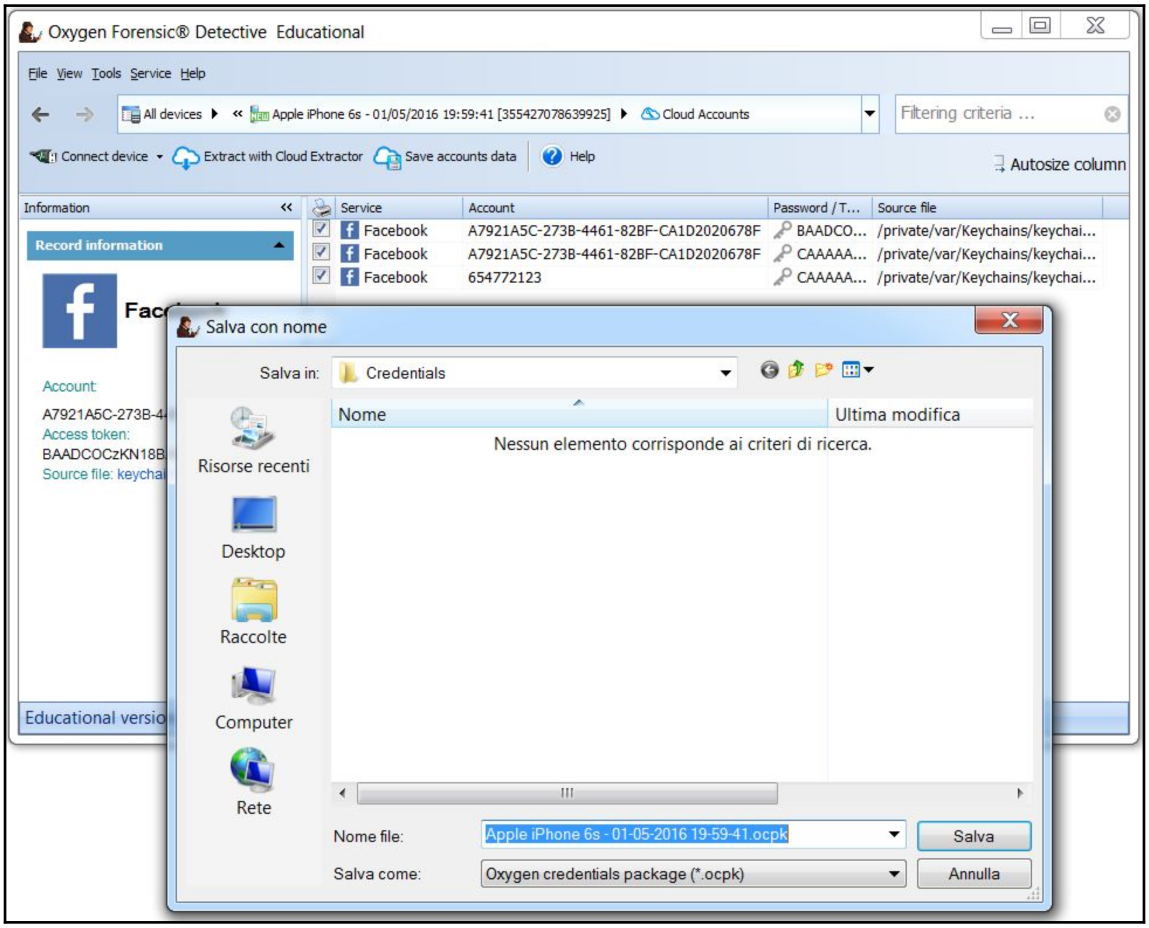
Facebook

Account:
A7921A5C-273B-4461-82BF-CA1...
Access token:
BAADCOCzKN18BAEaVqSNAy2jiv...
Source file: keychain-backup.plist

Service	Account	Password / T...	Source file
Facebook	A7921A5C-273B-4461-82BF-CA1D2020678F	BAADCO...	/private/var/Keychains/keychai...
Facebook	A7921A5C-273B-4461-82BF-CA1D2020678F	CAAAAA...	/private/var/Keychains/keychai...
Facebook	654772123	CAAAAA...	/private/var/Keychains/keychai...

Educational version: 8.3.1.105 Apple iPhone 6s Total: 3 Filtered: 3

Using the **Save Accounts Date** option you can export the credentials in the Oxygen Credentials Package format while saving the files on the computer:



Oxygen Forensic® Detective Educational

All devices « Apple iPhone 6s - 01/05/2016 19:59:41 [355427078639925] Cloud Accounts

Service	Account	Password / T...	Source file
Facebook	A7921A5C-273B-4461-82BF-CA1D2020678F	BAADCO...	/private/var/Keychains/keychai...
Facebook	A7921A5C-273B-4461-82BF-CA1D2020678F	CAAAAA...	/private/var/Keychains/keychai...
Facebook	654772123	CAAAAA...	/private/var/Keychains/keychai...

Salva con nome

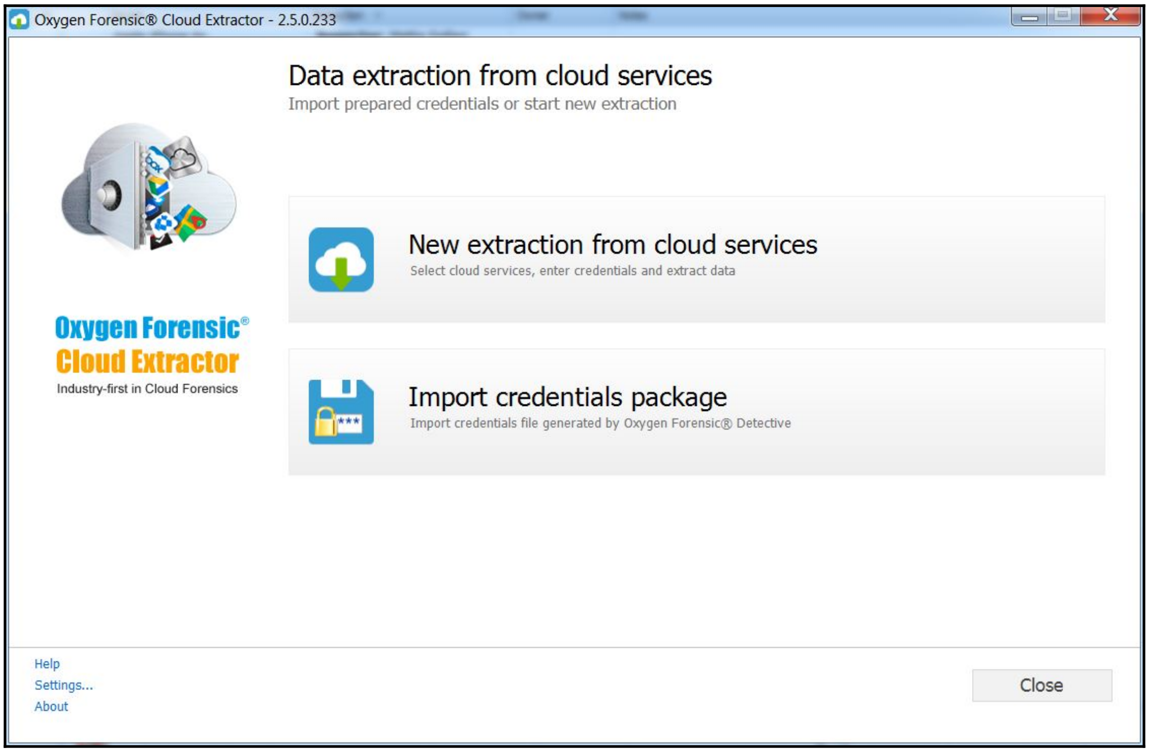
Salva in: Credentials

Nessun elemento corrisponde ai criteri di ricerca.

Nome file: Apple iPhone 6s - 01-05-2016 19-59-41.ocpk

Salva come: Oxygen credentials package (*.ocpk)

Once you have your credentials package from the main screen of the software, simply click on the **Cloud** button at the top right to access the setup wizard:



Oxygen Forensic® Cloud Extractor - 2.5.0.233

Data extraction from cloud services

Import prepared credentials or start new extraction

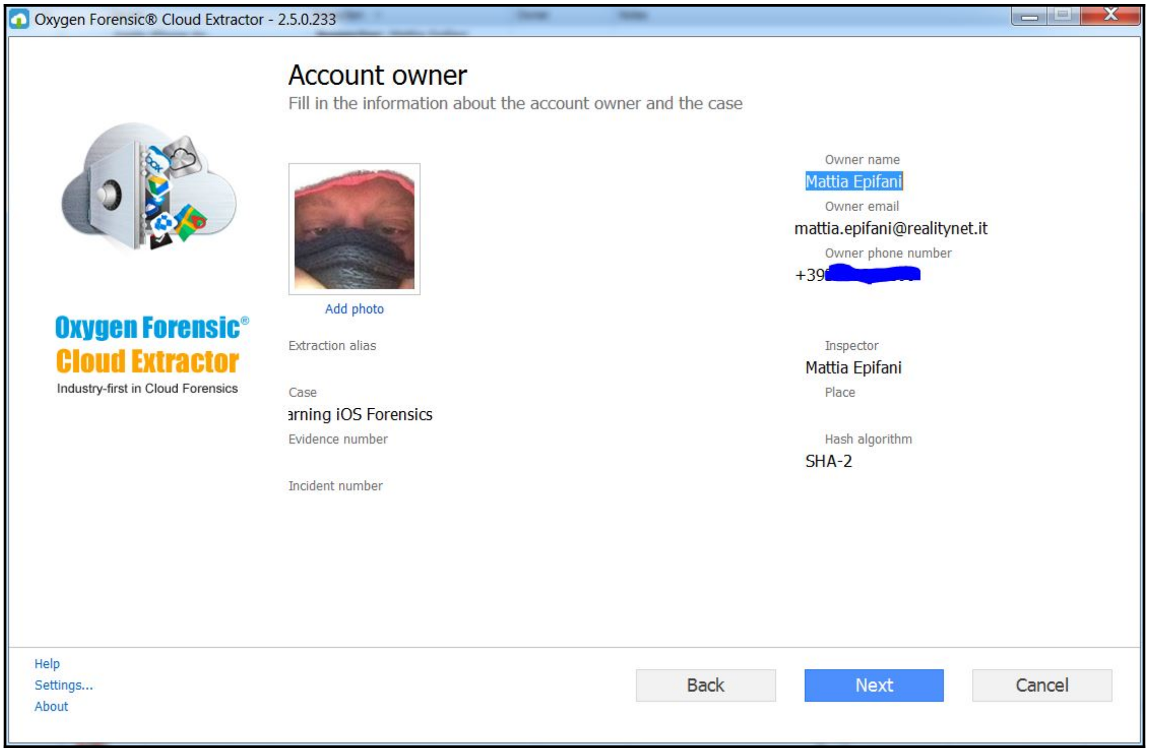
New extraction from cloud services

Select cloud services, enter credentials and extract data

Import credentials package

Import credentials file generated by Oxygen Forensic® Detective

In order to proceed, you must specify the **Import credentials package** option and select the exported file from the previous step. It then displays a view about the user, typically containing name and surname, profile photo, e-mail address, and telephone number, as shown here:


Oxygen Forensic® Cloud Extractor - 2.5.0.233

Account owner

Fill in the information about the account owner and the case

Owner name
Mattia Epifani
Owner email
mattia.epifani@realitynet.it
Owner phone number
+39[redacted]

Add photo

Extraction alias

Case
arning iOS Forensics

Evidence number

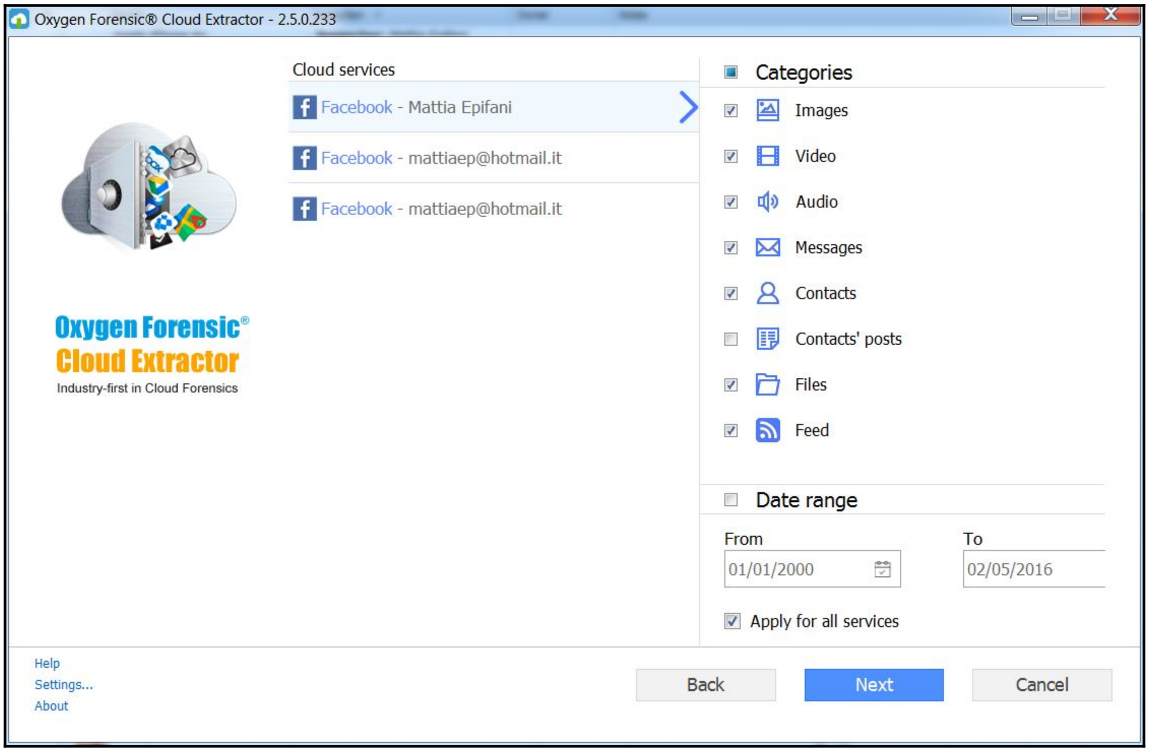
Incident number

Inspector
Mattia Epifani

Place

Hash algorithm
SHA-2

It is then possible to select which account you want to access, verify the correctness of the credentials, and set each of the extraction parameters. In the specific case of Facebook, you can specify which content you want to extract (**Images**, **Video**, **Audio**, **Messages**, **Contacts**, **Files**, and so on) and the timeframe of interest:


Oxygen Forensic® Cloud Extractor - 2.5.0.233

Cloud services
- Facebook - Mattia Epifani
- Facebook - mattiaep@hotmail.it
- Facebook - mattiaep@hotmail.it

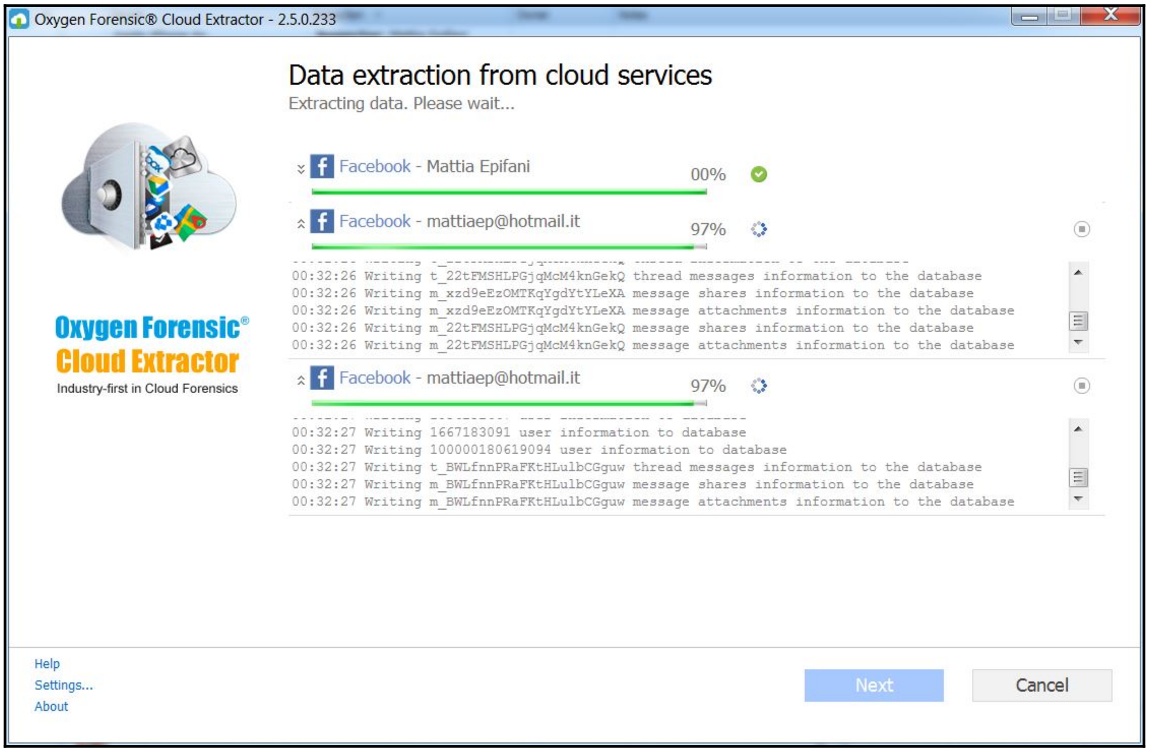
Categories
- [✓] Images
- [✓] Video
- [✓] Audio
- [✓] Messages
- [✓] Contacts
- [ ] Contacts' posts
- [✓] Files
- [✓] Feed

[ ] Date range
From 01/01/2000 To 02/05/2016

[✓] Apply for all services

Oxygen Forensic® Cloud Extractor
Industry-first in Cloud Forensics
Help
Settings...
About
Back
Next
Cancel

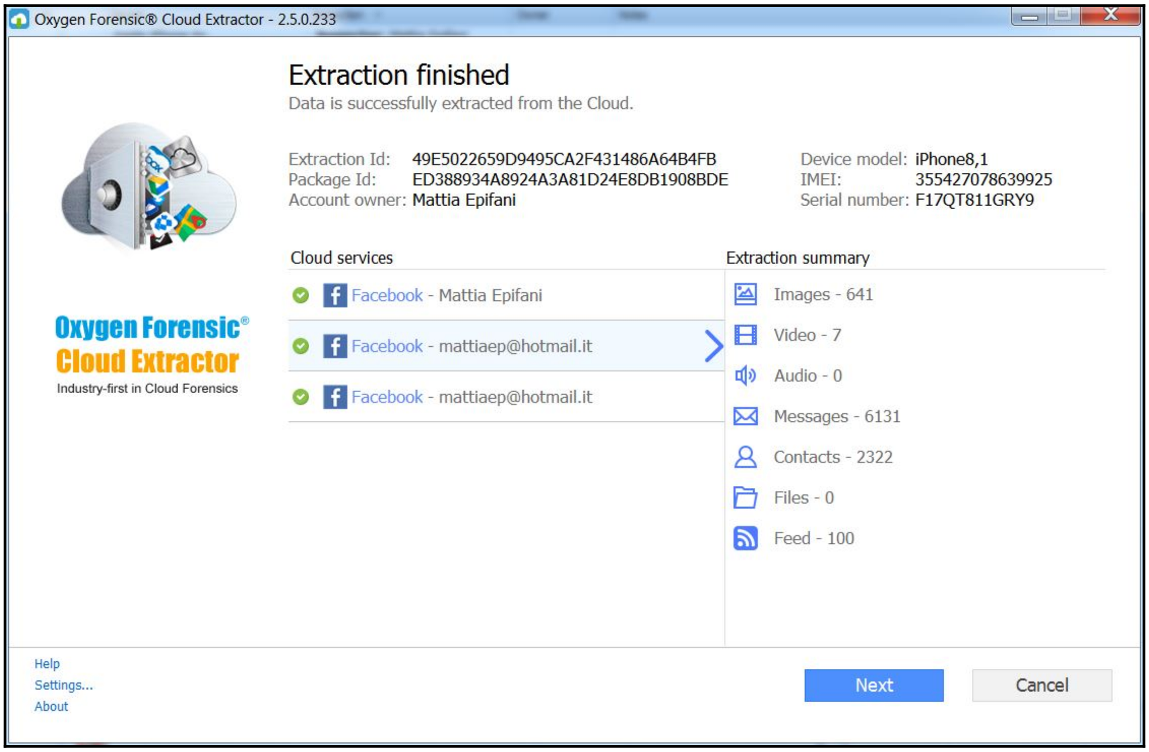
Once the extraction is launched, the software provides the user with a detailed real-time view of extracted data and the task completion percentage:


Oxygen Forensic® Cloud Extractor - 2.5.0.233
Data extraction from cloud services
Extracting data. Please wait...
Facebook - Mattia Epifani
00%
Facebook - mattiaep@hotmail.it
97%
Facebook - mattiaep@hotmail.it
97%
Oxygen Forensic®
Cloud Extractor
Industry-first in Cloud Forensics
Help
Settings...
About
Next
Cancel

```
00:32:26 Writing t_22tFMSHLPGjqMcM4knGekQ thread messages information to the database
00:32:26 Writing m_xzd9eEzOMTKqYgdYtYLeXA message shares information to the database
00:32:26 Writing m_xzd9eEzOMTKqYgdYtYLeXA message attachments information to the database
00:32:26 Writing m_22tFMSHLPGjqMcM4knGekQ message shares information to the database
00:32:26 Writing m_22tFMSHLPGjqMcM4knGekQ message attachments information to the database
```

```
00:32:27 Writing 1667183091 user information to database
00:32:27 Writing 100000180619094 user information to database
00:32:27 Writing t_BWLfnnPRaFKtHLulbCGguw thread messages information to the database
00:32:27 Writing m_BWLfnnPRaFKtHLulbCGguw message shares information to the database
00:32:27 Writing m_BWLfnnPRaFKtHLulbCGguw message attachments information to the database
```

When the extraction is completed, it shows a detailed summary of the extracted data, broken down by account and by type:



Oxygen Forensic® Cloud Extractor - 2.5.0.233

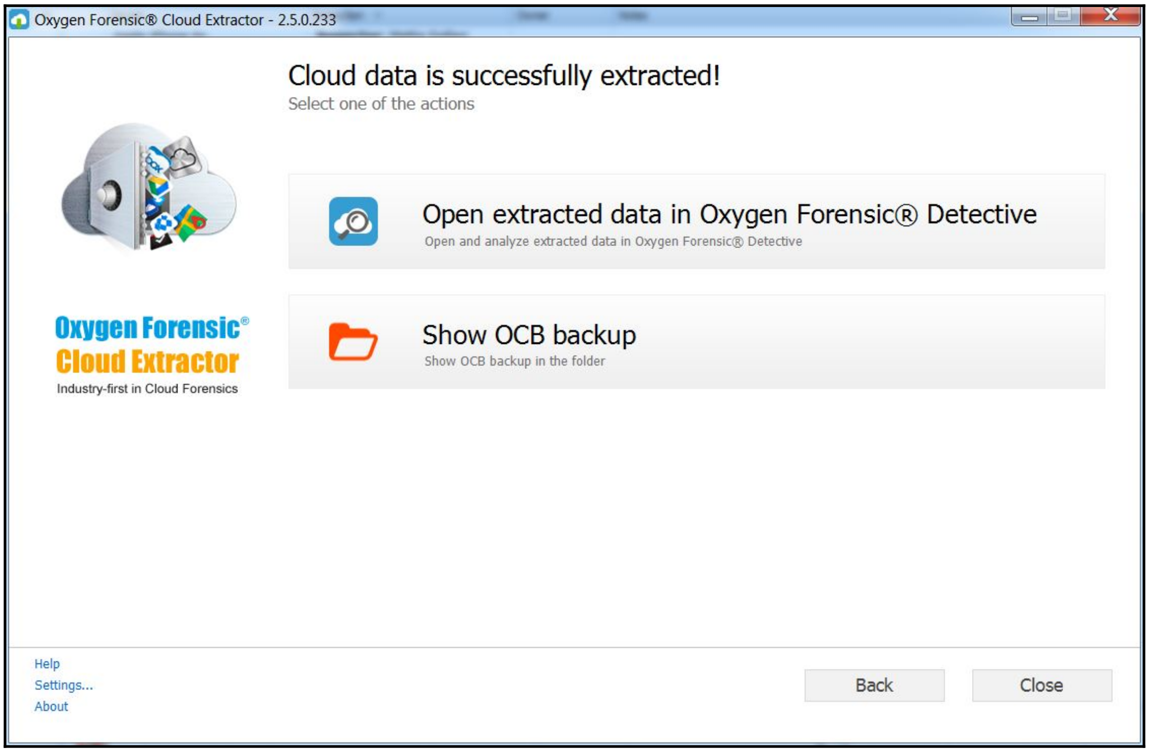
Extraction finished

Data is successfully extracted from the Cloud.

Extraction Id:	49E5022659D9495CA2F431486A64B4FB	Device model:	iPhone8,1
Package Id:	ED388934A8924A3A81D24E8DB1908BDE	IMEI:	355427078639925
Account owner:	Mattia Epifani	Serial number:	F17QT811GRY9

Cloud services	Extraction summary
Facebook - Mattia Epifani	Images - 641
Facebook - mattiaep@hotmail.it	Video - 7
Facebook - mattiaep@hotmail.it	Audio - 0
	Messages - 6131
	Contacts - 2322
	Files - 0
	Feed - 100

The extracted data can then be viewed within Oxygen Forensic Detective or saved in an archive **Oxygen Cloud Backup** (**OCB**) for later analysis:

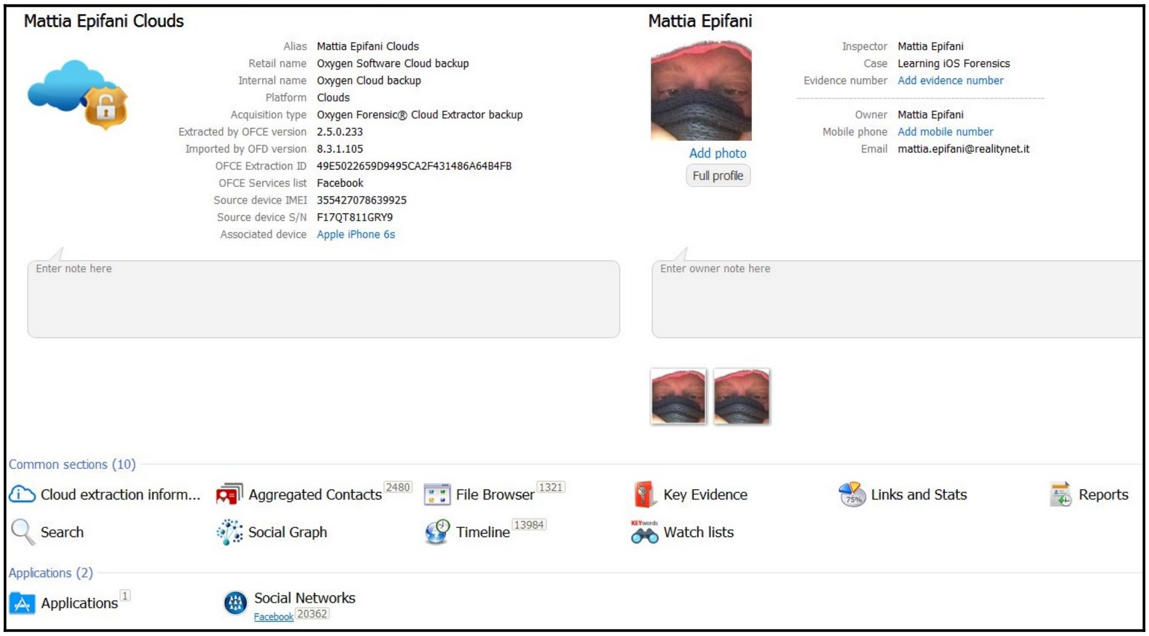

Oxygen Forensic® Cloud Extractor - 2.5.0.233

Cloud data is successfully extracted!
Select one of the actions

Open extracted data in Oxygen Forensic® Detective
Open and analyze extracted data in Oxygen Forensic® Detective

Show OCB backup
Show OCB backup in the folder

Opening of the backup within the software allows you to navigate easily around the extracted data:



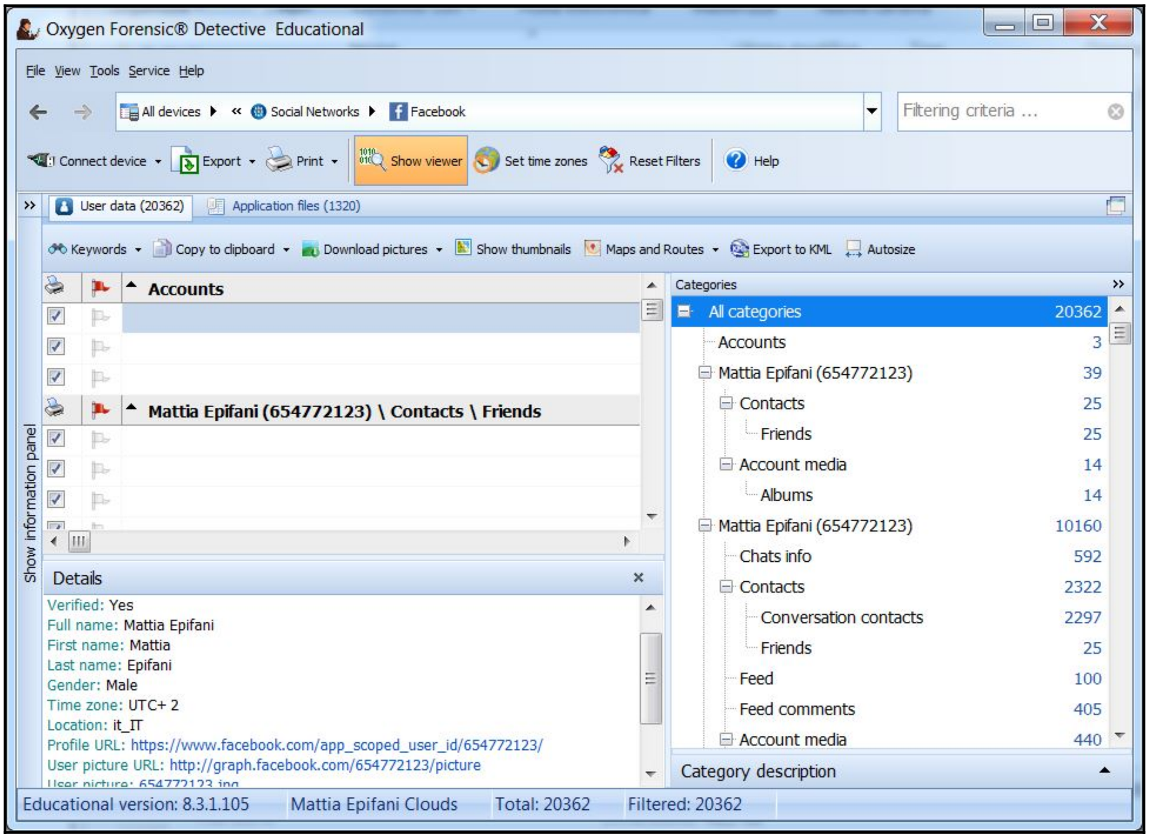
Mattia Epifani Clouds

Alias	Mattia Epifani Clouds
Retail name	Oxygen Software Cloud backup
Internal name	Oxygen Cloud backup
Platform	Clouds
Acquisition type	Oxygen Forensic® Cloud Extractor backup
Extracted by OFCE version	2.5.0.233
Imported by OFD version	8.3.1.105
OFCE Extraction ID	49E5022659D9495CA2F431486A64B4FB
OFCE Services list	Facebook
Source device IMEI	355427078639925
Source device S/N	F17QT811GRY9
Associated device	Apple iPhone 6s

Mattia Epifani

Inspector	Mattia Epifani
Case	Learning iOS Forensics
Evidence number	Add evidence number
Owner	Mattia Epifani
Mobile phone	Add mobile number
Email	mattia.epifani@realitynet.it

In the specific case of the current example, you will have access to the **Social Network** section and view all the extracted data in a structured way:


Oxygen Forensic® Detective Educational
File View Tools Service Help
All devices « Social Networks Facebook
Filtering criteria ...
Connect device Export Print Show viewer Set time zones Reset Filters Help
User data (20362) Application files (1320)
Keywords Copy to clipboard Download pictures Show thumbnails Maps and Routes Export to KML Autosize
Show information panel

Accounts

Mattia Epifani (654772123) \ Contacts \ Friends

Details

Verified: Yes
Full name: Mattia Epifani
First name: Mattia
Last name: Epifani
Gender: Male
Time zone: UTC+ 2
Location: it_IT
Profile URL: https://www.facebook.com/app_scoped_user_id/654772123/
User picture URL: http://graph.facebook.com/654772123/picture
User picture: 654772123.jpg

Categories	
All categories	20362
Accounts	3
Mattia Epifani (654772123)	39
Contacts	25
Friends	25
Account media	14
Albums	14
Mattia Epifani (654772123)	10160
Chats info	592
Contacts	2322
Conversation contacts	2297
Friends	25
Feed	100
Feed comments	405
Account media	440

Category description

Educational version: 8.3.1.105 | Mattia Epifani Clouds | Total: 20362 | Filtered: 20362

Summary

In this chapter, we introduced the iCloud service provided by Apple to store files on remote servers and backup its devices. In particular, we showed the techniques to download the backups stored on iCloud when you know the user credentials (Apple ID and password) and when you have access to a computer (either Windows or Mac OS X, turned on or off) where it is installed and that uses iCloud Control Panel. In the next chapter, we will introduce you to the most interesting and useful artifacts that can be found on iDevices.

Self-test questions

Q1. When is a new backup on iCloud automatically created?

1. Every 5 minutes
2. It depends on the iOS version
3. When the device is connected to the power cable, to a Wi-Fi network and is locked
4. When the device is locked

Q2. Which of these tools can be used to download a backup from iCloud?

1. iPhone Backup Analyzer
2. InflatableDonkey
3. UFED Physical Analyzer
4. iOS Forensic Toolkit

Q3. Which tool can be used to recover the auth token from a PC with iCloud Control Panel?

1. `Oauth.exe`
2. `Iloot.exe`
3. `Token.exe`
4. `Atex.exe`

Q4. Where are the log files related to iCloud Control Panel stored in Windows 7?

1. `C:\Users\[username]\AppData\Local\Apple Computer\Logs`
2. `C:\Users\[username]\AppData\Local\Apple\Logs`
3. `C:\Users\[username]\AppData\Roaming\Apple Computer\Logs`
4. `C:\Users\[username]\AppData\Roaming\Apple\Logs`

6
Analyzing iOS Devices

The goal of this chapter is to guide you through the analysis of important artifacts that are present on an iDevice. In the first part, the focus will be on the artifacts generated by the system's features or by the interaction of the user with it, referring mainly to the iOS configuration files and to the iOS native applications. In the second part, we will go through the manual analysis of some of the most common third-party applications, with the goal of giving you a general approach that you will be able to apply to all the different apps you will encounter in your way. About this topic, there are also several publications available, some of which are referenced also in the `Appendix A`, *References*. We will conclude with a case study to provide you also with a proprietary analysis software example. All of this focuses on the two main formats used to store data, the **SQLite** databases and the property list (`plist`) files.

How data is stored

Before actually starting the analysis of the artifacts we can find inside an iDevice, let's take a look at how data is structured and in which format it is stored. Inside the Apple filesystem, most of the user data are stored under `/private/var/mobile/`, or simply `/User/` that is a symlink pointing to the previous folder. For the tests in this chapter, we used an iPhone 4s running iOS 9.0.2:

```
# tree -d -L 2 /private/var/mobile/

|-- Applications
|-- Containers
|   |-- Bundle
|   |-- Data
|   `-- Shared
|-- Documents
|   `-- com.apple.springboard.settings
```

```
|-- Library
|   |-- Accounts
|   |-- AddressBook
|   |-- AggregateDictionary
...
|   |-- Keyboard
|   |-- Logs
|   |-- Mail
...
|   |-- Preferences
|   |-- SMS
|   |-- Safari
...
|   |-- SoftwareUpdate
|   |-- Spotlight
|   |-- SpringBoard
...
|-- Media
|   |-- AirFair
|   |-- Books
|   |-- DCIM
|   |-- Downloads
...
`-- MobileSoftwareUpdate
```

Prior to iOS 8, the applications and their data were all stored under `/private/var/mobile/Applications/` while now, starting from iOS 8, the filesystem layout has changed and the application data has been separated from its bundles. The current folder structure is as follows:

- `/private/var/mobile/Containers/Bundle/Application/<UUID>/`: This path is the actual path where the application bundle is stored.
- `/private/var/mobile/Containers/Data/Application/<UUID>/`: This path is the actual path where most of the application data is stored.
- `/private/var/mobile/Containers/Shared/AppGroup/<UUID>/`: As the name of the folder suggests, this path is the path where applications can store data with the aim of sharing it with other apps or extensions. This folder is also very important because you will find crucial data of some very popular apps, such as WhatsApp, Chat db, and so on.

While you may easily guess the meaning of most of the folders from the previous `tree` command line output, you may wonder what those names inside the `Application` folder are. These are the names of the apps represented by their **Universally Unique ID** (**UUID**). Inside each application folder, you will see, most of the time, the same structure that appears something like the following:

```
# tree -L 1 FAA3360F-18A5-4EA2-A331-53F2A49C5A8E/
FAA3360F-18A5-4EA2-A331-53F2A49C5A8E/
|-- Documents
|-- Library
|-- StoreKit
`-- tmp
```

The following structure is of particular importance:

- `<Application_Bundle_Home>/AppName.app`: This file is the application bundle, which will not be backed up
- `<Application_Data_Home>/Documents/`: This path contains application-specific data files
- `<Application_Data_Home>/Library/`: This path contains application-specific files
- `<Application_Data_Home>/Library/Preferences/`: This path contains application-preference files
- `<Application_Data_Home>/Library/Caches/`: This path contains application-specific support files, which will not be backed up
- `<Application_Data_Home>/tmp/`: This path contains temporary files not persistent between application launches, which will not be backed up

Within the Bundle application folder, the `iTunesMetadata.plist` file contains, among others, information related to the product, the Apple account name, and the date of purchase, which may be useful in some cases. You will find one of these files in each application bundle folder.

Within each Data application folder, there is a hidden file named `.com.apple.mobile_container_manager.metadata.plist`, This contains the name identifier corresponding to the UUID. It may prove to be quite handy if you are working on your analysis on a shell and need to identify quickly which applications are there:

```
    iLab1:/private/var/mobile/Containers/Data/Application root# find . -
type f -name ".com.apple.mobile_container_manager.metadata.plist" - exec
plutil {} \; | grep "MCMMetadataIdentifier"
        ...
        MCMMetadataIdentifier = "com.facebook.Messenger";
        MCMMetadataIdentifier = "com.apple.Maps";
        MCMMetadataIdentifier = "com.apple.mobilenotes";
        MCMMetadataIdentifier = "org.mozilla.ios.Firefox";
       MCMMetadataIdentifier = "com.facebook.Facebook";
        MCMMetadataIdentifier = "com.apple.mobilemail";
        MCMMetadataIdentifier = "net.whatsapp.WhatsApp";
        MCMMetadataIdentifier = "com.google.chrome.ios";
        MCMMetadataIdentifier = "com.skype.skype";
        MCMMetadataIdentifier = "com.getdropbox.Dropbox";
        MCMMetadataIdentifier = "com.apple.ServerDocuments";
        MCMMetadataIdentifier = "com.apple.SafariViewService";
        MCMMetadataIdentifier = "com.google.Gmail";
        MCMMetadataIdentifier = "com.toyopagroup.picaboo";
        MCMMetadataIdentifier = "com.apple.mobilesafari";
        MCMMetadataIdentifier = "ph.telegra.Telegraph";
        MCMMetadataIdentifier = "com.apple.iCloudDriveApp";
        MCMMetadataIdentifier = "com.google.Drive";
        ...
```

It is also useful especially, to identify which is your folder of interest, by matching the UUID with the application name you are looking for:

```
    iLab1:/private/var/mobile/Containers/Data/Application root# find . -
type f -name ".com.apple.mobile_container_manager.metadata.plist" - exec
grep -iF "Skype" {} \;
    Binary file ./86212B35-B575-4E69-89F1-
81F0CD8886A2/.com.apple.mobile_container_manager.metadata.plist  matches
```

Last but not least, regarding the format that Apple uses to store its files, you will encounter mostly two types, `plist`, which is mainly used for configuration files, and `SQLite databases`. We will look more into details about both formats in the next section.

Timestamps

A very important aspect that an analyst has to pay attention to is the timestamp convention used. This is crucial, especially if you are analyzing the artifacts manually without one of the specialized commercial tools. Instead of the classical **UNIX Epoch Time**, which represents the number of seconds elapsed since January 1, 1970 00:00:00, the iOS devices adopt the **MAC Absolute Time**, which represents the number of seconds elapsed since January 1, 2001 00:00:00. The difference between the two is 978, 307, 200 seconds. There are several resources available online that you could use to calculate it, or else you can do it on your Mac by adding the preceding value to the MAC Absolute Time value, as in the following example:

```
$ date -u -r `echo '314335349 + 978307200' | bc`
Sat Dec 18 03:22:29 UTC 2010
```

Remember to insert the `-u` switch in order to display it in UTC time or else the system will give you an output on your local time (or whatever is set as the local time on your machine).

Databases

The most common type of data storage on the iOS devices (just like on other mobile platforms in general) is the use of the SQLite databases. Both native as well as third-party applications heavily use SQLite database files to store their data, as we will see in more detail later.

There are several tools available, both free/open source and commercial, such as *SQLite Database Browser*, that offer a GUI interface, as well as the SQLite command-line utility, available from the official SQLite website at `http://www.sqlite.org/`. If you are using a Mac OS X machine for the analysis, it will come with the `sqlite3` utility preinstalled.

The property list files

The property list files, or `plist`, are the other most common data formats used in the iOS devices (and in Mac OS X as well). The `plist` files are mainly used to store configuration information, preferences, and settings. Their format can be compared to the XML format and they are usually represented as binary or plain text files.

A common tool used for parsing a `plist` file under Windows is *plist Editor Pro*, whereas if you are using Mac OS X you can either use XCode to view the `plist` files or the command-line utility, `plutil`.

The iOS configuration files

iOS has many preference and configuration files where it stores tons of data that may turn valuable during an investigation. This section provides you with a detailed (although not exhaustive) list of some of those files that are useful to keep in mind:

- **Account and device information**: Check out `/private/var/root/Library/Lockdown/data_ark.plist`. This contains various information about the device and about its account holder.
- **Account information**: Have a look at `/private/var/mobile/Library/Accounts/Accounts3.sqlite`. This file contains account information.
- **Account information**: Go to `/private/var/mobile/Library/DataAccess/AccountInformation.plist`. You'll find account information used to set up apps here.

- **Airplane Mode**: Check `/private/var/root/Library/Preferences/com.apple.preferences.network.plist`. This specifies whether Airplane Mode is presently enabled on the device.
- **Configuration information and settings**: Go to `/private/var/mobile/Library/Preferences/`. This contains the `plist` files with the system configuration and the settings of the Apple apps.
- **Lockdown certificate info**: Navigate to `/private/var/root/Library/Lockdown/pair_records/`. This contains information about the lockdown/pairing certificates and also the computers the iOS device has been paired with.
- **Network information**: Go to `/private/var/preferences/SystemConfiguration/com.apple.network.identification.plist`. This contains a cache of the IP networking information as the previous network addresses, router addresses, and name servers used. A timestamp for each network is also provided. This file is not present anymore on iOS 9, but you may still find it in case the device has been restored from an older iOS version.

- **Passwords**: Go to `/private/var/Keychains/`. This contains the password saved in the iDevice.
- **SIM card info**: Now have a look at `/private/var/wireless/Library/Preferences/com.apple.commcenter.plist`. This contains the ICCID and IMSI of the SIM card last used in the device.
- **Springboard**: Go to `/private/var/mobile/Library/Preferences/com.apple.springboard.plist`. This contains the order of applications in each screen.
- **Wi-Fi networks**: Now see `/private/var/preferences/SystemConfiguration/com.apple.wifi.plist`. This contains the list of the known Wi-Fi networks, the timestamp of last joined, and other useful information. For more information on this and a deeper analysis, you can have a look at the article available at `http://articles.forensicfocus.com/213/9/3/from-iphone-to-access-point/`.

Native iOS apps

iDevices come with some native applications already installed by Apple, such as Safari browser, e-mail client, calendar, and utilities linked to some basic phone functionalities, such as the Camera, Call History, or the SMS/iMessage. Most of the evidence produced by these native applications and functionalities are located, other than in the application folders themselves, in the `Library` folder:

- `/private/var/mobile/Library/`: In case of physical acquisition or inside the device
- `Backup Service/mobile/Library/`: In case of filesystem acquisition
- `Library`: In case of logical acquisition

Here, we can find data related to communication, preferences, Internet history and cache, keyboard keystrokes, and much more. Other than the `Library` folder, the other very important location is the `Media` folder, `/private/var/mobile/Media/`, where user-created pictures and audio files are usually stored among other things.

Address book

As one could imagine, the `AddressBook` folder under `Library` refers to the information present in the Contact application related to the personal contacts and is stored in SQLite database format. There are two databases of interest, `AddressBook.sqlitedb` and `AddressBookImages.sqlitedb`.

`AddressBook.sqlitedb` contains the actual information saved for each contact, such as name, surname, phone number, e-mail address, and so on. In this database, the tables of interest containing the information mentioned are mainly `ABPerson` and `ABMultiValue`.

`AddressBookImages.sqlitedb` is the database containing the images that the user may have associated to a contact, which is basically the image appearing every time a call to that contact is made or received. The main table of interest in this database is `ABFullSizeImage`.

Audio recordings

The Voice Memos app, preinstalled on the iDevices, lets the user record voice memos. These memos are stored in `/private/var/mobile/Media/Recordings/`. In this folder, there is the `Recordings.db` database that contains information about each voice memo stored, such as the date, duration, memo name, and filename of the actual audio file, which is stored in the same folder.

Calendar

The Calendar application allows the user to manually create events, as well as sync them with other applications, such as the related Mac OS X version of the app or other third-party applications and services. Such information is stored in the following two databases:

- `/private/var/mobile/Library/Calendar/Calendar.sqlitedb`
- `/private/var/mobile/Library/Calendar/Extras.db`

The `Calendar.sqlitedb` database is the main one and contains basically all the information related to the events present in the Calendar, while `Extras.db` contains other information such as the Calendar settings.

Call history

When we press the phone application icon, we see a lot of information, almost all coming from one database, `/private/var/wireless/Library/CallHistoryDB/CallHistory.storedata`. Here we can find tracks about incoming, outgoing, and missed calls along with the time and date they occurred and their duration. This database refers to both standard calls and FaceTime calls. As we can see in the following example, the table of interest is `call`:

```
# ls -l
-rw-r--r-- 1 mobile mobile  36864 Oct 22  2015 CallHistory.storedata
# sqlite3 CallHistory.storedata
SQLite version 3.8.10.2
Enter ".help" for instructions
sqlite> .mode line
sqlite> .tables
ZCALLDBPROPERTIES  Z_METADATA         Z_PRIMARYKEY
ZCALLRECORD        Z_MODELCACHE
sqlite> select * from ZCALLRECORD;
...
           ZANSWERED = 0
           ZCALLTYPE = 1
 ZDISCONNECTED_CAUSE =
     ZFACE_TIME_DATA =
ZNUMBER_AVAILABILITY = 0
         ZORIGINATED = 1
               ZREAD = 1
               ZDATE = 490357911.066614
           ZDURATION = 0.0
          ZDEVICE_ID =
   ZISO_COUNTRY_CODE = ch
           ZLOCATION = Switzerland
               ZNAME =
          ZUNIQUE_ID = E49FF4D0-8AF7-4A74-9D8D-43E2B5D7A574
            ZADDRESS = +4179******4
...
```

The most interesting fields are as follows:

- `ZANSWERED`: This indicates if the call has been accepted (`1`) or rejected (0).
- `ZCALLTYPE`: This indicates if it was a standard phone call (`1`), a FaceTime audio only call (`16`), or a full audio/video FaceTime call (`8`).
- `ZORIGINATED`: This indicates whether the call was outgoing (`1`) or incoming (0).
- `ZDATE`: This indicates the date and time of when the call happened. Note that this value is expressed in Mac Absolute time, which represents the number of seconds elapsed since 1/1/2001 00:00:00 GMT. Therefore, if you don't have an application that converts the time for you, simply add 978307200 and use the `date` command as follows (example from a Mac):

```
$ echo '490357911.066614+978307200' | bc  1468665111.066614
$ date -ur 1468665111
Sat Jul 16 10:31:51 UTC 2016
```

- `ZISO_COUNTRY_CODE`: This indicates, as the name suggests, the two letter ISO country code according to the prefix of the phone number indicated in the `ZADDRESS` field.
- `ZLOCATION`: This indicates the full name of the country according to the preceding field.
- `ZADDRESS`: Last but not least, this field indicates the phone number where the call came from or went to.

Starting from iOS 8, the path has slightly changed from `/private/var/wireless/Library/CallHistory/call_history.db` to `/private/var/wireless/Library/CallHistoryDB/CallHistory.storedata`. If a device has been upgraded from iOS 7 to iOS 8, you will find both the old and the new databases, with the call history data prior the upgrade.

There are yet two other important files related to the phone application to be analyzed. The plist file, `/private/var/mobile/Library/Preferences/com.apple.mobilephone.plist`, has the `DialerSavedNumber` value, which is the last phone number manually entered into the dialer and actually dialed. The important thing to note here is that this value will remain even if the user deletes the last call placed from the call history list, which will of course be also deleted from the `CallHistory.storedata` database we have just analyzed. The second file that may also be of interest during an investigation is `/private/var/mobile/Library/Preferences/com.apple.mobilephone.speeddial.plist`, which contains the phone numbers added to the phone favorites list.

E-mail

Apple Mail client-related data is stored at `/private/var/mobile/Library/Mail/`, which contains databases storing the e-mail messages sent, received, and drafted that are stored on the device, as well as a folder for each separate account (POP/IMAP) that has been configured within the Mail application. So, you may want to take a look at all the content you find in there. To give you an example, the folder content may look like the following command:

```
    # ls -l

    drwxr-xr-x 2 mobile mobile     68 Oct 26  2015  AttachmentPlaceholders/
    -rw-r--r-- 1 mobile mobile     42 Jul 17 19:24 AutoFetchEnabled
    -rw-r--r-- 1 mobile mobile 196608 Jul 17 19:24 Envelope\ Index
    -rw-r--r-- 1 mobile mobile  32768 Jul 17 19:24 Envelope\ Index-shm
    -rw-r--r-- 1 mobile mobile      0 Jul 17 19:24 Envelope\ Index-wal
    drwx------ 3 mobile mobile    136 Jul 17 19:24 IMAP-
demo.room.004\@gmail.com\@imap.gmail.com/
    -rw-r--r-- 1 mobile mobile    468 Jul 17 19:24 MailboxCollections.plist
    drwx------ 2 mobile mobile    102 Jul 17 19:24 Mailboxes/
    -rw-r--r-- 1 mobile mobile  86016 Jul 17 19:24 Protected\ Index
    -rw-r--r-- 1 mobile mobile  32768 Jul 17 19:24 Protected\ Index-shm
    -rw-r--r-- 1 mobile mobile      0 Jul 17 19:24 Protected\ Index-wal
    -rw-r--r-- 1 mobile mobile   4096 Jul 13 21:18 Recents
    -rw-r--r-- 1 mobile mobile  32768 Jul 17 19:11 Recents-shm
    -rw-r--r-- 1 mobile mobile 366712 Jul 17 19:24 Recents-wal
    drwxr-xr-x 2 mobile mobile     68 Jul 17 19:24 SceneThumbnailCache/
    drwx------ 2 mobile mobile     68 Oct 20  2015 Vault/
    -rw-r--r-- 1 mobile mobile    462 Jul 17 19:24 metadata.plist
```

Although without any extension, most of these files are SQLite databases (as you may guess from the presence of the `-shm` and `-wal` files). For example, the Envelope Index database contains the list of mailboxes and metadata, while the Protected Index database contains the list of the e-mails present in the Inbox, where the last one is the most recent:

```
# sqlite3 Protected\ Index
SQLite version 3.8.10.2
Enter ".help" for instructions
sqlite> .mode line
sqlite> .tables
message_data  messages
sqlite> SELECT * FROM messages;
message_id = 9
    sender = "Facebook" <update@facebookmail.com>
   subject = You have more friends on Facebook than you think
       _to = Demo < <account_username>@gmail.com>
...
message_id = 130
    sender = "PayPal" <paypal@e.paypal.it>
   subject = Accordi legali PayPal
       _to = <account_username>@gmail.com
...
```

Images and photos

User photos inside an iDevice are stored at `/private/var/mobile/Media/`, where the two main folders are as follows:

- `DCIM`: This folder contains the user-created photos via the built-in camera (usually in the `.jpg` format) and screenshots taken by the user by pressing the Power and Home buttons together (usually in the `.png` format)
- `PhotoData`: This folder contains, among other data, the photo albums synced with a computer or the cloud

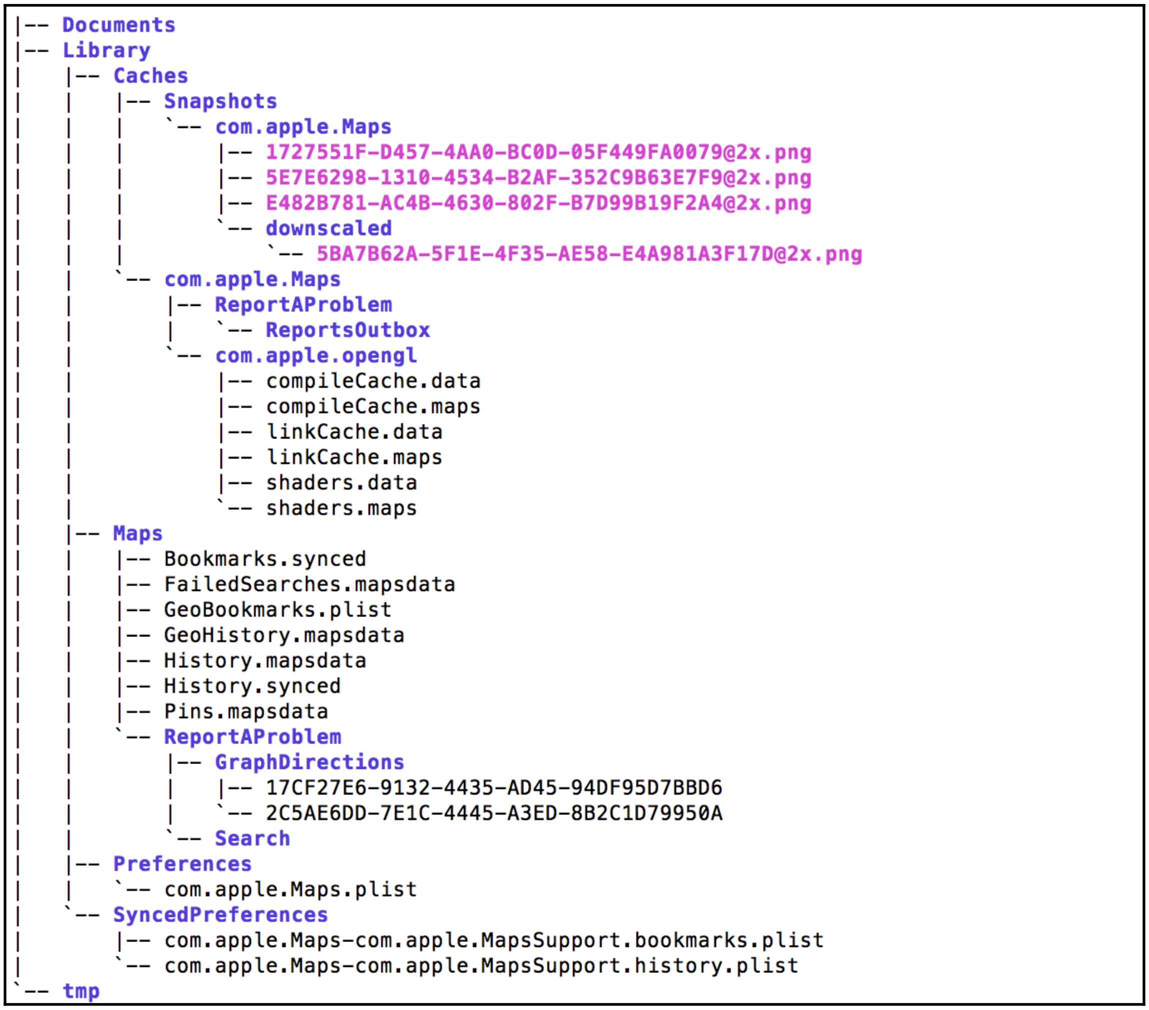
Moreover, it is very important not to forget the *thumbnails*. In fact, for each photo, iOS will generate a thumbnail, store it in `/private/var/mobile/Media/PhotoData/Thumbnails/`, and save any information about the original image in the `Photos.sqlite` database within the `PhotoData` folder. This is important because thumbnails and information related to the original picture may still be available or recoverable from the SQLite deleted entries (refer to the related section later on in this chapter) even if the original picture is not available anymore.

When analyzing photos, it is important to remember to check for the Exif metadata, which may contain other precious information such as geographical coordinates of where the photo was taken.

For an in-depth analysis of this topic, we advise the reader to have a look at the article available at `http://linuxsleuthing.blogspot.it/213/5/ios6-photo-streams-recover-deleted.html`

Maps

Since the release of iOS 6 in 2012, Apple includes its own Maps application. Files and locations of interest are located within the main folder `/private/var/mobile/Containers/Data/Application/2EA1D4AC-1C04-4CA5-8A77-349D47468457/`, which contains the history of the searches made by the users as well as the list of locations bookmarked, and also the `Library/Preferences/com.apple.Maps.plist` file, which contains information related to the last search that has been made by the user, such as longitude and latitude coordinates as well as the search query made.



```
|-- Documents
|-- Library
|   |-- Caches
|   |   |-- Snapshots
|   |   |   `-- com.apple.Maps
|   |   |       |-- 1727551F-D457-4AA0-BC0D-05F449FA0079@2x.png
|   |   |       |-- 5E7E6298-1310-4534-B2AF-352C9B63E7F9@2x.png
|   |   |       |-- E482B781-AC4B-4630-802F-B7D99B19F2A4@2x.png
|   |   |       `-- downscaled
|   |   |           `-- 5BA7B62A-5F1E-4F35-AE58-E4A981A3F17D@2x.png
|   |   `-- com.apple.Maps
|   |       |-- ReportAProblem
|   |       |   `-- ReportsOutbox
|   |       `-- com.apple.opengl
|   |           |-- compileCache.data
|   |           |-- compileCache.maps
|   |           |-- linkCache.data
|   |           |-- linkCache.maps
|   |           |-- shaders.data
|   |           `-- shaders.maps
|   |-- Maps
|   |   |-- Bookmarks.synced
|   |   |-- FailedSearches.mapsdata
|   |   |-- GeoBookmarks.plist
|   |   |-- GeoHistory.mapsdata
|   |   |-- History.mapsdata
|   |   |-- History.synced
|   |   |-- Pins.mapsdata
|   |   `-- ReportAProblem
|   |       |-- GraphDirections
|   |       |   |-- 17CF27E6-9132-4435-AD45-94DF95D7BBD6
|   |       |   `-- 2C5AE6DD-7E1C-4445-A3ED-8B2C1D79950A
|   |       `-- Search
|   |-- Preferences
|   |   `-- com.apple.Maps.plist
|   `-- SyncedPreferences
|       |-- com.apple.Maps-com.apple.MapsSupport.bookmarks.plist
|       `-- com.apple.Maps-com.apple.MapsSupport.history.plist
`-- tmp
```

Notes

The Notes application stores information about the user-created notes in `/private/var/mobile/Library/Notes/notes.sqlite`. The main tables of interest are `ZNOTE` and `ZNOTEBODY`; they contain note title, content, creation and modification date, and so on.

```
    # sqlite3 notes.sqlite

    SQLite version 3.8.10.2
```

```
Enter ".help" for instructions
sqlite> .mode line
sqlite> .tables
ZACCOUNT          ZNOTEATTACHMENT   ZPROPERTY         Z_MODELCACHE
ZNEXTID           ZNOTEBODY         ZSTORE            Z_PRIMARYKEY
ZNOTE             ZNOTECHANGE       Z_METADATA
```

Starting from iOS 9.3, Apple introduced the possibility of protecting/locking notes via a password. Since the user has to choose each note that has to be password protected (it is not an *all or nothing* model), you may not be able to access those notes directly.

Safari

Safari is the Apple browser that comes preinstalled with every iDevice. It allows the user to browse websites, save bookmarks, and so on. All these activities are stored in the two locations, `/private/var/mobile/Library/Safari/` and the Safari main application folder, `/private/var/mobile/Containers/Data/Application/C6254D15-92CE-4989-9427-26A589DFA8B7/`. In particular, the folder detail is given as follows:

- **Safari Bookmarks**: This information is stored in `/private/var/mobile/Library/Safari/Bookmarks.db`. It contains the database with the saved bookmarks.
- **Safari Cookies**: This information is stored at `/private/var/mobile/Library/Cookies/Cookies.binarycookies`. Website cookies are stored here. To parse this binary file you can use the `BinaryCookieReader.py` Python script available on GitHub (`https://gist.github.com/sh1nb1/4bb8b73737bfe5f5ab8`).
- **Safari Temporary Files**: This information is stored at `<app_folder>/Library/Caches/WebKit/NetworkCache/Version 4/`. This folder contains another two subfolders, `Blob` and `Record`, where temporary files and code of the visited pages are stored. This is important because it will be possible to see/rebuild the same page as it was seen/visited by the user. Another important thing to remember is that when the user deletes their browser cache, these folders will be cleared.
- **Safari search history**: This information is stored in `<app_folder>/Library/Preferences/com.apple.mobilesafari.plist`. It contains a list of recent searches made through Safari. An important thing to remember is that when the user deletes their browser cache or history, this file will not be erased, but in iOS 9 its content will be cleared.

- **Safari Suspended State**: This information is stored in `<app_folder>/Library/Safari/SuspendState.plist`. It contains the last state of Safari at the time when the user pressed the Home button, the iPhone was powered off, or the browser crashed. In order to be able to restore such a state when the browser resumes, this file contains the list of windows and websites that were open when one of the previously-mentioned events occurred and the browser closed.
- **Safari Thumbnails**: This information is stored at `<app_folder>/Library/Caches/Safari/Thumbnails/`. This folder contains screenshots of the last active browser pages viewed via WebKit, for example, by the third-party apps.

- **Safari Web Cache**: This information is stored in `Library/Caches/com.apple.mobilesafari/Cache.db`. It contains objects that are recently downloaded and cached in the Safari browser.
- **Safari History**: This information is stored in `<app_folder>/Library/Safari/History.db`. It contains the Safari web browser history. Of course, if it has been cleared by the user, it will not contain the history prior to that.

SMS/iMessage

Similar to the Call History, there is one database storing SMSes, MMSes, and iMessages sent or received by the user. The database is at `/private/var/mobile/Library/SMS/sms.db`, and it also contains the information related to attachments eventually present in MMSes or iMessages. In such cases, the files part of MMSes or iMessages are stored in the subfolder, `Library/SMS/Attachments/`. Finally, the last folder of interest regarding SMS is `Library/SMS/Drafts`, where each draft contains its own folder as the `plist` file, which is timestamped, identifying when the message was typed and then abandoned.

Voicemail

The `Voicemail` folder at `/private/var/mobile/Library/` contains both the audio file of each voicemail recorded message stored as AMR codec audio files and the `voicemail.db` database, where information related to each voicemail audio message file is saved, such as the sender, the date, the duration, and so on.

Other iOS forensic traces

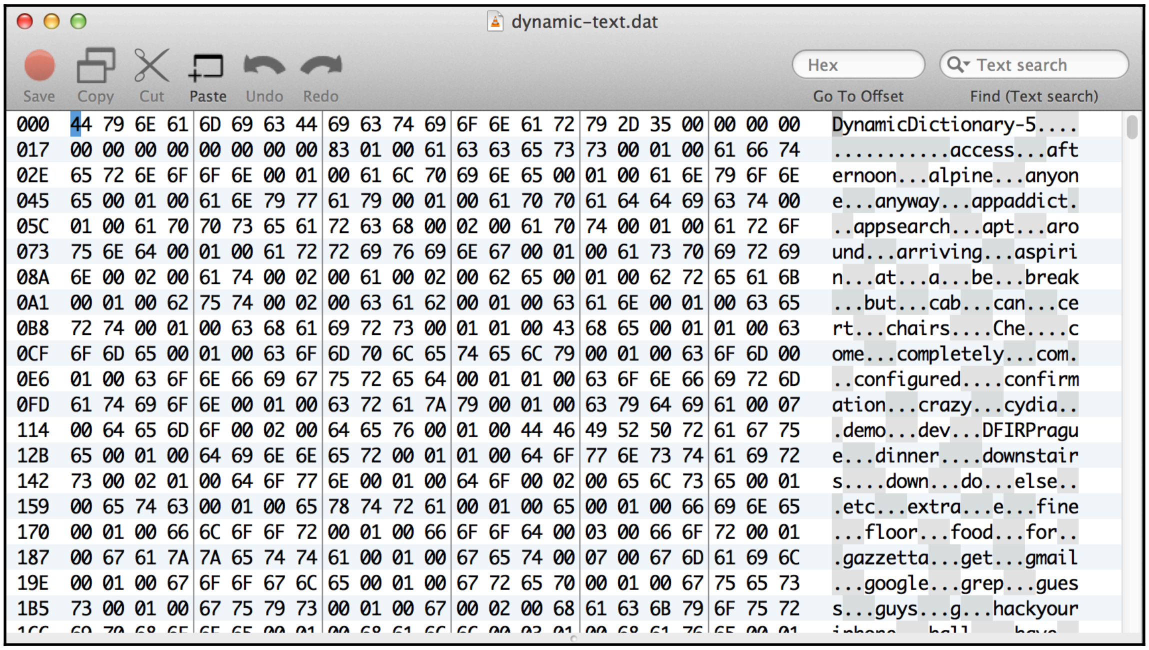
In this section, we will list some other locations of interesting artifacts. Those listed here are not strictly related to a particular application, but are rather generated from the usage of the device by the interaction of the user with the system.

Clipboard

The `pasteboardDB` file under `/private/var/mobile/Library/Caches/com.apple.UIKit.pboard` is a binary file that contains a cached copy of the data stored on the device's clipboard, which means that the data that have been cut/copied and pasted by the user (that is, passwords or other portions of text that may become relevant) will also be present there.

Keyboard

Two of the iOS features are the auto correction and auto completion of the text while the user is typing. To do this, every time the user types, iOS caches their text in the `dynamic-text.dat` file:



```
000  44 79 6E 61 6D 69 63 44 69 63 74 69 6F 6E 61 72 79 2D 35 00 00 00 00   DynamicDictionary-5....
017  00 00 00 00 00 00 00 00 83 01 00 61 63 63 65 73 73 00 01 00 61 66 74   ...........access...aft
02E  65 72 6E 6F 6F 6E 00 01 00 61 6C 70 69 6E 65 00 01 00 61 6E 79 6F 6E   ernoon...alpine...anyon
045  65 00 01 00 61 6E 79 77 61 79 00 01 00 61 70 70 61 64 64 69 63 74 00   e...anyway...appaddict.
05C  01 00 61 70 70 73 65 61 72 63 68 00 02 00 61 70 74 00 01 00 61 72 6F   ..appsearch...apt...aro
073  75 6E 64 00 01 00 61 72 72 69 76 69 6E 67 00 01 00 61 73 70 69 72 69   und...arriving...aspiri
08A  6E 00 02 00 61 74 00 02 00 61 00 02 00 62 65 00 01 00 62 72 65 61 6B   n...at...a...be...break
0A1  00 01 00 62 75 74 00 02 00 63 61 62 00 01 00 63 61 6E 00 01 00 63 65   ...but...cab...can...ce
0B8  72 74 00 01 00 63 68 61 69 72 73 00 01 01 00 43 68 65 00 01 01 00 63   rt...chairs....Che....c
0CF  6F 6D 65 00 01 00 63 6F 6D 70 6C 65 74 65 6C 79 00 01 00 63 6F 6D 00   ome...completely...com.
0E6  01 00 63 6F 6E 66 69 67 75 72 65 64 00 01 01 00 63 6F 6E 66 69 72 6D   ..configured....confirm
0FD  61 74 69 6F 6E 00 01 00 63 72 61 7A 79 00 01 00 63 79 64 69 61 00 07   ation...crazy...cydia..
114  00 64 65 6D 6F 00 02 00 64 65 76 00 01 00 44 46 49 52 50 72 61 67 75   .demo...dev...DFIRPragu
12B  65 00 01 00 64 69 6E 6E 65 72 00 01 01 00 64 6F 77 6E 73 74 61 69 72   e...dinner....downstair
142  73 00 02 01 00 64 6F 77 6E 00 01 00 64 6F 00 02 00 65 6C 73 65 00 01   s....down...do...else..
159  00 65 74 63 00 01 00 65 78 74 72 61 00 01 00 65 00 01 00 66 69 6E 65   .etc...extra...e...fine
170  00 01 00 66 6C 6F 6F 72 00 01 00 66 6F 6F 64 00 03 00 66 6F 72 00 01   ...floor...food...for..
187  00 67 61 7A 7A 65 74 74 61 00 01 00 67 65 74 00 07 00 67 6D 61 69 6C   .gazzetta...get...gmail
19E  00 01 00 67 6F 6F 67 6C 65 00 01 00 67 72 65 70 00 01 00 67 75 65 73   ...google...grep...gues
1B5  73 00 01 00 67 75 79 73 00 01 00 67 00 02 00 68 61 63 6B 79 6F 75 72   s...guys...g...hackyour
```

This file is located at `/private/var/mobile/Library/Keyboard`. This is the default file, but of course, iOS creates one for each language used and configured in the keyboard and stores it in the same folder. In the following example, the second file is related to the Italian keyboard configuration:

```
# ls -l
drwxr-xr-x 4 mobile mobile 136 Oct 26  2015 CoreDataUbiquitySupport/
drwxr-xr-x 2 mobile mobile 102 Oct 22  2015 de-dynamic.lm/
-rw------- 1 mobile wheel  307 Jul 17 18:35 dynamic-text.dat
drwxr-xr-x 2 mobile mobile 272 Jul 17 20:00 en-dynamic.lm/
drwxr-xr-x 2 mobile mobile 272 Jul 17 20:00 it-dynamic.lm/
-rw------- 1 mobile wheel   65 Jul 17 19:57 it_IT-dynamic-text.dat
```

Location

With iOS 4, there was the **Consolidated GPS cache**, a database containing location information associated with every Wi-Fi hotspot and cell tower that the device had been in range with. In such a database located at `/private/var/root/Library/Caches/locationd/consolidated.db`, the `WifiLocation` and `CellLocation` tables contained information cached locally by the device and included the Wi-Fi access points and cellular towers that came within range of the device at a given time and included a horizontal accuracy (in meters), believed to be a guesstimate at the distance from the device. Such data, other than remaining forever in that database, was allegedly sent periodically to Apple. After the so-called **location gate** scandal that arose after the discovery of such a database, Apple kind of dismissed it.

However, the new databases that took the place of `consolidated.db` are `cache_encryptedA.db`, `lockCache_encryptedA.db`, and `cache_encryptedB.db` stored under the `/private/var/root/Library/Caches/locationd/` folder. As for its predecessor, these databases contain geographical coordinates of frequent and recent locations, Wi-Fi access points, and apparently, cell towers that have been in the range of the device. The only differences in this case are that this data lasts only for 8 days before being cleared out. To dump these databases you may also use a Python script from Sarah Edwards (her Twitter handle is `@iamevltwin`), **iOS Location Scraper**, which can be downloaded from the GitHub repository at `https://github.com/mac4n6/iOS-Locations-Scraper`.

The other very important point to keep in mind regarding the geolocation artifacts is that many other applications, especially third-party ones such as those about fitness that people may use to keep track of their path when running, may store geographical coordinates and related timestamps as well and in clear text.

Snapshots

Every time a user pushes the Home button to move from an application screen back to the desktop, iOS uses a **fade-out** effect for the transition between the two screens. To do so, iOS creates screenshots of the current screen and then applies the fade-out effect to that picture. These screenshots are stored in the following locations:

- `/private/var/mobile/Library/Caches/Snapshots/`
- `/private/var/mobile/Containers/Data/Applications/<UUID>/Library/Caches/Snapshots/`

The first path refers to the preinstalled Apple applications, while the second is the path where to find the snapshots for each application. It is clear that this feature could be a goldmine of information. For example, there could be screenshots containing SMSes or e-mail messages that are no longer available because they have been deleted.

It is important to remember that only the last snapshot is taken for each application. Therefore, the analyst should interact and browse inside the device as little as possible in order not to overwrite and lose possible crucial evidence.

Wallpaper

Current images used as wallpapers are stored in `/private/var/mobile/Library/SpringBoard/`. There are two different images:`HomeBackgroundThumbnail.jpg`, which refers to the wallpaper when the device is unlocked, and `LockBackgroundThumbnail.jpg`, which refers to the wallpaper of the device when it is locked.

iOS crash reports

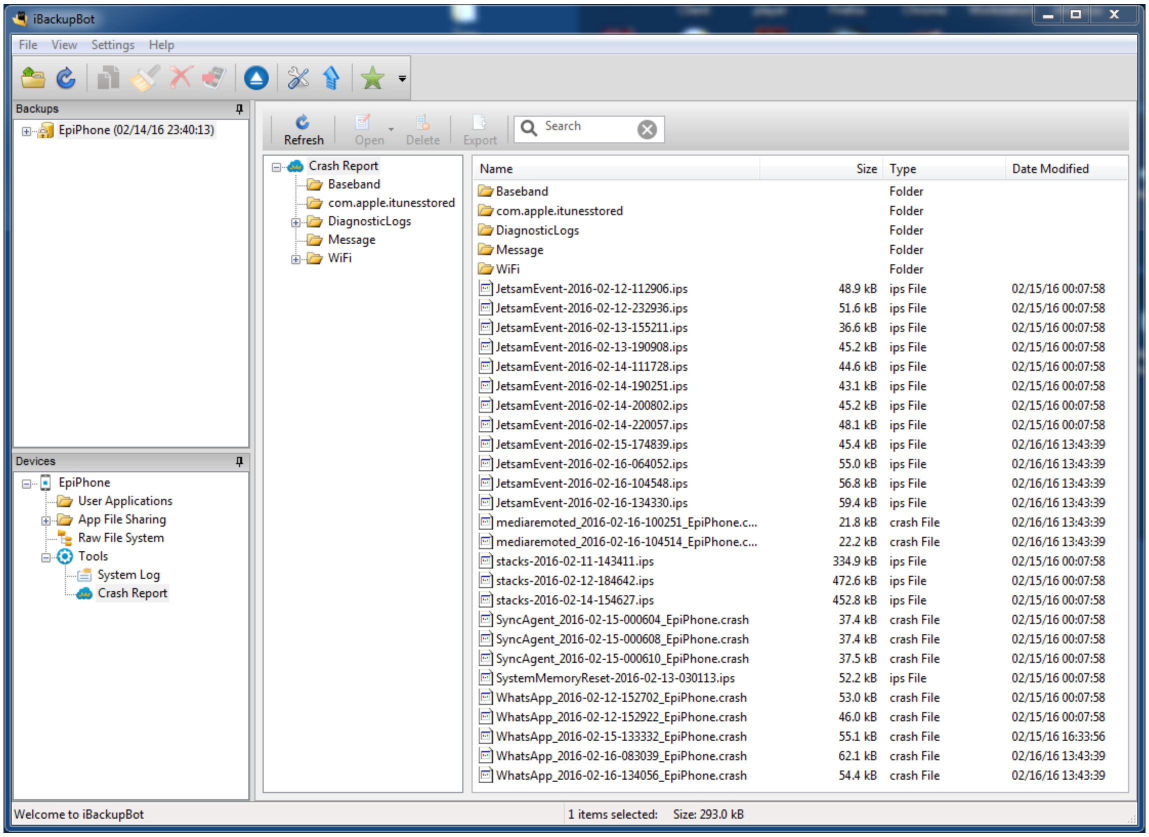
The iOS operating system uses crash reports to keep track of errors and crashes of native as well as third-party applications. A crash log typically contains a description of the system state when the application generated the error or terminated it. Crash reports are managed by a service that is responsible for collecting and aggregating the logs. For an analysis of the operation of the crash report service and the crash log structure refer to the following web pages:

- Understanding and Analyzing iOS Application Crash Reports: `https://developer.apple.com/library/ios/technotes/tn2151/_index.html`
- Understanding Crash Reports on iPhone OS: `https://developer.apple.com/videos/play/wwdc21/317/`
- Analyzing Crash Reports: `https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/AnalyzingCrashReports/AnalyzingCrashReports.html`
- Demystifying iOS Application Crash Logs: `https://www.raywenderlich.com/2374/demystifying-ios-application-crash-logs`

Typically, the devices are configured to automatically synchronize crash logs with a PC/Mac, every time the device is connected and runs iTunes. For this reason, it is advisable to check, within a computer, if you have all the crash logs of devices that have been linked together in time. Depending on the operating system, crash logs are stored in different locations:

- **Windows XP**: `C:\Documents and Settings\<USERNAME>\Application Data\Apple Computer\Logs\CrashReporter\MobileDevice\<DEVICE_NAME>`
- **Windows Vista/7/8/10**: `C:\Users\<USERNAME>\AppData\Roaming\Apple Computer\Logs\CrashReporter\MobileDevice\<DEVICE_NAME>`
- **Mac OS X**: `~/Library/Logs/CrashReporter/MobileDevice/<DEVICE_NAME>`

Crash reports can also be extracted from an iOS device directly dialoguing with the crash report service (yes, it has the same name) through iTunes on PC or Mac, or XCode on Mac. As explained in `Chapter 3`, *Evidence Acquisition from iDevices*, several third-party software offer the ability to remove the logs and save them for later analysis; among these we remember here are iBackupBot and iTools. They can be extracted from an unlocked device or from a turned on and locked device with a valid pairing certificate, and also if a backup password was set by the user. The following screenshot shows an example of a Crash Report extraction from an iPhone 6s using iBackupBot:

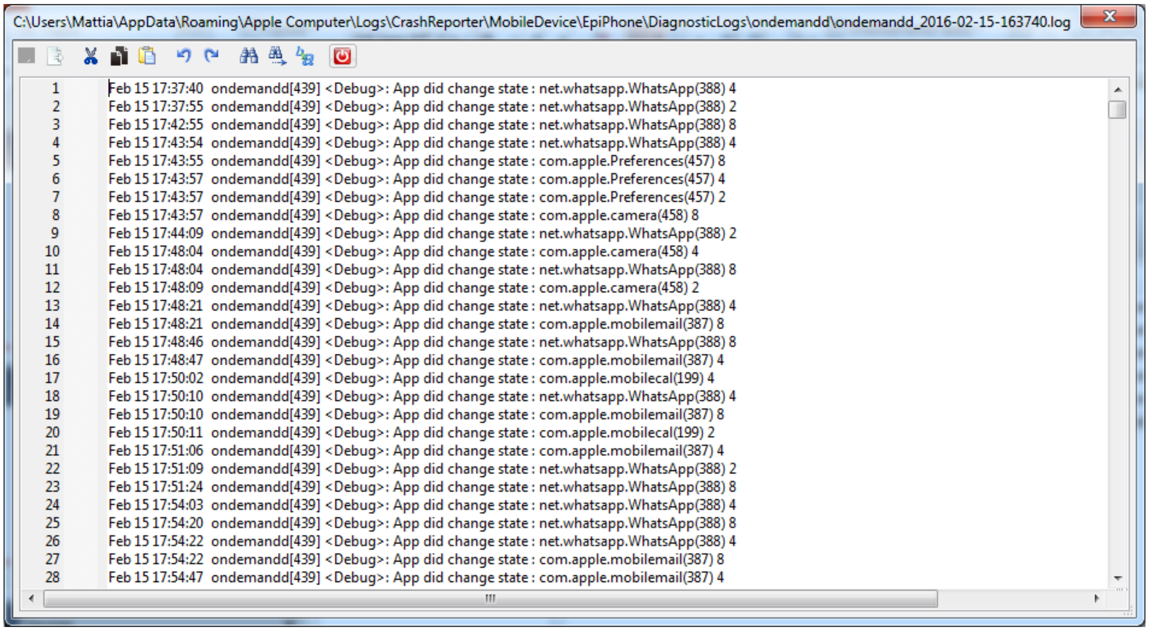


Name	Size	Type	Date Modified
Baseband		Folder	
com.apple.itunesstored		Folder	
DiagnosticLogs		Folder	
Message		Folder	
WiFi		Folder	
JetsamEvent-2016-02-12-112906.ips	48.9 kB	ips File	02/15/16 00:07:58
JetsamEvent-2016-02-12-232936.ips	51.6 kB	ips File	02/15/16 00:07:58
JetsamEvent-2016-02-13-155211.ips	36.6 kB	ips File	02/15/16 00:07:58
JetsamEvent-2016-02-13-190908.ips	45.2 kB	ips File	02/15/16 00:07:58
JetsamEvent-2016-02-14-111728.ips	44.6 kB	ips File	02/15/16 00:07:58
JetsamEvent-2016-02-14-190251.ips	43.1 kB	ips File	02/15/16 00:07:58
JetsamEvent-2016-02-14-200802.ips	45.2 kB	ips File	02/15/16 00:07:58
JetsamEvent-2016-02-14-220057.ips	48.1 kB	ips File	02/15/16 00:07:58
JetsamEvent-2016-02-15-174839.ips	45.4 kB	ips File	02/16/16 13:43:39
JetsamEvent-2016-02-16-064052.ips	55.0 kB	ips File	02/16/16 13:43:39
JetsamEvent-2016-02-16-104548.ips	56.8 kB	ips File	02/16/16 13:43:39
JetsamEvent-2016-02-16-134330.ips	59.4 kB	ips File	02/16/16 13:43:39
mediaremoted_2016-02-16-100251_EpiPhone.c...	21.8 kB	crash File	02/16/16 13:43:39
mediaremoted_2016-02-16-104514_EpiPhone.c...	22.2 kB	crash File	02/16/16 13:43:39
stacks-2016-02-11-143411.ips	334.9 kB	ips File	02/15/16 00:07:58
stacks-2016-02-12-184642.ips	472.6 kB	ips File	02/15/16 00:07:58
stacks-2016-02-14-154627.ips	452.8 kB	ips File	02/15/16 00:07:58
SyncAgent_2016-02-15-000604_EpiPhone.crash	37.4 kB	crash File	02/15/16 00:07:58
SyncAgent_2016-02-15-000608_EpiPhone.crash	37.4 kB	crash File	02/15/16 00:07:58
SyncAgent_2016-02-15-000610_EpiPhone.crash	37.5 kB	crash File	02/15/16 00:07:58
SystemMemoryReset-2016-02-13-030113.ips	52.2 kB	ips File	02/15/16 00:07:58
WhatsApp_2016-02-12-152702_EpiPhone.crash	53.0 kB	crash File	02/15/16 00:07:58
WhatsApp_2016-02-12-152922_EpiPhone.crash	46.0 kB	crash File	02/15/16 00:07:58
WhatsApp_2016-02-15-133332_EpiPhone.crash	55.1 kB	crash File	02/15/16 16:33:56
WhatsApp_2016-02-16-083039_EpiPhone.crash	62.1 kB	crash File	02/16/16 13:43:39
WhatsApp_2016-02-16-134056_EpiPhone.crash	54.4 kB	crash File	02/16/16 13:43:39

Welcome to iBackupBot 1 items selected: Size: 293.0 kB

Various information of interest from a forensic point of view can be identified from the analysis of the crash logs, such as the following:

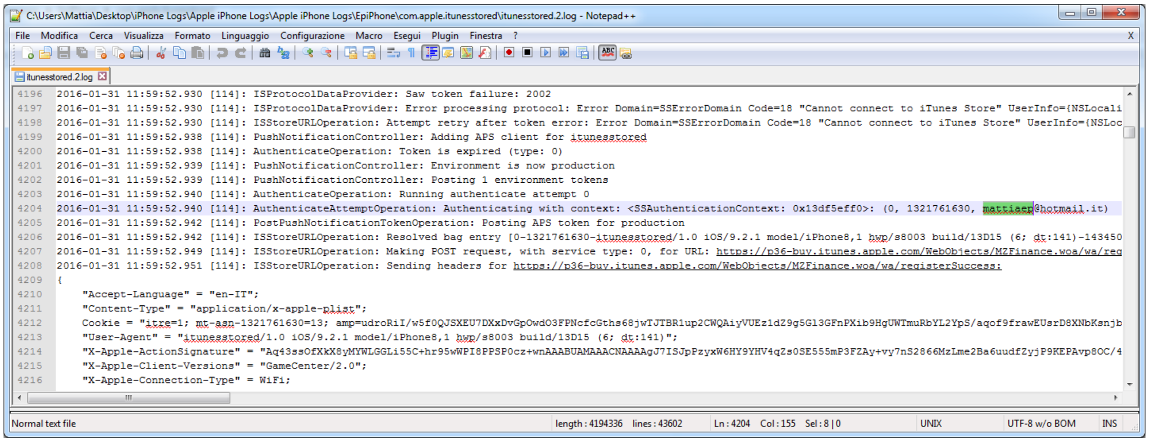
- Installed applications list and usage from various logs such as PowerLog, Security, and OnDemand



C:\Users\Mattia\AppData\Roaming\Apple Computer\Logs\CrashReporter\MobileDevice\EpiPhone\DiagnosticLogs\ondemandd\ondemandd_2016-02-15-163740.log

```
Feb 15 17:37:40  ondemandd[439] <Debug>: App did change state : net.whatsapp.WhatsApp(388) 4
Feb 15 17:37:55  ondemandd[439] <Debug>: App did change state : net.whatsapp.WhatsApp(388) 2
Feb 15 17:42:55  ondemandd[439] <Debug>: App did change state : net.whatsapp.WhatsApp(388) 8
Feb 15 17:43:54  ondemandd[439] <Debug>: App did change state : net.whatsapp.WhatsApp(388) 4
Feb 15 17:43:55  ondemandd[439] <Debug>: App did change state : com.apple.Preferences(457) 8
Feb 15 17:43:57  ondemandd[439] <Debug>: App did change state : com.apple.Preferences(457) 4
Feb 15 17:43:57  ondemandd[439] <Debug>: App did change state : com.apple.Preferences(457) 2
Feb 15 17:43:57  ondemandd[439] <Debug>: App did change state : com.apple.camera(458) 8
Feb 15 17:44:09  ondemandd[439] <Debug>: App did change state : net.whatsapp.WhatsApp(388) 2
Feb 15 17:48:04  ondemandd[439] <Debug>: App did change state : com.apple.camera(458) 4
Feb 15 17:48:04  ondemandd[439] <Debug>: App did change state : net.whatsapp.WhatsApp(388) 8
Feb 15 17:48:09  ondemandd[439] <Debug>: App did change state : com.apple.camera(458) 2
Feb 15 17:48:21  ondemandd[439] <Debug>: App did change state : net.whatsapp.WhatsApp(388) 4
Feb 15 17:48:21  ondemandd[439] <Debug>: App did change state : com.apple.mobilemail(387) 8
Feb 15 17:48:46  ondemandd[439] <Debug>: App did change state : net.whatsapp.WhatsApp(388) 8
Feb 15 17:48:47  ondemandd[439] <Debug>: App did change state : com.apple.mobilemail(387) 4
Feb 15 17:50:02  ondemandd[439] <Debug>: App did change state : com.apple.mobilecal(199) 4
Feb 15 17:50:10  ondemandd[439] <Debug>: App did change state : net.whatsapp.WhatsApp(388) 4
Feb 15 17:50:10  ondemandd[439] <Debug>: App did change state : com.apple.mobilemail(387) 8
Feb 15 17:50:11  ondemandd[439] <Debug>: App did change state : com.apple.mobilecal(199) 2
Feb 15 17:51:06  ondemandd[439] <Debug>: App did change state : com.apple.mobilemail(387) 4
Feb 15 17:51:09  ondemandd[439] <Debug>: App did change state : net.whatsapp.WhatsApp(388) 2
Feb 15 17:51:24  ondemandd[439] <Debug>: App did change state : net.whatsapp.WhatsApp(388) 8
Feb 15 17:54:03  ondemandd[439] <Debug>: App did change state : net.whatsapp.WhatsApp(388) 4
Feb 15 17:54:20  ondemandd[439] <Debug>: App did change state : net.whatsapp.WhatsApp(388) 8
Feb 15 17:54:22  ondemandd[439] <Debug>: App did change state : net.whatsapp.WhatsApp(388) 4
Feb 15 17:54:22  ondemandd[439] <Debug>: App did change state : com.apple.mobilemail(387) 8
Feb 15 17:54:47  ondemandd[439] <Debug>: App did change state : com.apple.mobilemail(387) 4
```

- iTunes username from `itunesstored.2.log`

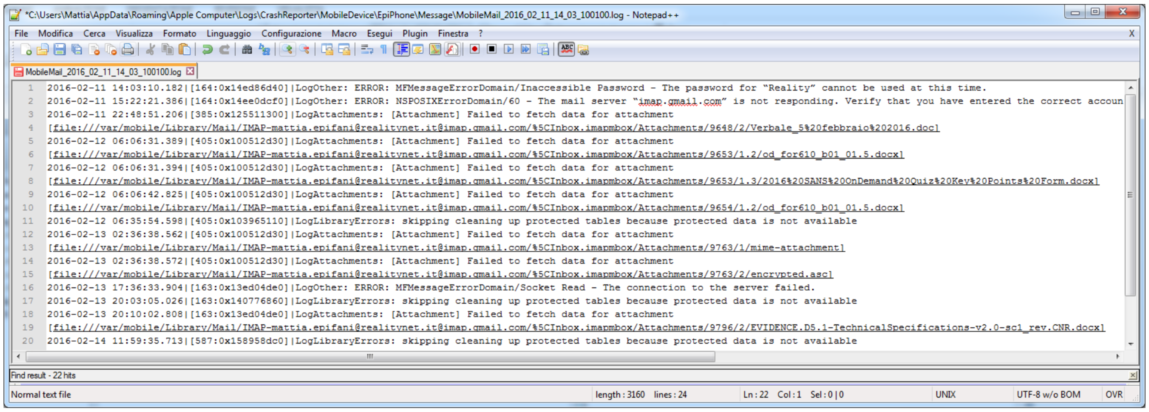


C:\Users\Mattia\Desktop\iPhone Logs\Apple iPhone Logs\Apple iPhone Logs\EpiPhone\com.apple.itunesstored\itunesstored.2.log - Notepad++

```
2016-01-31 11:59:52.930 [114]: ISProtocolDataProvider: Saw token failure: 2002
2016-01-31 11:59:52.930 [114]: ISProtocolDataProvider: Error processing protocol: Error Domain=SSErrorDomain Code=18 "Cannot connect to iTunes Store" UserInfo={NSLocali
2016-01-31 11:59:52.930 [114]: ISStoreURLOperation: Attempt retry after token error: Error Domain=SSErrorDomain Code=18 "Cannot connect to iTunes Store" UserInfo={NSLoc
2016-01-31 11:59:52.938 [114]: PushNotificationController: Adding APS client for itunesstored
2016-01-31 11:59:52.938 [114]: AuthenticateOperation: Token is expired (type: 0)
2016-01-31 11:59:52.939 [114]: PushNotificationController: Environment is now production
2016-01-31 11:59:52.939 [114]: PushNotificationController: Posting 1 environment tokens
2016-01-31 11:59:52.940 [114]: AuthenticateOperation: Running authenticate attempt 0
2016-01-31 11:59:52.940 [114]: AuthenticateAttemptOperation: Authenticating with context: <SSAuthenticationContext: 0x13df5eff0>: (0, 1321761630, mattiaep@hotmail.it)
2016-01-31 11:59:52.942 [114]: PostPushNotificationTokenOperation: Posting APS token for production
2016-01-31 11:59:52.942 [114]: ISStoreURLOperation: Resolved bag entry [0-1321761630-itunesstored/1.0 iOS/9.2.1 model/iPhone8,1 hwp/s8003 build/13D15 (6; dt:141)-143450
2016-01-31 11:59:52.949 [114]: ISStoreURLOperation: Making POST request, with service type: 0, for URL: https://p36-buy.itunes.apple.com/WebObjects/MZFinance.woa/wa/reg
2016-01-31 11:59:52.951 [114]: ISStoreURLOperation: Sending headers for https://p36-buy.itunes.apple.com/WebObjects/MZFinance.woa/wa/registerSuccess:
{
    "Accept-Language" = "en-IT";
    "Content-Type" = "application/x-apple-plist";
    Cookie = "itre=1; mt-asn-1321761630=13; amp=udroRiI/w5f0QJSXEU7DXxDvGpOwdO3FPNcfcGths68jwTJTBR1up2CWQAiyVUEz1dZ9g5Gl3GFnPXib9HgUWTmuRbYL2YpS/aqof9frawEUsrD8XNbKsnjb
    "User-Agent" = "itunesstored/1.0 iOS/9.2.1 model/iPhone8,1 hwp/s8003 build/13D15 (6; dt:141)";
    "X-Apple-ActionSignature" = "Aq43ssOfXkX8yMYWLGGLi55C+hr95wWPI8PPSP0cz+wnAAABUAMAAACNAAAAgJ7ISJpPzyxW6HY9YHV4qZs0SE555mP3FZAy+vy7nS2866MzLme2Ba6uudfZyjP9KEPAvp8OC/4
    "X-Apple-Client-Versions" = "GameCenter/2.0";
    "X-Apple-Connection-Type" = WiFi;
```

Normal text file   length : 4194336   lines : 43602   Ln : 4204   Col : 155   Sel : 8 | 0   UNIX   UTF-8 w/o BOM   INS

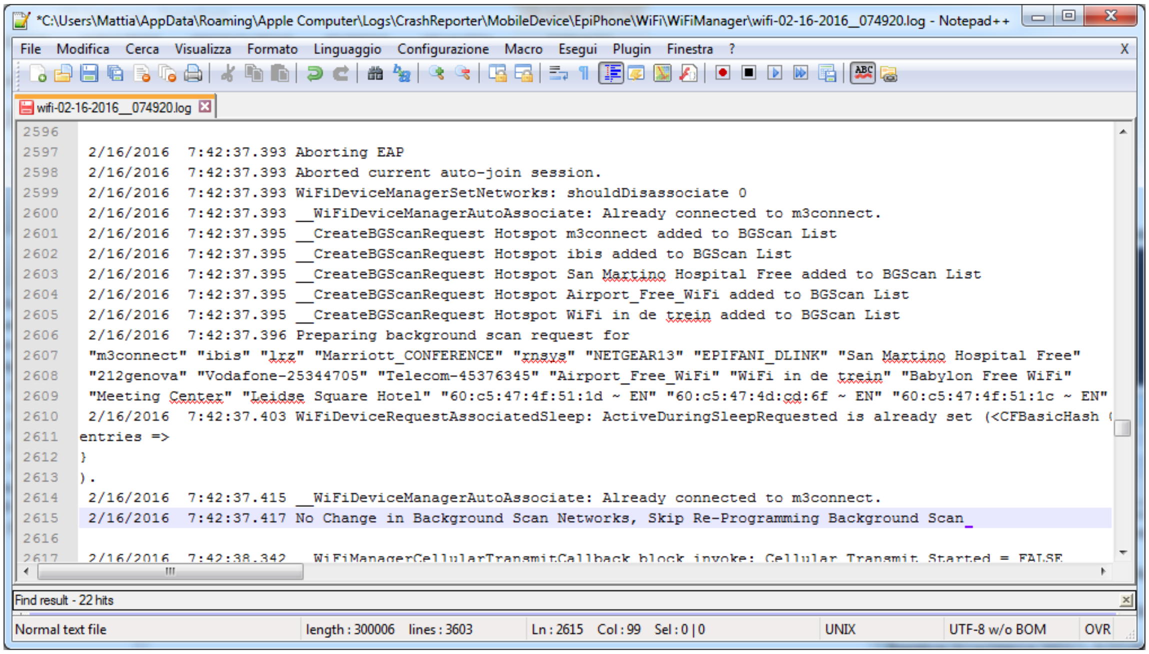
- Filename of e-mail attachments from MobileMail logs



```
2016-02-11 14:03:10.182|[164:0x14ed86d40]|LogOther: ERROR: MFMessageErrorDomain/Inaccessible Password - The password for “Reality” cannot be used at this time.
2016-02-11 15:22:21.386|[164:0x14ee0dcf0]|LogOther: ERROR: NSPOSIXErrorDomain/60 - The mail server “imap.gmail.com” is not responding. Verify that you have entered the correct accoun
2016-02-11 22:48:51.206|[385:0x125511300]|LogAttachments: [Attachment] Failed to fetch data for attachment
[file:///var/mobile/Library/Mail/IMAP-mattia.epifani@realitynet.it@imap.gmail.com/%5CInbox.imapmbox/Attachments/9648/2/Verbale_5%20febbraio%202016.doc]
2016-02-12 06:06:31.389|[405:0x100512d30]|LogAttachments: [Attachment] Failed to fetch data for attachment
[file:///var/mobile/Library/Mail/IMAP-mattia.epifani@realitynet.it@imap.gmail.com/%5CInbox.imapmbox/Attachments/9653/1.2/od_for610_b01_01.5.docx]
2016-02-12 06:06:31.394|[405:0x100512d30]|LogAttachments: [Attachment] Failed to fetch data for attachment
[file:///var/mobile/Library/Mail/IMAP-mattia.epifani@realitynet.it@imap.gmail.com/%5CInbox.imapmbox/Attachments/9653/1.3/2016%20SANS%20OnDemand%20Quiz%20Key%20Points%20Form.docx]
2016-02-12 06:06:42.825|[405:0x100512d30]|LogAttachments: [Attachment] Failed to fetch data for attachment
[file:///var/mobile/Library/Mail/IMAP-mattia.epifani@realitynet.it@imap.gmail.com/%5CInbox.imapmbox/Attachments/9654/1.2/od_for610_b01_01.5.docx]
2016-02-12 06:35:54.598|[405:0x103965110]|LogLibraryErrors: skipping cleaning up protected tables because protected data is not available
2016-02-13 02:36:38.562|[405:0x100512d30]|LogAttachments: [Attachment] Failed to fetch data for attachment
[file:///var/mobile/Library/Mail/IMAP-mattia.epifani@realitynet.it@imap.gmail.com/%5CInbox.imapmbox/Attachments/9763/1/mime-attachment]
2016-02-13 02:36:38.572|[405:0x100512d30]|LogAttachments: [Attachment] Failed to fetch data for attachment
[file:///var/mobile/Library/Mail/IMAP-mattia.epifani@realitynet.it@imap.gmail.com/%5CInbox.imapmbox/Attachments/9763/2/encrypted.asc]
2016-02-13 17:36:33.904|[163:0x13ed04de0]|LogOther: ERROR: MFMessageErrorDomain/Socket Read - The connection to the server failed.
2016-02-13 20:03:05.026|[163:0x140776860]|LogLibraryErrors: skipping cleaning up protected tables because protected data is not available
2016-02-13 20:10:02.808|[163:0x13ed04de0]|LogAttachments: [Attachment] Failed to fetch data for attachment
[file:///var/mobile/Library/Mail/IMAP-mattia.epifani@realitynet.it@imap.gmail.com/%5CInbox.imapmbox/Attachments/9796/2/EVIDENCE.D5.1-TechnicalSpecifications-v2.0-sc1_rev.CNR.docx]
2016-02-14 11:59:35.713|[587:0x158958dc0]|LogLibraryErrors: skipping cleaning up protected tables because protected data is not available
```

*C:\Users\Mattia\AppData\Roaming\Apple Computer\Logs\CrashReporter\MobileDevice\EpiPhone\Message\MobileMail_2016_02_11_14_03_100100.log - Notepad++
File Modifica Cerca Visualizza Formato Linguaggio Configurazione Macro Esegui Plugin Finestra ?
MobileMail_2016_02_11_14_03_100100.log
Find result - 22 hits
Normal text file
length : 3160 lines : 24
Ln : 22 Col : 1 Sel : 0 | 0
UNIX
UTF-8 w/o BOM
OVR

- List of Wi-Fi network and history of latest connections from Wi-Fi logs



```

 2/16/2016  7:42:37.393 Aborting EAP
 2/16/2016  7:42:37.393 Aborted current auto-join session.
 2/16/2016  7:42:37.393 WiFiDeviceManagerSetNetworks: shouldDisassociate 0
 2/16/2016  7:42:37.393 __WiFiDeviceManagerAutoAssociate: Already connected to m3connect.
 2/16/2016  7:42:37.395 __CreateBGScanRequest Hotspot m3connect added to BGScan List
 2/16/2016  7:42:37.395 __CreateBGScanRequest Hotspot ibis added to BGScan List
 2/16/2016  7:42:37.395 __CreateBGScanRequest Hotspot San Martino Hospital Free added to BGScan List
 2/16/2016  7:42:37.395 __CreateBGScanRequest Hotspot Airport_Free_WiFi added to BGScan List
 2/16/2016  7:42:37.395 __CreateBGScanRequest Hotspot WiFi in de trein added to BGScan List
 2/16/2016  7:42:37.396 Preparing background scan request for
 "m3connect" "ibis" "lrz" "Marriott_CONFERENCE" "rnsys" "NETGEAR13" "EPIFANI_DLINK" "San Martino Hospital Free"
 "212genova" "Vodafone-25344705" "Telecom-45376345" "Airport_Free_WiFi" "WiFi in de trein" "Babylon Free WiFi"
 "Meeting Center" "Leidse Square Hotel" "60:c5:47:4f:51:1d ~ EN" "60:c5:47:4d:cd:6f ~ EN" "60:c5:47:4f:51:1c ~ EN"
 2/16/2016  7:42:37.403 WiFiDeviceRequestAssociatedSleep: ActiveDuringSleepRequested is already set (<CFBasicHash (
entries =>
}
).
 2/16/2016  7:42:37.415 __WiFiDeviceManagerAutoAssociate: Already connected to m3connect.
 2/16/2016  7:42:37.417 No Change in Background Scan Networks, Skip Re-Programming Background Scan

 2/16/2016  7:42:38.342  __WiFiManagerCellularTransmitCallback_block_invoke: Cellular Transmit Started = FALSE
```

*C:\Users\Mattia\AppData\Roaming\Apple Computer\Logs\CrashReporter\MobileDevice\EpiPhone\WiFi\WiFiManager\wifi-02-16-2016__074920.log - Notepad++
File Modifica Cerca Visualizza Formato Linguaggio Configurazione Macro Esegui Plugin Finestra ?
wifi-02-16-2016__074920.log
Find result - 22 hits
Normal text file
length : 300006 lines : 3603
Ln : 2615 Col : 99 Sel : 0 | 0
UNIX
UTF-8 w/o BOM
OVR

Tracking device usage

Sarah Edwards has made an extensive research work on artifacts that track the usage of the device. By linking applications, data and network usage, health information (for example, workouts), timestamps, and geolocation data, it is indeed possible to understand what a given user under investigation was doing and where at a given point in time. In particular, she looked at the following artifacts:

- **CoreDuet**: `/private/var/mobile/Library/CoreDuet/`

  `coreduetd.db` (31 tables)

  `coreduetdClassA.db` (31 tables)

  `coreduetdClassD.db` (31 tables)

  `Knowledge/knowledgeC.db` (5 tables)

  `People/interactionC.db` (9 tables)

- **Battery Life (PowerLog)**: `/private/var/mobile/Library/BatteryLife/`

  `CurrentPowerlog.PLSQL` (257 tables)

  `Archives/powerlog_YYYY-MM-DD_XXXXXXXX.PLQSQL.gz` (Previous ~5 Days)

- **Health**: `/private/var/mobile/Library/Health/`

  `healthdb.sqlite` (11 tables)

  `healthdb_secure.sqlite` (16 tables)

- **Aggregate Dictionary**: `/private/var/mobile/Library/AggregateDictionary/`

  `ADDataStore.sqlitedb` (4 tables)

- **networkd**: `/private/var/networkd/`

  `netusage.sqlite` (13 tables)

- **routined**: `/private/var/mobile/Library/Caches/com.apple.routined/`

  `cache_encryptedB.db` (5 tables)

  `StateModel1.archive`

  `StateModel2.archive`

- **locationd**: `/private/var/root/Library/Caches/locationd/`

  `cache_encryptedA.db` (79 tables)

  `lockCache_encryptedA.db` (51 tables)

  `cache_encryptedB.db` (167 tables)

  `cache_encryptedC.db` (9 tables)

The details of Sarah's work mentioned here can be found at her website, more specifically the *The iOS of Sauron: How iOS Tracks Everything You Do* talk (`https://www.mac4n6.com/resources/`).

Third-party application analysis

In the previous paragraphs, you have seen where important artifacts related to the iOS system settings and preferences, native iOS applications, and device features are located. These are locations to be aware of, and it is important to know how to analyze them since they are common to all iDevices. Now, in the following paragraphs, we are going to show you a practical analysis of some of the most used third-party applications.

Social Network and Instant Messaging applications

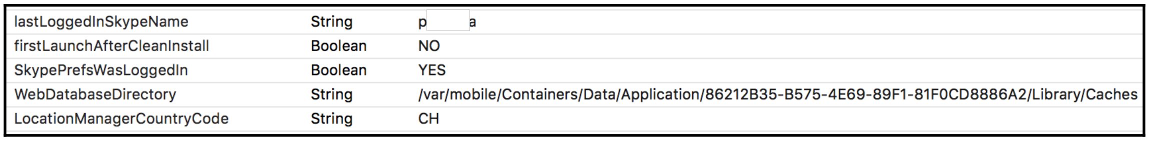
Probably the most widely used category, Social Network and Instant Messaging applications can represent a treasure trove for the analyst, as now they have become the main means for communication. In this section, we will give you an overview of the main artifacts you may find in some of the most popular applications, as in which location you should pay particular attention during the analysis.

Skype

Skype is probably the most known and used software for VoIP and chatting:

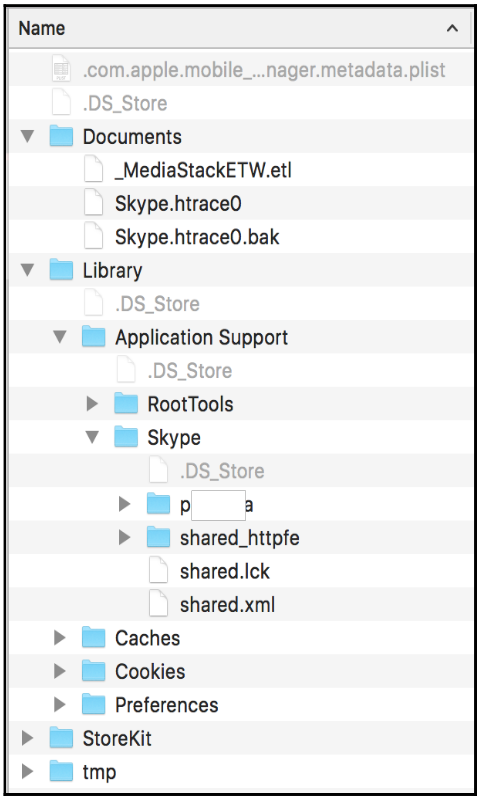
```
# tree -L 3 86212B35-B575-4E69-89F1-81F0CD8886A2/

|-- Documents
|   |-- Skype.htrace0
|   |-- Skype.htrace0.bak
|   `-- _MediaStackETW.etl
|-- Library
|   |-- Application\ Support
|   |   |-- RootTools
|   |   `-- Skype
|   |-- Caches
|   |   |-- Snapshots
|   |   |-- com.plausiblelabs.crashreporter.data
|   |   |-- com.skype.skype
|   |   |-- net.hockeyapp.sdk.ios
|   |   |-- offline-storage-ecs.data
|   |   |-- offline-storage.data
|   |   |-- <skype_name>
|   |   |-- skype-cache-501.<skype_name>.ContactsListAdapter-Contacts-
SingleSelect.plist
|   |   |-- skype-cache-501.<skype_name>.ContactsListAdapter-Favorites-
SingleSelect.plist
...
|   |   |-- skype-cache-501.<skype_name>.chathistory-viewmodels
<contact_name>.plist
...
|   |   |-- skype-cache-501.<skype_name>.chatinput-saved-
<contact_name>.plist
|   |   `-- skype-cache-501.<skype_name>.conversations.plist
 |   |-- Cookies
|   |   `-- Cookies.binarycookies
|   `-- Preferences
|       `-- com.skype.skype.plist
|-- StoreKit
|   `-- receipt
`-- tmp
```

Starting from the `Library/Preferences` folder, we can find the first important information inside the `com.skype.skype.plist` file, the username, as shown in the following screenshot:



lastLoggedInSkypeName	String	p[redacted]a
firstLaunchAfterCleanInstall	Boolean	NO
SkypePrefsWasLoggedIn	Boolean	YES
WebDatabaseDirectory	String	/var/mobile/Containers/Data/Application/86212B35-B575-4E69-89F1-81F0CD8886A2/Library/Caches
LocationManagerCountryCode	String	CH

However, the preceding screenshot shows only the last username that has logged in. If we want to know all the profiles that have been logged in from this device, we have to look for other folders under `Library/Application Support/Skype/`, where we will find one folder for each account logged in with that device:



```
Name
  .com.apple.mobile_...nager.metadata.plist
  .DS_Store
▼ Documents
    _MediaStackETW.etl
    Skype.htrace0
    Skype.htrace0.bak
▼ Library
    .DS_Store
  ▼ Application Support
      .DS_Store
    ▶ RootTools
    ▼ Skype
        .DS_Store
      ▶ p[redacted]a
      ▶ shared_httpfe
        shared.lck
        shared.xml
  ▶ Caches
  ▶ Cookies
  ▶ Preferences
▶ StoreKit
▶ tmp
```

Inside every user's folder, we find all the databases storing information such as contacts list, chats, and so on. Here, the structure is pretty much the same as the PC/desktop version. In fact, you can open the `main.db` file, where you can find all information clearly stored, as you can see from the interesting names of the tables as follows:

```
# sqlite3 main.db

SQLite version 3.8.10.2
Enter ".help" for instructions
sqlite> .mode line
sqlite> .tables
Accounts            Chats               MediaDocuments
Translators
Alerts              ContactGroups       MessageAnnotations
VideoMessages
AppSchemaVersion    Contacts            Messages
Videos
CallMembers         Conversations       Participants
Voicemails
Calls               DbMeta              SMSes
tracker_journal
ChatMembers         LegacyMessages      Transfers
```

Refer to the following screenshot:

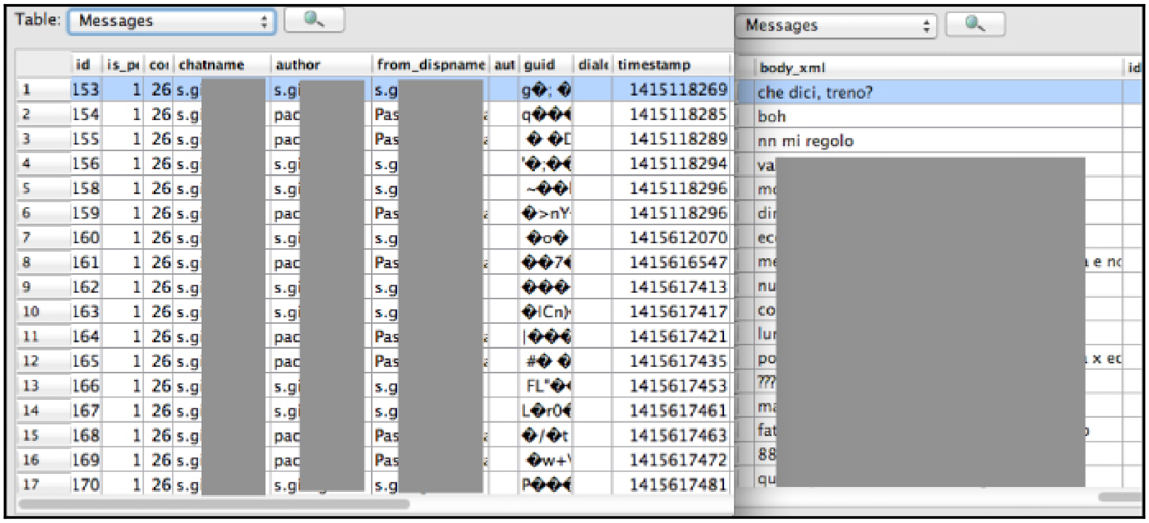


Table: Messages

	id	is_p	co	chatname	author	from_dispname	aut	guid	dial	timestamp
1	153	1	26	s.g [redacted]	s.gi [redacted]	s.g [redacted]		g�; �		1415118269
2	154	1	26	s.g [redacted]	pac [redacted]	Pas [redacted]		q���		1415118285
3	155	1	26	s.g [redacted]	pac [redacted]	Pas [redacted]		� �[		1415118289
4	156	1	26	s.g [redacted]	s.gi [redacted]	s.g [redacted]		'�;��		1415118294
5	158	1	26	s.g [redacted]	s.gi [redacted]	s.g [redacted]		~��I		1415118296
6	159	1	26	s.g [redacted]	pac [redacted]	Pas [redacted]		�>nY		1415118296
7	160	1	26	s.g [redacted]	s.gi [redacted]	s.g [redacted]		�o�		1415612070
8	161	1	26	s.g [redacted]	pac [redacted]	Pas [redacted]		��7�		1415616547
9	162	1	26	s.g [redacted]	s.gi [redacted]	s.g [redacted]		���		1415617413
10	163	1	26	s.g [redacted]	s.gi [redacted]	s.g [redacted]		�ICn)		1415617417
11	164	1	26	s.g [redacted]	pac [redacted]	Pas [redacted]		|���		1415617421
12	165	1	26	s.g [redacted]	pac [redacted]	Pas [redacted]		#� �		1415617435
13	166	1	26	s.g [redacted]	s.gi [redacted]	s.g [redacted]		FL"��		1415617453
14	167	1	26	s.g [redacted]	s.gi [redacted]	s.g [redacted]		L�r0�		1415617461
15	168	1	26	s.g [redacted]	pac [redacted]	Pas [redacted]		�/�t		1415617463
16	169	1	26	s.g [redacted]	pac [redacted]	Pas [redacted]		�w+\		1415617472
17	170	1	26	s.g [redacted]	s.gi [redacted]	s.g [redacted]		P���		1415617481

Messages

body_xml	id
che dici, treno?	
boh	
nn mi regolo	
va [redacted]	
mo [redacted]	
di [redacted]	
ec [redacted]	
me [redacted] e nc	
nu [redacted]	
co [redacted]	
lu [redacted]	
po [redacted] x ec	
??? [redacted]	
ma [redacted]	
fat [redacted]	
88 [redacted]	
qu [redacted]	

This means that you can use any of your favorite Skype analysis utilities to parse these files, such as **SkypeLogView** from **Nirsoft** and so on. Finally, within the application folder itself, you may also find the Voicemail messages and screenshots, as we have addressed previously in the *Snapshots* section, in addition to files transferred via Skype, logs, and much more.

WhatsApp

Although it is technically an Instant Messaging application, WhatsApp has almost completely replaced the classical SMS. Therefore, it is very likely that you will encounter it during a mobile forensics analysis. Let's have a look at its internal folder structure that, as you may have realized, differs really very little from one application to the other:

```
# tree -L 3 6E3B21B5-9E07-4F65-B7FD-57E12CEF9E2C/
|-- Documents |   |-- StatusMessages.plist
|   |-- SyncHistory.plist
|   |-- blockedcontacts.dat
|   |-- calls.backup.log
|   `-- calls.log
|-- Library
|   |-- Caches
|   |   |-- ProfilePictures
|   |   |-- Snapshots
|   |   `-- mmap-images
|   |-- FieldStats
|   |   `-- fieldstats.active
|   |-- Logs
|   |   |-- Handoff
|   |   |-- whatsapp-2016-07-29-20-58-09.227.7.log
|   |   `-- whatsapp-2016-07-30-18-50-21.581.8.log
|   `-- Preferences
|       `-- net.whatsapp.WhatsApp.plist
|-- StoreKit
|   `-- receipt
`-- tmp
```

We have now understood that to get a first hint and useful information for starting with an application, we may want to start looking inside the `plist` configuration file under `Library/Preferences/`. In this case, we are looking for `net.whatsapp.WhatsApp.plist`. Here again, you will find some basic information, such as the username, the phone number the WhatsApp account was linked to, and so on.

However, what we may realize here is that we don't have the chat database, the famous `ChatStorage.sqlite` file. This is because Whatsapp is one of those applications that stores the data in the `/private/var/mobile/Containers/Shared/AppGroup/` folder:

```
iLab1:/private/var/mobile/Containers/Shared/AppGroup root# tree
332A098D-368C-4378-A503-91BF33284D4B

|-- Axolotl.sqlite
|-- ChatSearch.sqlite
|-- ChatStorage.sqlite
|-- Contacts.sqlite
|-- Jobs.sqlite
|-- Jobs.sqlite-shm
|-- Jobs.sqlite-wal
|-- Library
|   |-- Caches
|   `-- Preferences
|       `-- group.net.whatsapp.WhatsApp.shared.plist
|-- Media
|   `-- Profile
|       |-- 393205573087-1420495528.thumb
|       |-- 393205573087-1456251558-1456251649.thumb
|       |-- 393497320069-1432382263.thumb
|       |-- 393930831043-1402697710.thumb
|       |-- 41788236232-1467887746-1467901783.thumb
|       `-- 41788236232-1468256570.jpg
|-- cck.dat
`-- connection.lock
```

Regarding the actual content of the messages exchanged, the main database is `Documents/ChatStorage.sqlite`, whose structure is as follows:

```
    # sqlite3 ChatStorage.sqlite
    SQLite version 3.8.10.2
    Enter ".help" for instructions
    sqlite> .mode line
    sqlite> .tables
    ZWABLACKLISTITEM       ZWAGROUPMEMBER         ZWAMESSAGEINFO
    ZWACHATPROPERTIES      ZWAGROUPMEMBERSCHANGE  ZWAPROFILEPICTUREITEM
    ZWACHATPUSHCONFIG      ZWAMEDIAITEM           Z_METADATA
    ZWACHATSESSION         ZWAMESSAGE             Z_MODELCACHE
    ZWAGROUPINFO           ZWAMESSAGEDATAITEM     Z_PRIMARYKEY
```

The ZWAMESSAGE table is the one containing the messages exchanged, their timestamp, the name of who the user was chatting with, and so on, as shown in the following screenshot:

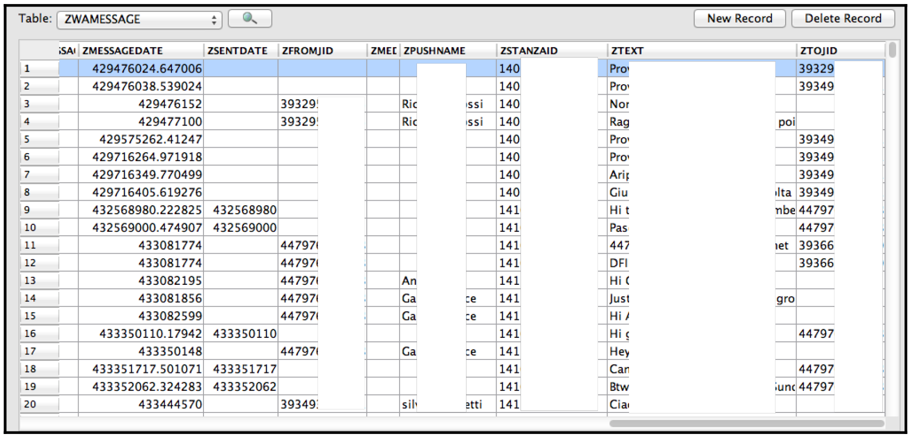


Table: ZWAMESSAGE

	SA	ZMESSAGEDATE	ZSENTDATE	ZFROMJID	ZMED	ZPUSHNAME	ZSTANZAID	ZTEXT	ZTOJID
1		429476024.647006					140	Pro	39329
2		429476038.539024					140	Pro	39349
3		429476152		39329		Ric ssi	140	Nor	
4		429477100		39329		Ric ssi	140	Rag poi	
5		429575262.41247					140	Pro	39349
6		429716264.971918					140	Pro	39349
7		429716349.770499					140	Ari	39349
8		429716405.619276					140	Giu lta	39349
9		432568980.222825	432568980				141	Hi t mbe	44797
10		432569000.474907	432569000				141	Pas	44797
11		433081774		44797			141	447 et	39366
12		433081774		44797			141	DFI	39366
13		433082195		44797		An	141	Hi	
14		433081856		44797		Ga ce	141	Just gro	
15		433082599		44797		Ga ce	141	Hi	
16		433350110.17942	433350110				141	Hi	44797
17		433350148		44797		Ga ce	141	Hey	
18		433351717.501071	433351717				141	Car	44797
19		433352062.324283	433352062				141	Btw unc	44797
20		433444570		39349		sil etti	141	Cia	

The ZWACHATSESSION table stores information about the open chats, both with a single user and group chats, and you can correlate these data with those in the ZWAGROUPMEMBER and ZWAGROUPINFO tables in order to find out which users belong to which group chat. Finally, in ZWAMEDIAITEM stores references to the multimedia files (pictures, audio messages, and videos) exchanged, with an indication of the user involved, timestamps, and the location where the multimedia file has been stored within an iDevice.

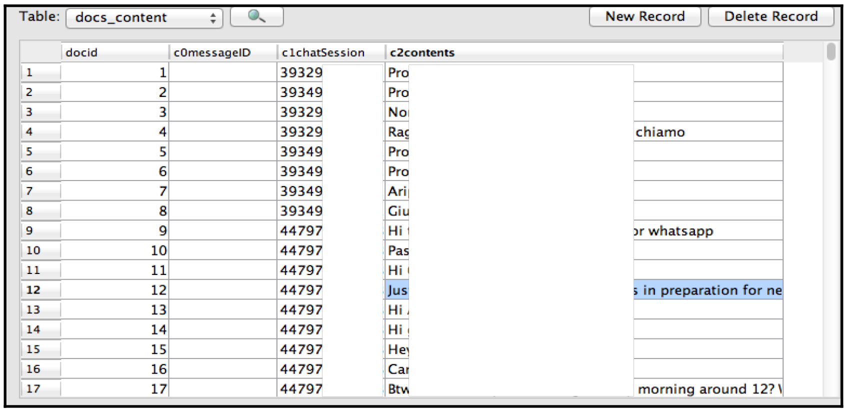


Table: docs_content

	docid	c0messageID	c1chatSession	c2contents
1	1		39329	Pro
2	2		39349	Pro
3	3		39329	No
4	4		39329	Rag chiamo
5	5		39349	Pro
6	6		39349	Pro
7	7		39349	Ari
8	8		39349	Giu
9	9		44797	Hi r whatsapp
10	10		44797	Pas
11	11		44797	Hi
12	12		44797	Jus s in preparation for ne
13	13		44797	Hi
14	14		44797	Hi
15	15		44797	Hey
16	16		44797	Car
17	17		44797	Btw morning around 12? \

However, you will also find the chat contents inside `Documents/ChatSearch.sqlite` within the `docs_content` tables, as shown in the preceding screenshot.

Facebook and Messenger

Facebook is the most known and widely used social network. For this reason, other than for the fact that it is now integrated with iOS, you will most likely have to analyze the Facebook app in almost all of your investigations. As you can imagine, the amount of information stored by Facebook is very high, and in particular, it concerns three areas, user personal information, a cache of images related to profiles and visited pages, and information related to the external sites visited within the Facebook app through the links present on the posts. Due to the obviously large amount of possible information retrievable in the Facebook app and the page limitation of a book, the goal of this section is to give you a glimpse and some hints on possible artifacts and where to find them.

The account information is saved inside `Library/Preferences/com.facebook.Facebook.plist`. Among other information, you will find the e-mail address and the Facebook ID of the profile configured within the app, as well as the date of the last time the app has been used.

Information related to contacts is saved in `Library/Caches/_store_<ID>/messender_contacts_v1/fbsyncstore.db` within the `people` table.

In `Library/Caches/_store_<ID>/image_cache_v7/` are stored images viewed while surfing through the pages of the social network (for example, posts of other users and so on), while inside the `Library/WebKit/WebsiteData/LocalStorage/StorageTracker.db` database and the `Library/Caches/WebKit/NetworkCache/` folder are stored contents of other websites visited through the application internal browser, including the related URL and the corresponding files (for example, the JPG image, the HTML page, the CSS style sheet, and so on).

Starting from April 2014, the instant messaging function has been completely moved from the Facebook app to Facebook Messenger app, launched by Facebook itself on August 2009. Therefore, all Facebook users have been forced to download and use Messenger to communicate among themselves.

Also in this case, interesting artifacts are stored in both the `.../Containers/Data/...` and `.../Containers/Shared/...` application folders. In particular, you will find the chat content database at `/private/var/mobile/Containers/Shared/AppGroup/<UUID>/_store_<ID>/messenger_messages_v1/orca2.db`. In this database, chats are saved as a blob data field within the tables.

Telegram

Telegram is probably the second most used Instant Messaging application after Whatsapp, therefore chances of having to deal with it during an analysis are quite high as well. As always, the account information can be found at `<App_folder>/Library/Preferences/` within the `ph.telegra.Teleraph.plist` file.

Similar to WhatsApp, Facebook, and Messenger, Telegram stores many of its data in the `.../Container/Shared/AppGroup/` folder, as we can see from the following folder structure:

```
-- 9C8C5F63-4EF3-4D6C-9F7F-6C71B9A0970A

|   |-- Caches
|   |   |-- com.plausiblelabs.crashreporter.data
|   |   |   `-- ph.telegra.Telegraph
|   |   |       `-- queued_reports
|   |   |-- net.hockeyapp.sdk.ios
|   |   `-- ph.telegra.Telegraph
|   |       `-- com.apple.opengl
|   |           |-- compileCache.data
|   |           |-- compileCache.maps
|   |           |-- linkCache.data
|   |           |-- linkCache.maps
|   |           |-- shaders.data
|   |           `-- shaders.maps
|   |-- Documents
|   |   |-- mtkeychain
|   |   |   |-- Telegram_meta.bin
|   |   |   |-- Telegram_persistent.bin
|   |   |   |-- Telegram_primes.bin
|   |   |   `-- Telegram_temp.bin
|   |   |-- tgdata.db
|   |   |-- tgdata.db-journal
|   |   |-- tgdata_index.db
|   |   |-- tgdata_index.db-journal
|   |   `-- wallpaper-data
|   |       `-- _currentWallpaper.jpg
|   |-- Library
|   |   `-- Caches
|   `-- shared-auth-info
```

More specifically, under the `Documents` folder we find the `tgdata.db` database, which contains all information about contacts, conversations, exchanged files, and so on.

Some of the tables of particular interest are as follows:

- `users_v29`: This contains the contact list and related Telegram user ID (uid).
- `contacts_v29`: This contains the list of all uid.
- `messages_v29`: This contains the list of all messages exchanged, with to/from fields to identify who is the source and who is the destination, as well as a conversation ID (cid), so that it is easier to group messages belonging to the same conversation. Messages from secret chats are also clearly stored here.
- `convesations_v29`: This contains the list of active conversations as seen in the **Chats** screen of the app, and also the last message exchanged. The name of the table does not contain a typo from us, but it really misses the *r* !
- `encrypted_cids_v29`: This contains the conversation IDs of the secret chats. All cid related to secret chats start with the `-` (minus) symbol.

Those listed here are just some of the interesting tables, to give you a glimpse of what you can find, but as you can see from the preceding `sqlite3` command output there are many more:

```
# sqlite3 tgdata.db
SQLite version 3.8.10.2
Enter ".help" for instructions
sqlite> .mode line
sqlite> .tables
actions_v29                              links_v29
assets_v29                               local_files_v29
blacklist_v29                            media_cache_v29
botinfo_29                               media_v29
broadcast_conversations_v29              messages_v29
channel_cached_v29                       outbox_v29
channel_conversations_v29                peer_cached_v29
channel_delete_messages_v29              peer_incoming_actions_29
channel_history_version_v29              peer_incoming_encrypted_actions_29
channel_leave_v29                        peer_outgoing_actions_29
channel_message_holes_v29                peer_outgoing_actions_resend_29
channel_message_tags_v29                 peer_photos_v29
channel_message_unimportant_groups_v29   peer_rating_29
channel_message_unimportant_holes_v29    peers_v29
channel_messages_randomid_v29            random_ids_v29
channel_messages_v29                     secret_media_v29
channel_pending_messages_v29             selfdestruct_v29
channel_read_history2_v29                server_assets_v29
contacts_v29                             service_v29
convesations_v29                         shared_media_index_29
dismiss_report_spam_v29                  shared_media_index_built_29
encrypted_cids_v29                       shared_media_index_downloaded2_29
file_deletion_v29                        storedFiles_v29
files_v29                                tempMessages_v29
future_v29                               user_cached_v29
history_holes_29                         users_v29
```

Signal

Last but not least, we could not exclude**Signal** from this list of IM apps analyzed. Unfortunately for the forensics analyst and fortunately for the end user, there is not much to say as Signal does indeed what it promises, all messages are securely encrypted within the device (data-at-rest) other than when sent/received through the network (data-in-transit). The folders structure is as follows:

```
iLab1:/private/var/mobile/Containers/Data/Application/6B6D8B22-
F0D1-47A2-8818-134AE95B2DEC root# tree
|-- Documents
|   |-- Attachments
|   |   `-- 4034125268946418778.jpeg
|   |-- Signal.sqlite
|   |-- Signal.sqlite-shm
|   `-- Signal.sqlite-wal
|-- Library
|   |-- Caches
|   |   |-- Logs
|   |   |   `-- org.whispersystems.signal\ 2016-07-31\ 15-59.log
|   |   |-- Snapshots
|   |   |   `-- org.whispersystems.signal
|   |   |       |-- 52A21AE5-A070-40AF-B91B-711C86867100@2x.png
|   |   |       |-- A8B33136-73B0-4FBB-9AA4-436A0557C595@2x.png
|   |   |       `-- downscaled
|   |   |           `-- 90479DEB-F853-4097-9C31-5F1B58617EBA@2x.png
|   |   `-- org.whispersystems.signal
|   |       |-- Cache.db
|   |       |-- Cache.db-shm
|   |       |-- Cache.db-wal
|   |       `-- fsCachedData
|   |           `-- 691B3FAC-2F2A-435D-9A23-B14DCB253298
|   `-- Preferences
|       `-- org.whispersystems.signal.plist
|-- StoreKit
`-- tmp
```

All of Signal's data (that is, chat messages, contact lists, and so on) is inside the `Document/Signal.sqlite` database that is, as we already said, encrypted. However, there are two things that are not encrypted. They are as follows:

- **Attachments**: Oddly enough, the attachments exchanged are clearly stored and can be retrieved from the `Document/Attachments/` folder.
- **Snapshots**: Signal has an **Enable Screen Security** option that would prevent this, but for some reason is not set by default. Therefore, it is possible to find the screen snapshot (we have talked about snapshots in an earlier section of this chapter) in the `Library/Caches/Snapshots/` folder.

As we said, not too much information is available, but keep in mind the attachments and the snapshots, as they may reveal important traces for your investigation.

Cloud storage applications

Cloud storage applications are very popular on mobile devices, since the Cloud somewhat extends the device storage capability and allows the user to have access to their data anywhere and anytime. Therefore, it is very probable that you will encounter at least one on this class of apps during your analysis. In this section, we just want to give you a glimpse of some artifacts you can find in two of the most popular cloud storage services.

Dropbox

The Dropbox iOS app is stored in `/private/var/mobile/Containers/Data/Applications/9238CCE2-5C38-4843-9F76-D73B7C0CAB23/`. In the `Documents/Users/<user_id>/` folder there are several databases of interest, in particular the following:

- `Dropbox.sqlite`: This database, within the `ZCACHEDFILE` table, contains information about all the files the user interacted with, such as opened, saved, or simply browsed
- `Uploads.sqlite`: This database, within the `ZQUEUEDUPLOAD` table, contains the name/path of uploaded files, the file mime type, and timestamp of the upload

The `Library/Caches/` folder contains a local copy of the *opened files*, but it is *available only if we can perform a physical acquisition* (not logical/backup acquisition).

Finally, another interesting folder is `Library/Application Support/Dropbox/<hex_ID>/Account/contact_cache/`, where two very interesting json files, among others, are present:

- `me`: This contains information about the account holder such as name, surname, and e-mail address registered with the current account.
- `all_searchable`: This contains account information such as name, surname, and e-mail address of every accounts the user has ever shared a file with. This list also has historical records referring to account the user does not currently share files with anymore, but have done so in the past.

Google Drive

The Google Drive iOS App is in `/private/var/mobile/Containers/Data/Application/F86EB4B3-5DB1-4C72-B576-546694440A5D/`. The folder structure is the usual, with `Documents` and `Library`, being the main folders where you can find interesting artifacts. More specifically, within `Documents/drivekit/users/<user_ID>/cello/cello.db`, you will find the database containing information about the Drive folders structure, files accessed, files moved to the Bin (with the timestamp) or definitely deleted (without the timestamp of when it happened), and so on, as well as a copy of the files saved into **My Drive** or as **Available Offline**.

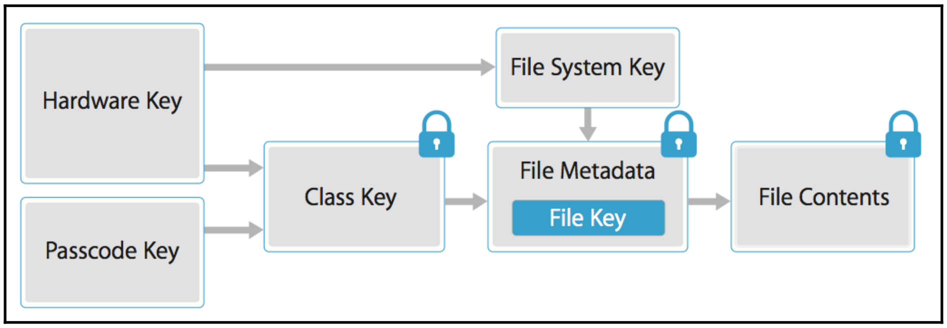
The `Library/Caches/com.google.commmon.SSO/<user_ID>/` folder contains, among others, the `Profile.plist` file that, as the name suggests, shows where the used information is stored such as name, surname, username, and e-mail address.

Deleted data recovery

In this section, we will give you a quick overview on the difficulties of performing file carving operations on an iOS device and will help you understand why and what are the possibilities. We will also see the particular case of recovering the SQLite deleted records.

File carving – is it feasible?

Apple uses a technology called**Data Protection** in order to further protect data stored in flash memory on iDevices. Every time a file is created, a new 256-bit *per-file key* is generated and it is used to encrypt the file content using AES encryption. The *per-file key* is then wrapped with one of the data protection class keys and then stored in the file's metadata, which is in turn encrypted with the filesystem key (the **EMF key**), which is generated from the unique hardware UID. The following diagram, which is taken directly from the Apple iOS Security official paper (last update on May 2016, refer to `Appendix A`, *References*), summarizes the entire process:



With this premise, it is clear that the classic file carving procedure will not work, since in the unallocated space there will only be encrypted content. An interesting approach on how to carve deleted images from the iOS devices has been published by D'Orazio and others (refer to `Appendix A`, *References*). What they suggest is to exploit the journaling feature of the iOS filesystem, HFS+. In fact, by analyzing and comparing both the catalog and journal files of the HFS+ filesystem, it could be possible to identify information about deleted files, such as file and metadata locations, their timestamp, and so on. Based on this information from the journal, the analyst should be able to search and recover the deleted files, locate the cryptography keys, and then decrypt the image file. Heather Mahalik (her Twitter handle is `@HeatherMahalik`) also describes a similar approach in her book, *Practical Mobile Forensics*, by *Heather Mahalik*, published by *Packt Publishing*. Of course, such approaches require physical acquisition to be possible for the target device.

However, that approach may work only if the device has not been restored, wiped, or upgraded to a new OS version, because in such cases, the filesystem key (EMF) would be erased and a new key recreated. Therefore, without the original EMF key, all contents in the unallocated space referring to a period prior to the restoring/wiping/upgrading is gone forever.

Carving SQLite deleted records

We will not go into the details of the SQLite structure (for more information, refer to `Appendix A`, *References*), since it is out of the scope of this book. However, it is important for you to know that other than deleted files, it is also possible to recover deleted records within the SQLite databases. Mari DeGrazia (her Twitter handle is `@maridegrazia`) has developed a useful Python script that parses the database file and carves out deleted entries. Its usage is as simple as running a single-line command as follows:

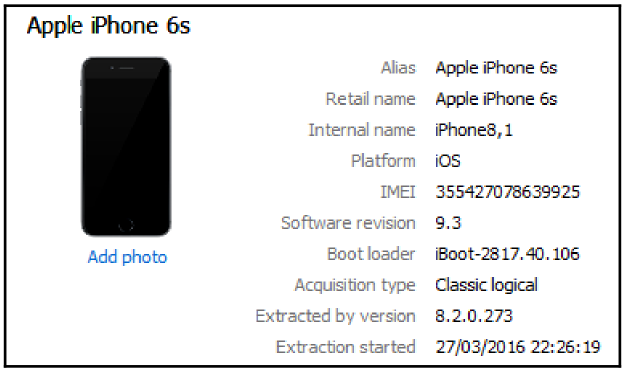
```
$ python sqlparse.py -f mmssms.db -r -o report.txt
```

You can find it on her website and GitHub repository; she has also provided a GUI version of the tool (refer to `Appendix A`, *References*, and `Appendix B`, *Tools for iOS Forensics*). Moreover, it is always useful to run a `strings` command on the database file as well. You may be able to recover a portion of deleted entries content that may have been missed by the tools.

Case study – iOS analysis with Oxygen Forensics

The acquisition of an iPhone made using Oxygen Forensics can be analyzed directly within the same tool. In fact, during the acquisition, all the files are parsed by the software, which offers the user a complete GUI to access and search for information in the data. The following screenshot used to show the different functionalities of the software, refers to a logical classic type of acquisition from an iPhone 6s with iOS 9.3. Some descriptions of the features of the Oxygen Forensics Suite have been taken directly from the vendor website at `http://www.oxygen-forensic.com`.

The following screenshot summarizes the main information related to the acquired device model, operating system version, serial number, acquisition type, extraction date, investigator name, case number, and evidence number:

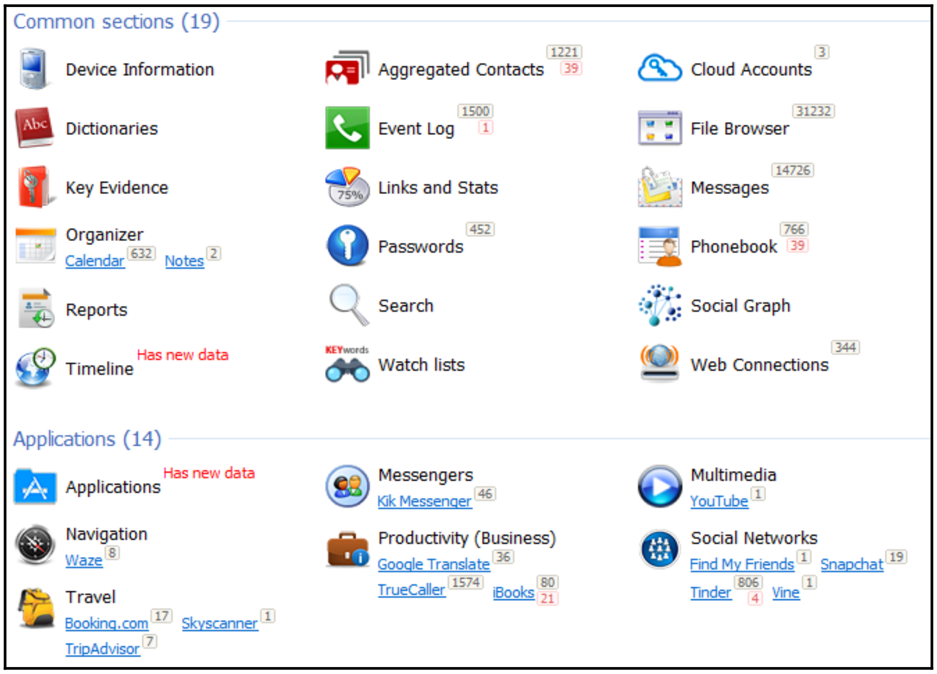


Apple iPhone 6s

Add photo

Alias	Apple iPhone 6s
Retail name	Apple iPhone 6s
Internal name	iPhone8,1
Platform	iOS
IMEI	355427078639925
Software revision	9.3
Boot loader	iBoot-2817.40.106
Acquisition type	Classic logical
Extracted by version	8.2.0.273
Extraction started	27/03/2016 22:26:19

Moreover, two separate areas are also present, the first one refers to **Common sections**, that is the information related to native applications and to the grouping functionalities offered by the software; the second one refers to the activities of the main applications installed on the device by the user.



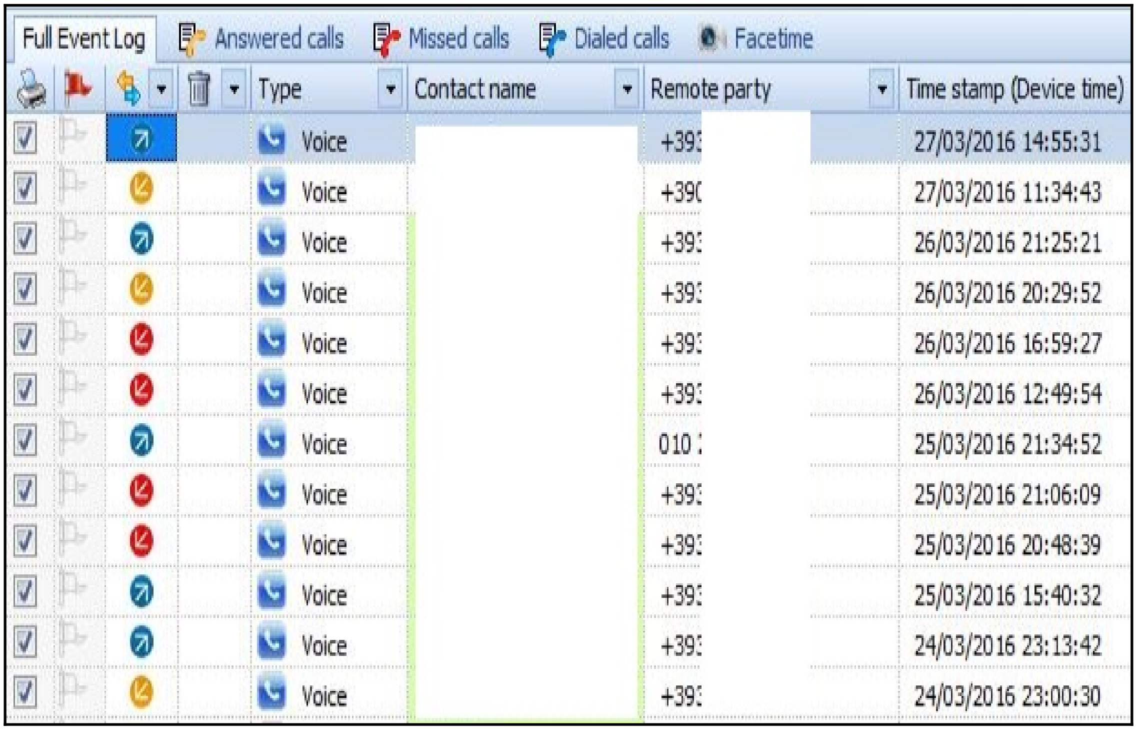
Common sections (19)

- Device Information
- Aggregated Contacts 1221 39
- Cloud Accounts 3
- Dictionaries
- Event Log 1500 1
- File Browser 31232
- Key Evidence
- Links and Stats
- Messages 14726
- Organizer: Calendar 632, Notes 2
- Passwords 452
- Phonebook 766 39
- Reports
- Search
- Social Graph
- Timeline Has new data
- Watch lists
- Web Connections 344

Applications (14)

- Applications Has new data
- Messengers: Kik Messenger 46
- Multimedia: YouTube 1
- Navigation: Waze 8
- Productivity (Business): Google Translate 36, TrueCaller 1574, iBooks 80 21
- Social Networks: Find My Friends 1, Snapchat 19, Tinder 806 4, Vine 1
- Travel: Booking.com 17, Skyscanner 1, TripAdvisor 7

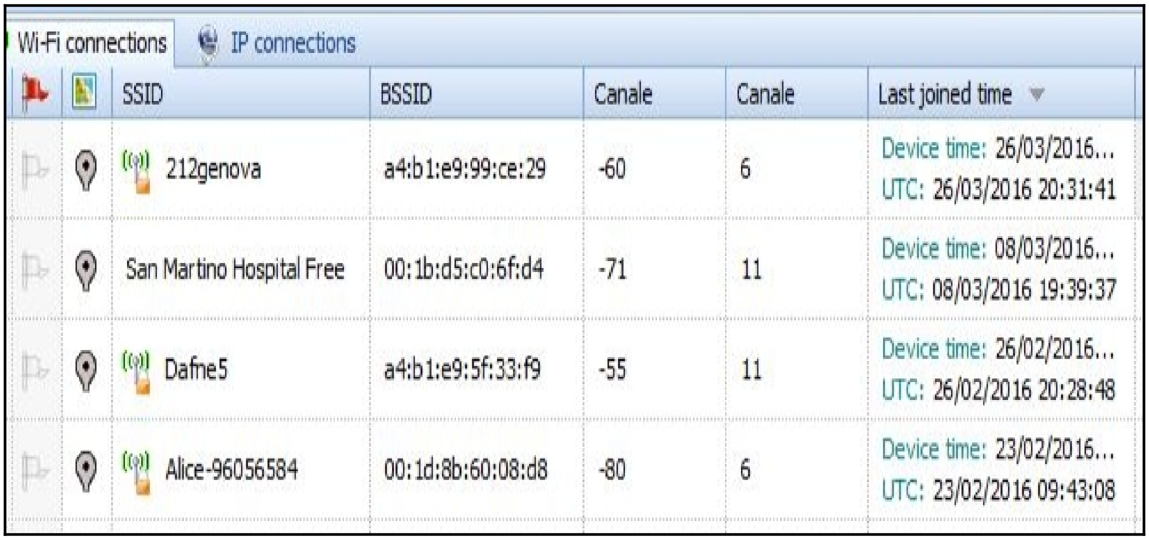
The analysis of native applications lets the analyst recover much information, such as the phonebook with assigned photos, calendar events and notes, call log (FaceTime, dialed, received, and missed calls), messages (SMS/MMS and iMessages), and voicemail. The following screenshot shows an example of a call history:



Full Event Log | Answered calls | Missed calls | Dialed calls | Facetime

				Type	Contact name	Remote party	Time stamp (Device time)
☑		↗		Voice	[redacted]	+393 [redacted]	27/03/2016 14:55:31
☑		↙		Voice	[redacted]	+390 [redacted]	27/03/2016 11:34:43
☑		↗		Voice	[redacted]	+393 [redacted]	26/03/2016 21:25:21
☑		↙		Voice	[redacted]	+393 [redacted]	26/03/2016 20:29:52
☑		↙		Voice	[redacted]	+393 [redacted]	26/03/2016 16:59:27
☑		↙		Voice	[redacted]	+393 [redacted]	26/03/2016 12:49:54
☑		↗		Voice	[redacted]	010 [redacted]	25/03/2016 21:34:52
☑		↙		Voice	[redacted]	+393 [redacted]	25/03/2016 21:06:09
☑		↙		Voice	[redacted]	+393 [redacted]	25/03/2016 20:48:39
☑		↗		Voice	[redacted]	+393 [redacted]	25/03/2016 15:40:32
☑		↗		Voice	[redacted]	+393 [redacted]	24/03/2016 23:13:42
☑		↙		Voice	[redacted]	+393 [redacted]	24/03/2016 23:00:30

Moreover, with the Oxygen Forensics Suite, it is possible to recover information related to Wi-Fi access points, IP connections, and locations. The following screenshot shows the details of Wi-Fi networks stored in the device under analysis. For each network, the SSID, MAC address of the router/access point, and the connection timestamps (last joined time and last auto joined time) are listed. From websites such as `https://www.wigle.net/`, it is possible to trace the MAC addresses and find the physical position of where the device was.

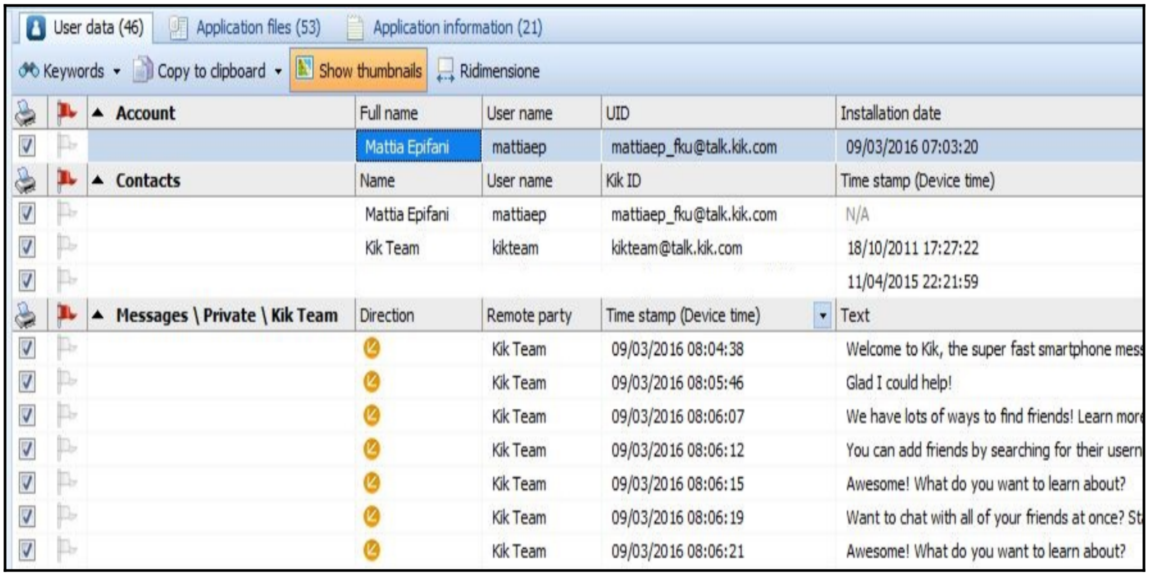


SSID	BSSID	Canale	Canale	Last joined time
212genova	a4:b1:e9:99:ce:29	-60	6	Device time: 26/03/2016... UTC: 26/03/2016 20:31:41
San Martino Hospital Free	00:1b:d5:c0:6f:d4	-71	11	Device time: 08/03/2016... UTC: 08/03/2016 19:39:37
Dafne5	a4:b1:e9:5f:33:f9	-55	11	Device time: 26/02/2016... UTC: 26/02/2016 20:28:48
Alice-96056584	00:1d:8b:60:08:d8	-80	6	Device time: 23/02/2016... UTC: 23/02/2016 09:43:08

Regarding the analysis of the applications installed by the user, the software extracts and interprets both databases and configuration files (usually in the `plist` format) for the most common applications present on the Apple Store. These applications are split in the following categories:

- **Messengers**: Facebook, Skype, WhatsApp, Viber, Telegram, Facebook Messenger, ChatON, Fring, Kakao Talk, Tango, WeChat, Yahoo, Google Hangouts, KiK Messenger, QQ, testPlus, Line, and so on
- **Navigations**: Google Maps, Apple Maps, Waze, and so on
- **Browser**: Safari, Google Chrome, Mercury, Dolphin, Yandex, and so on
- **Social networks**: Facebook, LinkedIn, Twitter, Instagram, Google+, Tinder, Grindr, Foursqare, Snapchat, Vkontakte, Pokemon Go, and so on
- **Travel**: Booking, SkyScanner, TripAdvisor, and so on
- **Productivity business**: Google Drive, Dropbox, Google Mail, OneDrive, Yahoo Mail, iBooks, and so on

The following screenshot shows an example of Kik Messenger analysis:



User data (46) | Application files (53) | Application information (21)

Keywords | Copy to clipboard | Show thumbnails | Ridimensione

		Account	Full name	User name	UID	Installation date
✓			Mattia Epifani	mattiaep	mattiaep_fku@talk.kik.com	09/03/2016 07:03:20
		Contacts	Name	User name	Kik ID	Time stamp (Device time)
✓			Mattia Epifani	mattiaep	mattiaep_fku@talk.kik.com	N/A
✓			Kik Team	kikteam	kikteam@talk.kik.com	18/10/2011 17:27:22
✓						11/04/2015 22:21:59
		Messages \ Private \ Kik Team	Direction	Remote party	Time stamp (Device time)	Text
✓				Kik Team	09/03/2016 08:04:38	Welcome to Kik, the super fast smartphone mes
✓				Kik Team	09/03/2016 08:05:46	Glad I could help!
✓				Kik Team	09/03/2016 08:06:07	We have lots of ways to find friends! Learn mor
✓				Kik Team	09/03/2016 08:06:12	You can add friends by searching for their usern
✓				Kik Team	09/03/2016 08:06:15	Awesome! What do you want to learn about?
✓				Kik Team	09/03/2016 08:06:19	Want to chat with all of your friends at once? St
✓				Kik Team	09/03/2016 08:06:21	Awesome! What do you want to learn about?

Finally, the software offers advanced functionalities for cross-searching data as follows:

- **Aggregated Contacts**: This section analyzes the contacts from multiple sources such as the Phonebook, Messages, Event Log, Skype, chat, and messaging applications in **Aggregated Contacts**. This section automatically reveals the same people in different sources and groups them in one metacontact.
- **Dictionaries**: This section shows all the words ever entered in device messages, notes, and calendars.
- **Links and Stats**: This section reveals social connections between users of mobile devices under investigation and their contacts. The **Links and Stats** section provides a tool to explore social connections between device users by analyzing calls, text, multimedia and e-mail messages, and Skype activities.

- **Timeline**: This section organizes all calls, messages, calendar events, geo data, and other activities in a chronological way, so that the analyst can follow the conversation history without the need to switch between different sections.
- **Social Graph**: This section is a workplace that allows the analyst to review connections between mobile device owners and their contacts, pinpoint connections between multiple device owners, and detect their common contacts.

Other than the automated analysis, the Oxygen Forensics Suite also offers the ability to navigate inside the filesystem and view all the different file types (documents, images, videos, and audio). There are also two embedded tools to view SQLite databases and `plist` files. The first one also offers the possibility to recover deleted records from databases, therefore, giving the possibility to retrieve calls, messages, photo thumbnails, contact photos, applications databases, and so on.

The use of this software also made it very easy for users not having high technical skills. It allows performing searches of keywords in a very intuitive way, also applying filters on every field of the application analyzed. Finally, it also allows exporting findings and it automatically generates a report in different formats (Word, Excel, PDF, HTML, and so on).

A detailed list of the feature available for the iOS devices can be found at `http://www.oxygen-forensic.com/en/compare/devices/software-for-iphone`.

Summary

In this chapter, we showed how to approach the analysis of both native iOS applications that come with every iOS device, and third-party applications. We saw some of the most common applications, but the approach is the same for any other. The importance of being able to parse the `plist` files and SQLite databases, and to carve out deleted records from the latter also became clear, since these are the two main data structures an analyst will have to deal with in every analysis. Last but not least, this chapter provided you with a good amount of locations of interesting forensics artifacts, as well as tools to analyze them. Remember that in-depth analysis, references, and tools are available at `Appendix A`, *References*, and `Appendix B`, *Tools for iOS Forensics*.

The next chapter is dedicated to the analysis of mobile applications, whether they are malicious/suspicious or not. We will go through the setup of an analysis environment and then learn the application analysis principle, which will allow you to analyze the complete behavior of a mobile application.

Self-test questions

Q1. In which iOS folder is most of the user's information of interest saved?

1. `/private/var/mobile/`
2. `/Users/mobile/`
3. `/private/var/user/mobile/`
4. `/private/user/mobile/`

Q2. Which is the timestamp convention used in iOS?

1. UNIX Epoch Time
2. Apple Time
3. Windows Time
4. MAC Absolute Time

Q3. What does the file `/private/var/root/Library/Lockdown/data_ark.plist` contain?

1. Last store search
2. Information about the device and about its account holder
3. List of installed applications
4. Password saved in the iDevice

Q4. In which file is the information related to the SIM card used in the iDevice stored?

1. `ClearedSections.plist`
2. `com.apple.network.identification.plist`
3. `com.apple.commcenter.plist`
4. `com.apple.springboard.plist`

Q5. What is the name of the database containing the user address book?

1. `AddressBook.db`
2. `AddressBook.sqlited`b
3. `AddressBook.sqlite`
4. `AB.db`

Q6. In which folder is the call history saved?

1. `/private/var/CallHistory`
2. `/private/var/wireless/Library/CallHistory/`
3. `/private/var/wireless/Library/CallHistoryDB/`
4. `/private/var/Library/CallHistory/`

Q7. What kind of file is used to store Safari browsing history?

1. SQLite
2. Txt
3. Plist
4. HTML

Q8. How is the file containing the keyboard cache used for auto correction and auto completion called?

1. `UserDictionary.txt`
2. `Dict.dat`
3. `Dynamic-Text.dat`
4. `Text.dat`

7
Applications and Malware Analysis

Although malware for iOS devices is not so common, it is more common when considering jailbroken devices. As a forensic analyst, you may be required to analyze a malicious application, or more generally the behavior of a suspicious application you have never seen before. While we are not trying to write a comprehensive guide to static reverse engineer iOS applications, this chapter gives an overview of how to analyze an application, whether it is malicious or not. In this chapter, you will first learn how to set up the working environment, and install and configure the basic tools needed for iOS application analysis. Then, we will move on to application analysis principles, learning at which state data can exist and where to look for it. Finally, we will see some tools in action that can help to speed up analysis and automate some tasks.

Setting up the environment

The first step to take in order to properly set up a testing environment for iOS application analysis is to jailbreak your testing device. This is because, as an analyst, you need to have full control of what is happening in the device, being able to access all kinds of information, whether they are stored in the memory or being sent over the network.

How to jailbreak an iPhone is out of the scope of this book, so we will not go into the details on how to do it. It is also quite simple. Just download one of the software options available, such as Evasi0n (for iOS 7), Redsn0w (for iOS 8), PanGu (for iOS 8 or 9), and TaiG9 (for iOS 9) and follow the instructions.

Once the device has been jailbroken and Cydia installed, you also need to install the following tools:

- **OpenSSH**: This tool allows you to log in to your jailbroken device via Wi-Fi or USB and have a root shell access into it
- **MTerminal**: This tool allows you to run terminal commands on your device directly from your device, rather than logging in via SSH from a different system
- **BigBoss recommended tools**: This package contains a series of useful command-line tools such as `apt`, `make`, `wget`, `sqlite3`, and so on

Something you will always need to do when analyzing a malicious application is to interact with your iPhone via shell, whether to install new tools or launch specific commands from the shell; this is why we installed OpenSSH. The first thing you need to do is to change your default root password, which is `alpine`, in order to prevent someone else logging remotely into your device (and with root privileges). To do this, launch the MTerminal application you just installed and run the following commands:

```
# su root
Password:
# passwd
Changing password for root.
New password:
Retype new password:
#
```

Now, there is a nice and comfortable way to connect to your iPhone via USB instead of being obliged to go over Wi-Fi. In your computer, edit the `~/.ssh/config` file by adding the following entry:

```
Host usb
HostName 127.0.0.1
Port 2222
User root
RemoteForward 8080 127.0.0.1:8080
```

This will map the `usb` hostname to the `ssh` connection with the proper parameters needed. Moreover, the last row sets up port forwarding such that any connections to port `8080` on the iPhone will be forwarded to port `8080` locally on the laptop. This will be useful when you have to set up a proxy to intercept the network communications, as you will see later in this chapter. Now, you need something listening on port `2222`: `usbmuxd`. This daemon is in charge of multiplexing connections over USB to the iDevice. To complete the procedure on OS X, you can simply use the following command:

```
$ brew install usbmuxd
$ iproxy 2222 22
$ ssh usb
```

Done! Now you have a shell in your iPhone via USB.

Before installing the other tools, it is a good practice to make sure that the baseline is up to date. To do this, just execute the following commands from your root shell:

```
# apt-get update
# apt-get upgrade
```

The `update` command gets the latest packages list from the default repository, while the `upgrade` command will fetch the new versions of packages that already exist on the device and don't have the latest version installed using the information received by the `update` command run before.

The following sections will quickly introduce three important utilities, which you will find useful for dumping encrypted content from memory (for example, code) and the Keychain password container.

class-dump

class-dump is a command-line tool used to extract the **Objective-C** class information from (decrypted) iOS applications, and it comes installed with the Cydia package. Note that it will work only with Objective-C apps and not with Swift.

Finally, be aware that the old `class-dump-z` is not compatible with 64-bit architectures, which means from iPhone 5s on.

Keychain Dumper

A very interesting and useful tool is Keychain Dumper that, as the name suggests, will let you dump the contents from the Keychain. Normally, the way an application is granted access to the Keychain is specified in its entitlements, which define the information that can be accessed by that application. The way this tool works is that the binary is signed with a self-signed certificate with wildcard entitlements. Hence, it is able to access all the Keychain items. To install `keychain_dumper`, just download the `zip` archive from the GitHub repo at `https://github.com/ptoomey3/Keychain-Dumper` and unpack it. Then, you only need to copy the `keychain_dumper` binary to the phone as follows:

```
$ scp keychain_dumper root@usb:/tmp/
```

Then, make sure that `keychain_dumper` is executable and validate that `/private/var/Keychains/keychain-2.db` is world readable. If not, you can set them as follows:

```
# chmod u+x keychain_dumper
# chmod +r /private/var/Keychains/keychain-2.db
```

You should now be able to run the tool without any issues:

```
# ./keychain_dumper
Generic Password
----------------
Service: AirPort
Account: ******** Work
Entitlement Group: apple
Label: (null)
Generic Field: (null)
Keychain Data: s***iami**********
...
...
```

As you can see from the preceding output, by default, `keychain_dumper` only dumps generic and Internet passwords. However, you can also specify optional flags to dump additional information from the Keychain, as shown from the help (`-h`) command as follows:

```
# ./keychain_dumper -h
Usage: keychain_dumper [-e]|[-h]|[-agnick]
<no flags>: Dump Password Keychain Items (Generic Password, Internet
Passwords)
-a: Dump All Keychain Items (Generic Passwords, Internet Passwords,
Identities, Certificates, and Keys)
-e: Dump Entitlements
-g: Dump Generic Passwords
-n: Dump Internet Passwords
-i: Dump Identities
-c: Dump Certificates
-k: Dump Keys
```

dumpDecrypted

An executable of an application is encrypted when downloaded from the App Store. The `dumpDecrypted` tool, developed by Stefan Esser (his Twitter handle is `@i0n1c`), runs the targeted app and dumps it decrypted from memory to disk. To install `dumpDecrypted`, download the `zip` archive from its GitHub page (`https://github.com/stefanesser/dumpdecrypted`) in your Mac (it is for OS X only), unzip it, and compile it on your Mac first. Alternatively, `dumpDecrypted` comes also as a tool that can be installed via idb as we will see later.

Application analysis

When analyzing an application, you need to look at all its activities and interactions with the system by analyzing all the traces and artifacts left on the system while it was running and after it has run and to/from the system. This means being able to understand how and with whom the application communicates by sending and receiving data. Therefore, you need to look at the three states where data can exist. The following are the three states where data can exist:

Data at rest

With data at rest, we refer to all the data recorded on storage media; in our case, on the mobile device's internal memory. These are the `plist` files, the `sqlite` databases, logs, and any other information we can retrieve directly from the media itself. We will not go into much details here, since this procedure is the same as for the forensic analysis of a specific application that is going through the application directory tree structure to check its files and analyze the system logs. Refer to `Chapter 6`, *Analyzing iOS Devices*, for more details.

Data in use

Data in use is, as the name suggests, all data being currently used by the application. Such data resides in the memory (RAM) of the device. In a standard malware analysis for computer malware, memory analysis is, whenever possible, part of the game. Unfortunately, for iOS, but in general, for the entire mobile panorama, memory acquisition and analysis is not well developed yet, although some utilities/proof-of-concepts to dump the memory have been implemented. However, memory analysis and runtime manipulation/abuse are out of the scope of this book, but you can explore them yourself and refer to **memscan** (`https://hexplo.it/introducing-memscan/`) or **heapdump-ios** (`https://blog.netspi.com/ios-tutorial-dumping-the-application-heap-from-memory/`) to learn about memory analysis, and *Hacking and Securing iOS Applications*, *Jonathan Zdziarski*, *O'Reilly Media*, to learn about runtime manipulation/abuse.

Another interesting tool you may want to keep in your arsenal is **Frida** (`http://www.frida.re/`), a dynamic instrumentation framework that allows you to inject your own scripts into processes to execute custom debugging logic. Frida has a client/server model, which means you will need to install it both on your computer and on the iDevice.

On Mac, it is as easy as to type the following:

```
sudo pip install frida
```

While on your jailbroken iDevice, start Cydia and add Frida's repository by navigating to **Manage** | **Sources** | **Edit** |**Add** and entering `https://build.frida.re`. You should now be able to find and install the Frida package. You can find more details about the documentation on Frida's official website. However, talking about memory, a very useful utility built on top of Frida is Fridump (`https://github.com/Nightbringer21/fridump`), an open source memory dumping tool. If Frida is correctly installed on both your computer and your iDevice (bear in mind that it would be better to have the same version installed on both), the approach to follow would be first to get the process name via Frida using the following command:

```
$ frida-ps -U
 PID  Name
----  ----------------
1744  Cydia
 137  Mail
1738  Settings
1808  Skype
  78  BTServer
1792  CacheDeleteAppCo
...
```

The `-U` parameter indicates that you are targeting the USB connected device.

Once you get the process name, you can pass it to `fridump` as shown in the following example:

```
$ python fridump.py -u -s --max-size 1048576 Skype
        ______    _     _
        |  ___|  (_)   | |
        | |_ _ __ _  __| |_   _ _ __ ___  _ __
        |  _| '__| |/ _` | | | | '_ ` _ \| '_ \
        | | | |  | | (_| | |_| | | | | | | |_) |
        \_| |_|  |_|\__,_|\__,_|_| |_| |_| .__/
                                         | |
                                         |_|
Current Directory: /Projects/iosmem/fridump
Output directory is set to: /Projects/iosmem/dump
Creating directory...
Starting Memory dump...
Progress: [########################################----]88.42%
```

The preceding example will dump the Skype process from the device connected via USB (`-u`), reading in chunks of 1 MB (`--max-size`) and extracting also the strings on all dump files (`-s`).

Fridump has been tested successfully on iOS 8, while on iOS 9 from time to time the dump gets stuck close to the end. Even in this case, most of the content is being dumped and you will be able to retrieve what you would expect from any memory dump in the classical computer environment: configuration files, passwords, code executed, everything in clear. Some more information, as well as some case examples, is retrievable from the author's website (`http://pentestcorner.com/introduction-to-fridump/`).

Data in transit

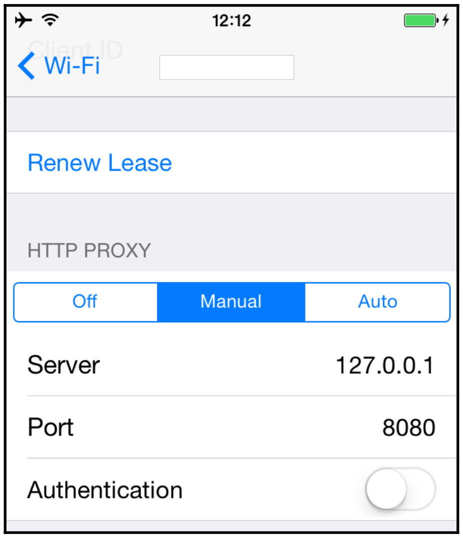
Data in transit refers to any information that is being transferred between two nodes in a network, which is, in our case, all data sent and received by the target application. Being able to observe and manipulate data sent over the network by an application is extremely interesting and useful for behavioral/dynamic analysis in the case of a suspicious app.

Before starting, remember to isolate the device from the networks (all of them), especially if you are analyzing a malicious application. Therefore, create an ad hoc wireless network that is isolated (not connected to the Internet or to your internal network), then set your iPhone to Airplane Mode and switch on only the Wi-Fi afterwards so that the other network interfaces remain off.

To begin with, you need to route the traffic of the phone through your computer in order to pose yourself as the man in the middle. To use the trick in your `ssh` configuration, as we did before, start by launching `iproxy` and establishing an `ssh` connection to your phone as follows:

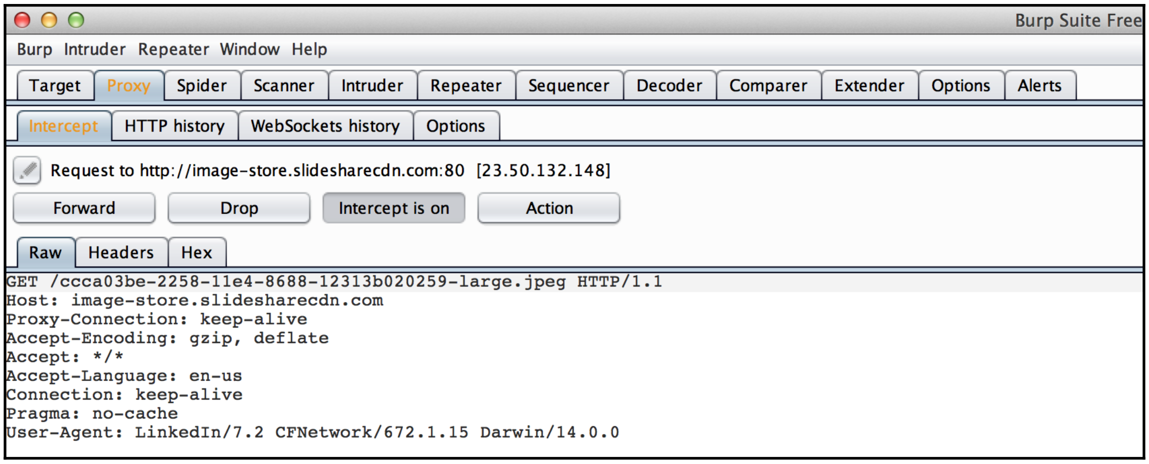
```
$ iproxy 2222 22
$ ssh usb
```

Then, from your device network configuration, set up an HTTP proxy to manual toward localhost `127.0.0.1` port `8080`. It will be redirected to your Mac to port `8080` :



Server 127.0.0.1
Port 8080

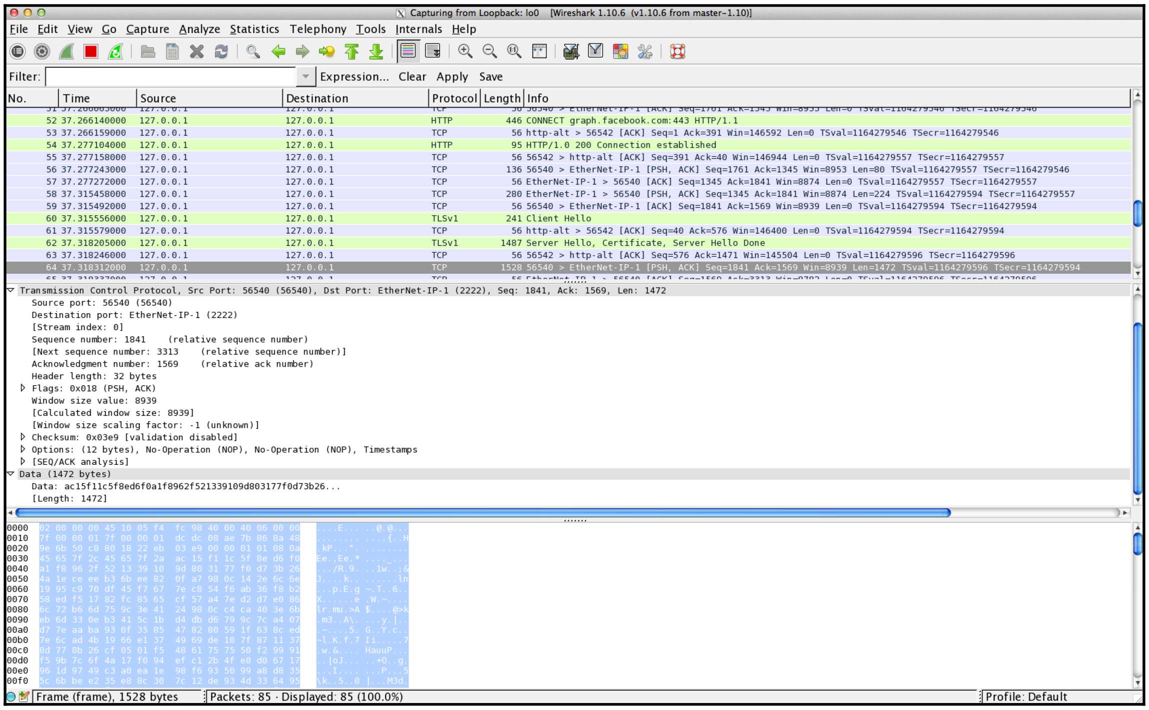
Now that the iPhone is set up, you need to set up a proxy listening on your local host port `8080`. **Burp Proxy** is probably the most popular proxy (`http://portswigger.net/burp/`); it is cross-platform and there is a free version that works just fine for our purposes. However, there are many others out there, so pick your favorite one. Once an HTTP request has been intercepted, with Burp you can perform several actions, such as modifying the request parameters, intercepting and modifying the response, and much more:


Burp Suite Free
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is on Action
Raw Headers Hex

Request to http://image-store.slidesharecdn.com:80 [23.50.132.148]

```
GET /ccca03be-2258-11e4-8688-12313b020259-large.jpeg HTTP/1.1
Host: image-store.slidesharecdn.com
Proxy-Connection: keep-alive
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-us
Connection: keep-alive
Pragma: no-cache
User-Agent: LinkedIn/7.2 CFNetwork/672.1.15 Darwin/14.0.0
```

However, although Burp is great at intercepting the HTTP/HTTPS protocol, you may want to have a look at all the traffic, because some applications may not use standard HTTP to communicate, and record it for further analysis on a later stage. To do so, you will need to install Wireshark, the standard de facto packet analyzer, together with `tcpdump`, and run a capture on your loopback interface `127.0.0.1`:


Capturing from Loopback: lo0 [Wireshark 1.10.6 (v1.10.6 from master-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No.  Time          Source     Destination  Protocol Length Info
  51 37.266063000  127.0.0.1  127.0.0.1    TCP        56 56540 > EtherNet-IP-1 [ACK] Seq=1761 Ack=1345 Win=8953 Len=0 TSval=1164279546 TSecr=1164279546
  52 37.266140000  127.0.0.1  127.0.0.1    HTTP      446 CONNECT graph.facebook.com:443 HTTP/1.1
  53 37.266159000  127.0.0.1  127.0.0.1    TCP        56 http-alt > 56542 [ACK] Seq=1 Ack=391 Win=146592 Len=0 TSval=1164279546 TSecr=1164279546
  54 37.277104000  127.0.0.1  127.0.0.1    HTTP       95 HTTP/1.0 200 Connection established
  55 37.277158000  127.0.0.1  127.0.0.1    TCP        56 56542 > http-alt [ACK] Seq=391 Ack=40 Win=146944 Len=0 TSval=1164279557 TSecr=1164279557
  56 37.277243000  127.0.0.1  127.0.0.1    TCP       136 56540 > EtherNet-IP-1 [PSH, ACK] Seq=1761 Ack=1345 Win=8953 Len=80 TSval=1164279557 TSecr=1164279546
  57 37.277272000  127.0.0.1  127.0.0.1    TCP        56 EtherNet-IP-1 > 56540 [ACK] Seq=1345 Ack=1841 Win=8874 Len=0 TSval=1164279557 TSecr=1164279557
  58 37.315458000  127.0.0.1  127.0.0.1    TCP       280 EtherNet-IP-1 > 56540 [PSH, ACK] Seq=1345 Ack=1841 Win=8874 Len=224 TSval=1164279594 TSecr=1164279557
  59 37.315492000  127.0.0.1  127.0.0.1    TCP        56 56540 > EtherNet-IP-1 [ACK] Seq=1841 Ack=1569 Win=8939 Len=0 TSval=1164279594 TSecr=1164279594
  60 37.315556000  127.0.0.1  127.0.0.1    TLSv1     241 Client Hello
  61 37.315579000  127.0.0.1  127.0.0.1    TCP        56 http-alt > 56542 [ACK] Seq=40 Ack=576 Win=146400 Len=0 TSval=1164279594 TSecr=1164279594
  62 37.318205000  127.0.0.1  127.0.0.1    TLSv1    1487 Server Hello, Certificate, Server Hello Done
  63 37.318246000  127.0.0.1  127.0.0.1    TCP        56 56542 > http-alt [ACK] Seq=576 Ack=1471 Win=145504 Len=0 TSval=1164279596 TSecr=1164279596
  64 37.318312000  127.0.0.1  127.0.0.1    TCP      1528 56540 > EtherNet-IP-1 [PSH, ACK] Seq=1841 Ack=1569 Win=8939 Len=1472 TSval=1164279596 TSecr=1164279594
  65 37.318337000  127.0.0.1  127.0.0.1    TCP        56 EtherNet-IP-1 > 56540 [ACK] Seq=1569 Ack=3313 Win=8782 Len=0 TSval=1164279596 TSecr=1164279596
```

```
▽ Transmission Control Protocol, Src Port: 56540 (56540), Dst Port: EtherNet-IP-1 (2222), Seq: 1841, Ack: 1569, Len: 1472
    Source port: 56540 (56540)
    Destination port: EtherNet-IP-1 (2222)
    [Stream index: 0]
    Sequence number: 1841    (relative sequence number)
    [Next sequence number: 3313    (relative sequence number)]
    Acknowledgment number: 1569    (relative ack number)
    Header length: 32 bytes
  ▷ Flags: 0x018 (PSH, ACK)
    Window size value: 8939
    [Calculated window size: 8939]
    [Window size scaling factor: -1 (unknown)]
  ▷ Checksum: 0x03e9 [validation disabled]
  ▷ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▷ [SEQ/ACK analysis]
▽ Data (1472 bytes)
    Data: ac15f11c5f8ed6f0a1f8962f521339109d803177f0d73b26...
    [Length: 1472]
```

```
0000  02 00 00 00 45 10 05 f4  fc 98 40 00 40 06 00 00   ....E... ..@.@...
0010  7f 00 00 01 7f 00 00 01  dc dc 08 ae 7b 86 8a 48   ........ ....{..H
0020  9e 6b 50 c8 80 18 22 eb  03 e9 00 00 01 01 08 0a   .kP...". ........
0030  45 65 7f 2c 45 65 7f 2a  ac 15 f1 1c 5f 8e d6 f0   Ee.,Ee.* ...._...
0040  a1 f8 96 2f 52 13 39 10  9d 80 31 77 f0 d7 3b 26   .../R.9. ..1w..;&
0050  4a 1e ce ee b3 6b ee 82  0f a7 98 0c 14 2e 6c 6e   J....k.. ......ln
0060  19 95 c9 70 df 45 f7 67  7e c8 54 f6 ab 36 f8 b2   ...p.E.g ~.T..6..
0070  58 ed f5 17 82 fc 85 65  cf 57 a4 7e d2 d7 e0 86   X......e .W.~....
0080  6c 72 b6 6d 75 9c 3e 41  24 98 0c c4 ca 40 3e 6b   lr.mu.>A $....@>k
0090  eb 6d 33 0e b3 41 5c 1b  d4 db d6 79 9c 7c a4 07   .m3..A\. ...y.|..
00a0  d7 7e aa ba 93 0f 35 85  47 82 80 59 1f 63 8c ed   .~....5. G..Y.c..
00b0  7e 6c ad 4b 19 66 e1 37  49 69 de 18 7f 87 11 37   ~l.K.f.7 Ii.....7
00c0  8d 77 8b 26 cf 05 01 f5  48 61 75 75 50 f2 99 91   .w.&.... HauuP...
00d0  f5 9b 7c 6f 4a 17 f0 94  ef c1 2b 4f e8 d0 67 17   ..|oJ... ..+O..g.
00e0  96 1d 97 49 c3 a0 ea 1e  98 f6 93 50 99 a8 d8 35   ...I.... ...P...5
00f0  5c 6b be e2 35 e8 8c 30  7c 12 de 93 4d 33 64 95   \k..5..0 |...M3d.
```

Frame (frame), 1528 bytes
Packets: 85 · Displayed: 85 (100.0%)
Profile: Default

Of course, on a jailbroken iPhone, you have full control and may choose to install and go via `tcpdump` directly on the device.

Automating the analysis

This section will quickly introduce some tools that will help you during the analysis, either by speeding up the most common tasks or providing you with some extra and very useful functionalities.

idb

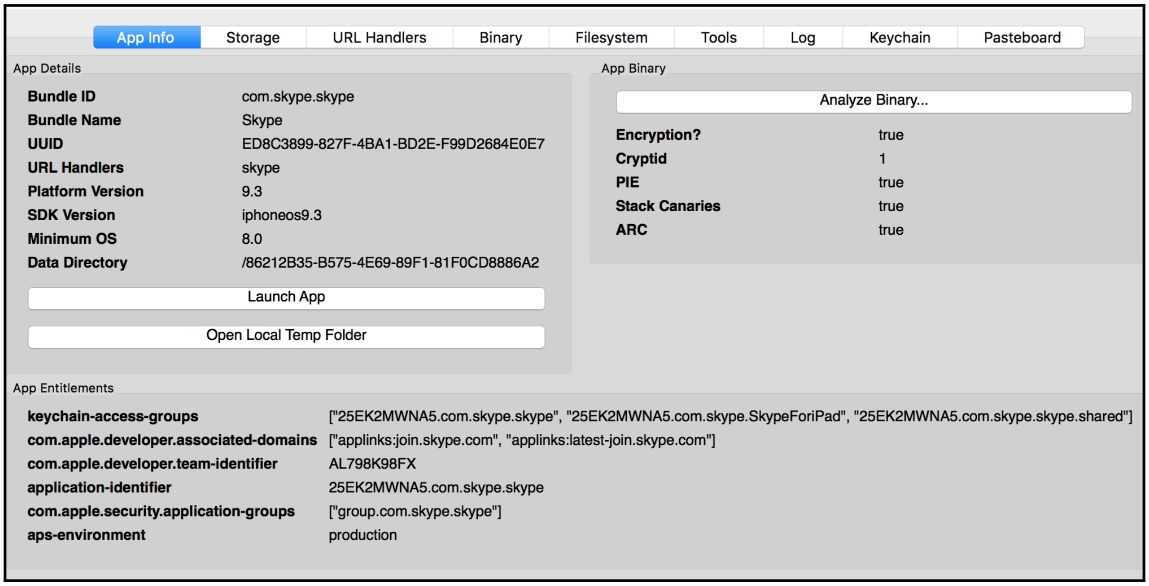
Developed and maintained by Daniel Mayer (his Twitter handle is `@DanlAMayer`), idb is a tool that simplifies some of the most common tasks related to iOS application analysis. Originally built with a penetration tester/researcher focus, it can be of great value for any type of application analysis, thanks to the number of tools that it incorporates and the features offered. Written in Ruby, the installation procedure is quite straightforward; you just need to run the following commands:

```
$ rvm install 2.1 --enable-shared
$ gem install bundler
$ brew install qt cmake usbmuxd libimobiledevice
$ gem install idb
$ idb
```

This is the procedure for Mac OS X. For more information on building and running it on other systems, you can refer to the official page at `http://www.idbtool.com/`.

Once you have launched idb after following the configuration steps to install some needed extra tools on the device, you will have to connect idb to the USB device, select an application and start the analysis. Note that, even if it will be running a nice GUI, you will be prompted for the root password of the iDevice on the shell where you launched idb.

Clicking on **Analyze Binary...**, as you can see in the following screenshot, will give the first information on the binary itself, verifying, among other things, that **PIE**, Stack Smashing Protection, and **ARC** are enabled, which would reduce the likelihood of finding memory corruption vulnerabilities to exploit. Moreover, if the binary application is encrypted, `idb` will run `dumpdecrypted` to decrypt it before analyzing it:



App Details

Bundle ID	com.skype.skype
Bundle Name	Skype
UUID	ED8C3899-827F-4BA1-BD2E-F99D2684E0E7
URL Handlers	skype
Platform Version	9.3
SDK Version	iphoneos9.3
Minimum OS	8.0
Data Directory	/86212B35-B575-4E69-89F1-81F0CD8886A2

App Binary

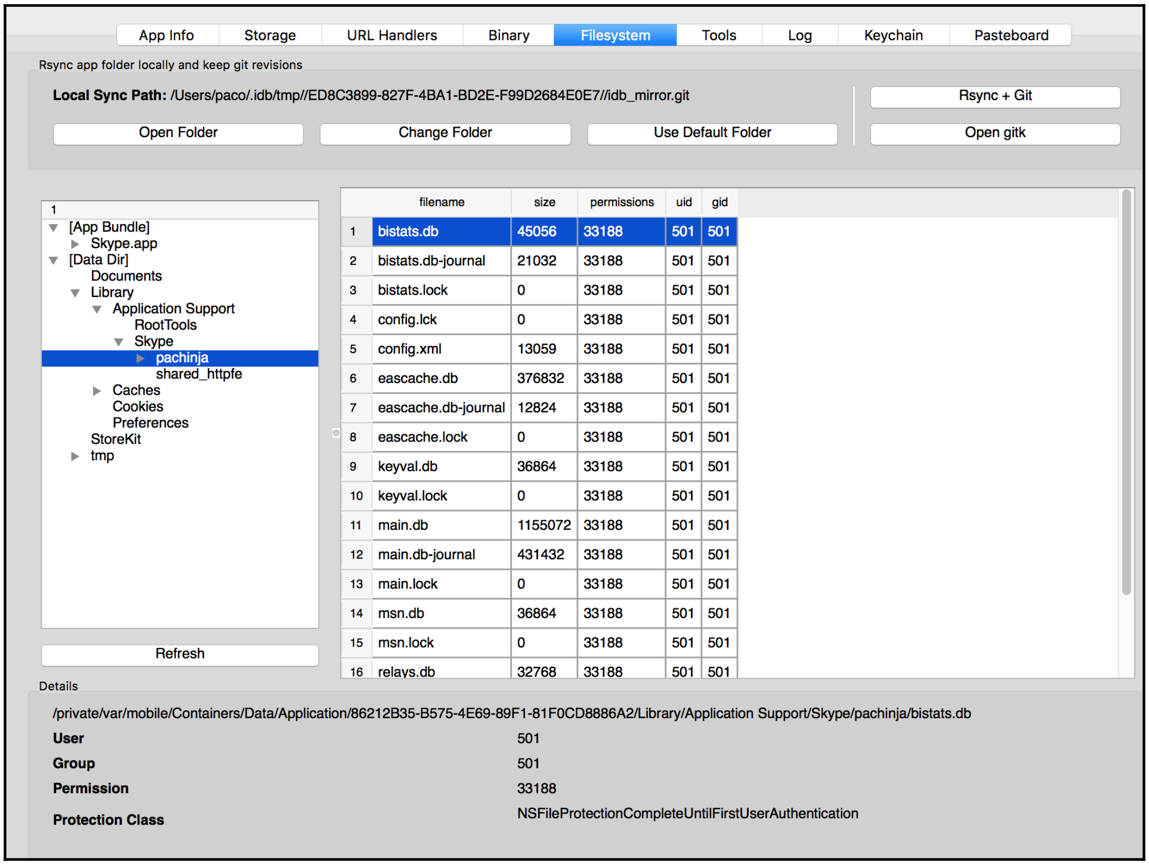
Encryption?	true
Cryptid	1
PIE	true
Stack Canaries	true
ARC	true

App Entitlements

keychain-access-groups	["25EK2MWNA5.com.skype.skype", "25EK2MWNA5.com.skype.SkypeForiPad", "25EK2MWNA5.com.skype.skype.shared"]
com.apple.developer.associated-domains	["applinks:join.skype.com", "applinks:latest-join.skype.com"]
com.apple.developer.team-identifier	AL798K98FX
application-identifier	25EK2MWNA5.com.skype.skype
com.apple.security.application-groups	["group.com.skype.skype"]
aps-environment	production

Other information related to the binary app can also be extracted from the **Binary** tab. As you will see, idb extracts all the strings from the decrypted binary. This is a standard step you would do when analyzing standard computer malware. This is of great use since here you may find the API keys, credentials, encryption keys, URLs, and other useful hints. From a static analysis perspective, idb binary analysis allows you to dump all the class information.

Talking about data at rest, under the **Storage** tab, you will be able to analyze all the files related to your target application, such as `plist`, the `sqlite` databases, and `Cache.db`, which contains cached HTTP requests/responses and offline data cached by web applications such as images, HTML, JavaScript, style sheets, and more. The idb tool will also allow you to navigate through the app tree structure from the **Filesystem** tab, taking and storing subsequent different snapshots to navigate and compare at a later stage:



Local Sync Path: /Users/paco/.idb/tmp//ED8C3899-827F-4BA1-BD2E-F99D2684E0E7//idb_mirror.git

```
1
▼ [App Bundle]
    ▶ Skype.app
▼ [Data Dir]
      Documents
    ▼ Library
        ▼ Application Support
              RootTools
            ▼ Skype
                ▶ pachinja
                  shared_httpfe
        ▶ Caches
          Cookies
          Preferences
      StoreKit
    ▶ tmp
```

	filename	size	permissions	uid	gid
1	bistats.db	45056	33188	501	501
2	bistats.db-journal	21032	33188	501	501
3	bistats.lock	0	33188	501	501
4	config.lck	0	33188	501	501
5	config.xml	13059	33188	501	501
6	eascache.db	376832	33188	501	501
7	eascache.db-journal	12824	33188	501	501
8	eascache.lock	0	33188	501	501
9	keyval.db	36864	33188	501	501
10	keyval.lock	0	33188	501	501
11	main.db	1155072	33188	501	501
12	main.db-journal	431432	33188	501	501
13	main.lock	0	33188	501	501
14	msn.db	36864	33188	501	501
15	msn.lock	0	33188	501	501
16	relays.db	32768	33188	501	501

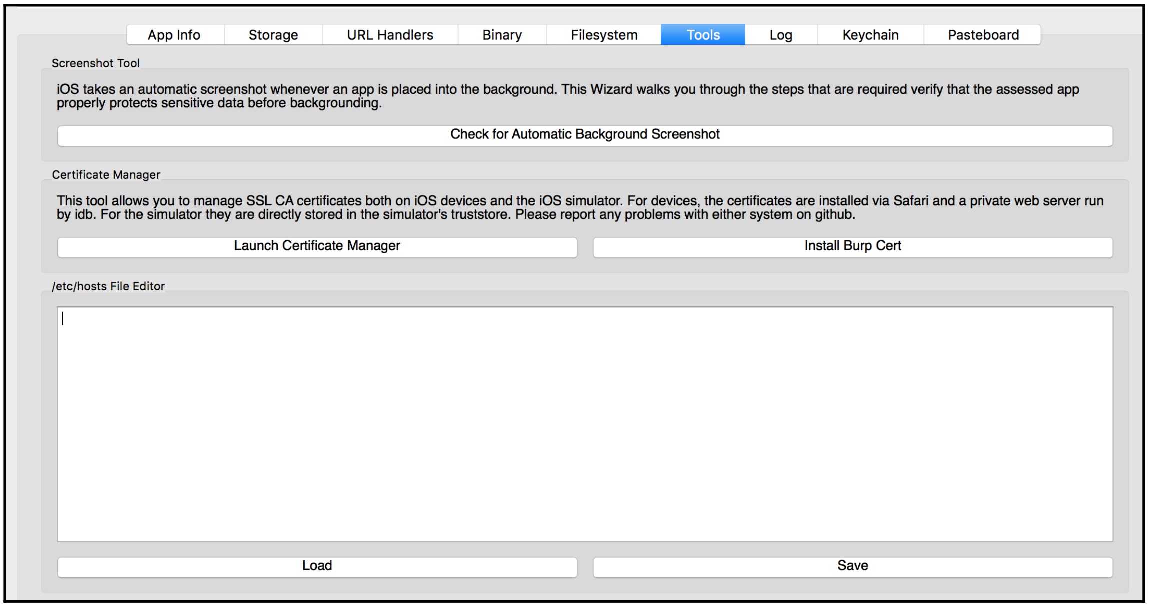
/private/var/mobile/Containers/Data/Application/86212B35-B575-4E69-89F1-81F0CD8886A2/Library/Application Support/Skype/pachinja/bistats.db

User	501
Group	501
Permission	33188
Protection Class	NSFileProtectionCompleteUntilFirstUserAuthentication

Two other interesting functionalities provided are **URL Handlers**, which shows you the list of URL handlers and includes a basic fuzzer that can be used to fuzz input data via the URL schemes, and the **Keychain** dumper, which is a functionality that allows you to dump the Keychain using `keychain_dump`.

The **Tools** tab contains several different tools that are quite handy; they are as follows:

- **Background screenshot**: Although this tool is more useful for forensics/security purposes, it looks for an eventual screenshot taken by the system when putting the application in the background by pushing the Home button.
- **Certification manager**: This tool will speed up the management and installation of the CA certificate. This is extremely useful, for example, when using Burp for HTTPS traffic and an application that actually checks that SSL is in place.
- **/etc/hosts file editor**: As we have seen before for the data in transit, apps do not always use the HTTP/S protocol, so Burp will not intercept. With this editor, you can quickly access and modify `/etc/hosts` of the iPhone in order to redirect the traffic toward custom services you may have fired up for the analysis.



Screenshot Tool

iOS takes an automatic screenshot whenever an app is placed into the background. This Wizard walks you through the steps that are required verify that the assessed app properly protects sensitive data before backgrounding.

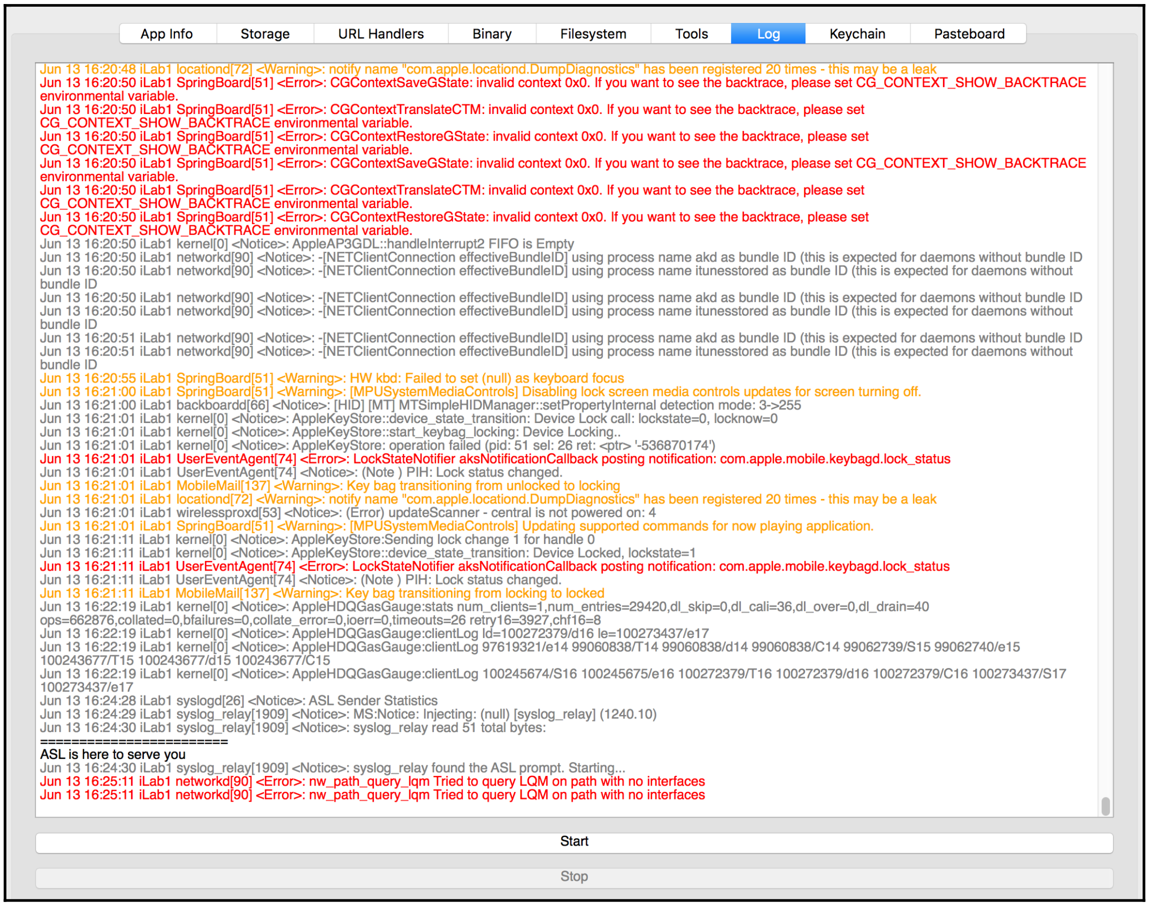
Certificate Manager

This tool allows you to manage SSL CA certificates both on iOS devices and the iOS simulator. For devices, the certificates are installed via Safari and a private web server run by idb. For the simulator they are directly stored in the simulator's truststore. Please report any problems with either system on github.

/etc/hosts File Editor

Last but not least, `idb` offers a real-time log (`syslog`) and pasteboard viewer (refer to the following screenshot) via the **Log** and **Pasteboard** tabs, respectively. Although it may not seem of great use to monitor the pasteboard when you are the one testing the application, it may surprise you to know that applications use the pasteboard also for **Inter-Process Communication** (**IPC**). By default, idb monitors only the main (default) pasteboard, but you can add additional pasteboard names to the list on the right-hand side so that you will also be able to monitor the private pasteboards.

Regarding the Log panel, idb includes both system messages and any log statements that the app produces using `NSLog`, which often discloses sensitive data:



```
Jun 13 16:20:48 iLab1 locationd[72] <Warning>: notify name "com.apple.locationd.DumpDiagnostics" has been registered 20 times - this may be a leak
Jun 13 16:20:50 iLab1 SpringBoard[51] <Error>: CGContextSaveGState: invalid context 0x0. If you want to see the backtrace, please set CG_CONTEXT_SHOW_BACKTRACE
environmental variable.
Jun 13 16:20:50 iLab1 SpringBoard[51] <Error>: CGContextTranslateCTM: invalid context 0x0. If you want to see the backtrace, please set
CG_CONTEXT_SHOW_BACKTRACE environmental variable.
Jun 13 16:20:50 iLab1 SpringBoard[51] <Error>: CGContextRestoreGState: invalid context 0x0. If you want to see the backtrace, please set
CG_CONTEXT_SHOW_BACKTRACE environmental variable.
Jun 13 16:20:50 iLab1 SpringBoard[51] <Error>: CGContextSaveGState: invalid context 0x0. If you want to see the backtrace, please set CG_CONTEXT_SHOW_BACKTRACE
environmental variable.
Jun 13 16:20:50 iLab1 SpringBoard[51] <Error>: CGContextTranslateCTM: invalid context 0x0. If you want to see the backtrace, please set
CG_CONTEXT_SHOW_BACKTRACE environmental variable.
Jun 13 16:20:50 iLab1 SpringBoard[51] <Error>: CGContextRestoreGState: invalid context 0x0. If you want to see the backtrace, please set
CG_CONTEXT_SHOW_BACKTRACE environmental variable.
Jun 13 16:20:50 iLab1 kernel[0] <Notice>: AppleAP3GDL::handleInterrupt2 FIFO is Empty
Jun 13 16:20:50 iLab1 networkd[90] <Notice>: -[NETClientConnection effectiveBundleID] using process name akd as bundle ID (this is expected for daemons without bundle ID
Jun 13 16:20:50 iLab1 networkd[90] <Notice>: -[NETClientConnection effectiveBundleID] using process name itunesstored as bundle ID (this is expected for daemons without
bundle ID
Jun 13 16:20:50 iLab1 networkd[90] <Notice>: -[NETClientConnection effectiveBundleID] using process name akd as bundle ID (this is expected for daemons without bundle ID
Jun 13 16:20:50 iLab1 networkd[90] <Notice>: -[NETClientConnection effectiveBundleID] using process name itunesstored as bundle ID (this is expected for daemons without
bundle ID
Jun 13 16:20:51 iLab1 networkd[90] <Notice>: -[NETClientConnection effectiveBundleID] using process name akd as bundle ID (this is expected for daemons without bundle ID
Jun 13 16:20:51 iLab1 networkd[90] <Notice>: -[NETClientConnection effectiveBundleID] using process name itunesstored as bundle ID (this is expected for daemons without
bundle ID
Jun 13 16:20:55 iLab1 SpringBoard[51] <Warning>: HW kbd: Failed to set (null) as keyboard focus
Jun 13 16:21:00 iLab1 SpringBoard[51] <Warning>: [MPUSystemMediaControls] Disabling lock screen media controls updates for screen turning off.
Jun 13 16:21:00 iLab1 backboardd[66] <Notice>: [HID] [MT] MTSimpleHIDManager::setPropertyInternal detection mode: 3->255
Jun 13 16:21:01 iLab1 kernel[0] <Notice>: AppleKeyStore::device_state_transition: Device Lock call: lockstate=0, locknow=0
Jun 13 16:21:01 iLab1 kernel[0] <Notice>: AppleKeyStore::start_keybag_locking: Device Locking..
Jun 13 16:21:01 iLab1 kernel[0] <Notice>: AppleKeyStore: operation failed (pid: 51 sel: 26 ret: <ptr> '-536870174')
Jun 13 16:21:01 iLab1 UserEventAgent[74] <Error>: LockStateNotifier aksNotificationCallback posting notification: com.apple.mobile.keybagd.lock_status
Jun 13 16:21:01 iLab1 UserEventAgent[74] <Notice>: (Note ) PIH: Lock status changed.
Jun 13 16:21:01 iLab1 MobileMail[137] <Warning>: Key bag transitioning from unlocked to locking
Jun 13 16:21:01 iLab1 locationd[72] <Warning>: notify name "com.apple.locationd.DumpDiagnostics" has been registered 20 times - this may be a leak
Jun 13 16:21:01 iLab1 wirelessproxd[53] <Notice>: (Error) updateScanner - central is not powered on: 4
Jun 13 16:21:01 iLab1 SpringBoard[51] <Warning>: [MPUSystemMediaControls] Updating supported commands for now playing application.
Jun 13 16:21:11 iLab1 kernel[0] <Notice>: AppleKeyStore:Sending lock change 1 for handle 0
Jun 13 16:21:11 iLab1 kernel[0] <Notice>: AppleKeyStore::device_state_transition: Device Locked, lockstate=1
Jun 13 16:21:11 iLab1 UserEventAgent[74] <Error>: LockStateNotifier aksNotificationCallback posting notification: com.apple.mobile.keybagd.lock_status
Jun 13 16:21:11 iLab1 UserEventAgent[74] <Notice>: (Note ) PIH: Lock status changed.
Jun 13 16:21:11 iLab1 MobileMail[137] <Warning>: Key bag transitioning from locking to locked
Jun 13 16:22:19 iLab1 kernel[0] <Notice>: AppleHDQGasGauge:stats num_clients=1,num_entries=29420,dl_skip=0,dl_cali=36,dl_over=0,dl_drain=40
ops=662876,collated=0,bfailures=0,collate_error=0,ioerr=0,timeouts=26 retry16=3927,chf16=8
Jun 13 16:22:19 iLab1 kernel[0] <Notice>: AppleHDQGasGauge:clientLog ld=100272379/d16 le=100273437/e17
Jun 13 16:22:19 iLab1 kernel[0] <Notice>: AppleHDQGasGauge:clientLog 97619321/e14 99060838/T14 99060838/d14 99060838/C14 99062739/S15 99062740/e15
100243677/T15 100243677/d15 100243677/C15
Jun 13 16:22:19 iLab1 kernel[0] <Notice>: AppleHDQGasGauge:clientLog 100245674/S16 100245675/e16 100272379/T16 100272379/d16 100272379/C16 100273437/S17
100273437/e17
Jun 13 16:24:28 iLab1 syslogd[26] <Notice>: ASL Sender Statistics
Jun 13 16:24:29 iLab1 syslog_relay[1909] <Notice>: MS:Notice: Injecting: (null) [syslog_relay] (1240.10)
Jun 13 16:24:30 iLab1 syslog_relay[1909] <Notice>: syslog_relay read 51 total bytes:
========================
ASL is here to serve you
Jun 13 16:24:30 iLab1 syslog_relay[1909] <Notice>: syslog_relay found the ASL prompt. Starting...
Jun 13 16:25:11 iLab1 networkd[90] <Error>: nw_path_query_lqm Tried to query LQM on path with no interfaces
Jun 13 16:25:11 iLab1 networkd[90] <Error>: nw_path_query_lqm Tried to query LQM on path with no interfaces
```

Summary

In this chapter, we introduced some tools for analyzing iOS applications, suspicious or not, mainly from a behavioral/dynamic point of view. You learned how to quickly analyze the binary, review the data and logs produced by the targeted application, intercept, manipulate, and analyze the data sent and received over the network, and automate most of the tasks with ad hoc toolkits, such as idb.

Self-test questions

Q1. Which tool can be used to extract Objective-C class information from iOS applications?

1. OpenSSH
2. MTerminal
3. class-dump
4. Keychain Dumper

Q2. Which tool can be used to dump an unencrypted application from memory?

1. usbmuxd
2. Keychain Dumper
3. dumpDecrypted
4. OpenSSH

Q3. Which tool can be used to verify the pasteboard content?

1. dumpDecrypted
2. iRet
3. iLoot
4. idb

Q4. Which tools would you use to best analyze data in transit?

1. Burp Proxy + Wireshark
2. iproxy + Wireshark
3. dumpDecrypted + tcpdump
4. tcpdump + iproxy

Q5. Which set of tools allow automating a series of tasks in order to analyze and reverse engineer iOS applications?

1. iLoot
2. idb
3. class-dump
4. dumpDecrypted

A
References

Publications freely available

Here's a list of publications that are available for free:

- *Guidelines on Mobile Device Forensics, Rick Ayers, Sam Brothers, Wayne Jansen, NIST Special Publication 800-101, 2014* is available at `http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-101r1.pdf`
- *Guidelines for Managing the Security of Mobile Devices in the Enterprise, Murugiah Souppaya, Karen Scarafone, NIST Special Publication 800-124, 2013* is available at `http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-124r1.pdf`
- *Technical Considerations for Vetting 3rd Party Mobile Applications (Draft), Jeffrey Voas, Steve Quirolgico, Christoph Michael, Karen Scarafone, NIST Special Publication 800-163 (Draft), 2014* is available at `http://csrc.nist.gov/publications/drafts/800-163/sp800_163_draft.pdf`
- *iOS Forensic, Christian Javier D'Orazio, 2013* is available at `https://wiki.cis.unisa.edu.au/wki/images/7/7c/DORAZIO_iOS_Forensics_Final_Revise.pdf`
- *iOS Forensics Investigative Methods, Jonathan Zdziarski, 2012* is available at `http://www.zdziarski.com/blog/wp-content/uploads/2013/05/iOS-Forensic-Investigative-Methods.pdf`

- *SIM and USIM Filesystem, A Forensics Perspective, Antonio Savoldi, Paolo Gubian, Proceedings of the 2007 ACM Symposium on Applied Computing, 2007* is available at `https://www.researchgate.net/publication/220998796_SIM_and_USIM_filesystem_a_forensics_perspective`
- *A Hypothesis-based Approach to Digital Forensic Investigations, Brian Carrier, CERIAS Tech Report 2006-06* is available at `https://www.cerias.purdue.edu/assets/pdf/bibtex_archive/2006-06.pdf`
- *A Road Map for Digital Forensic Research, 2001* is a report from the first **Digital Forensic Research Workshop** (**DFRWS**), which is available at `http://www.dfrws.org/sites/default/files/session-files/pres-social_networking_applications_on_mobile_devices.pdf`

Tools, manuals, and reports

Here is a list of some online tools, manuals, and reports:

- *Elcomsoft iOS Forensic Toolkit Guide, Colby Lahaie, Champlain College* is available at `http://www.champlain.edu/Documents/LCDI/Elcomsoft_iOS_Forensic_Toolkit_Guide.pdf`
- *Cellebrite iOS Device Physical Extraction Manual, Cellebrite* is available at `http://www.ume-update.com/UFED/iOS_User_Manual.pdf`
- *What Happens When You Press that Button?, Cellebrite* is available at `https://www.cellebrite.com/collateral/Explaining_Cellebrite_UFED_Data_Extraction_Processes.pdf`
- *Phone Forensics Express v2.1.2.2761 Test Results for Mobile Device Acquisition Tool* is available at `https://www.dhs.gov/sites/default/files/publications/508_Test%20Report_NIST_Mobile_Phone%20Forensics%20Express%20v2.1.2.2761_December%202015.pdf`

- *Lantern v4.5.6 Test Results for Mobile Device Acquisition Tool* is available at `https://www.dhs.gov/sites/default/files/publications/508_Test%20Report_NIST_Lantern%20v4.5.6_August%202015_Final_1.pdf`
- *Test Results for Mobile Device Acquisition Tool: Lantern v2.3, NIST* is available at `https://ncjrs.gov/pdffiles1/nij/241154.pdf`
- *iOS Forensics with Open-Source Tools, Andrey Belenko, Zeronights, 2014* is available at `http://2014.zeronights.org/assets/files/slides/belenko.pdf`
- *Elcomsoft iOS Forensic Toolkit, Physical Acquisition for 64-bit Devices and iOS 9 Support* is available at `https://www.elcomsoft.com/PR/eift_151118_en.pdf`
- *Cellebrite Unlocking Services Now Available* for Apple iOS 9 is available at `http://www.cellebrite.com/Pages/cellebrite-unlocking-services-now-available-for-apple-ios-9-and-samsung-galaxy-s6-and-s7-devices`
- *Apple iOS and Watch OS Artifacts* is available at `https://docs.google.com/spreadsheets/d/1zAqSSiLZ-Fw6RrXg7qegkxbE7ejXI_mpxQre_ybWRWw/`
- *IP Box User Tutorial* is available at `http://www.champlain.edu/Documents/LCDI/IP_BOX_User_Tutorial_Edited.docx`
- *IP Box Documentation* is available at `http://www.teeltech.com/wp-content/uploads/2014/11/IP-Box-documentation-rev2-1-16-2015.pdf`

Apple's official documentation

The official Apple documentation can be downloaded directly from Apple's website. The most interesting papers about security and forensics are the following:

- *Legal Process Guidelines, U.S. Law Enforcement* is available at `https://www.apple.com/privacy/government-information-requests`
- *iOS Security, May 2016* is available at `https://www.apple.com/business/docs/iOS_Security_Guide.pdf`
- *iPhone User Guide* is available at http://help.apple.com/iphone/ and `http://manuals.info.apple.com/MANUALS/1000/MA1565/en_US/iphone_user_guide.pdf`

- iPhone Tech Specs is available at `http://support.apple.com/specs/#iphone`
- *iPad User Guide* is available at `http://support.apple.com/manuals/#ipad` and `http://manuals.info.apple.com/MANUALS/1000/MA1595/en_US/ipad_user_guide.pdf`
- iPad Tech Specs is available at `http://support.apple.com/specs/#ipad`
- *iPod touch User Guide* is available at `http://support.apple.com/manuals/#ipodtouch` and `http://manuals.info.apple.com/MANUALS/1000/MA1596/en_US/ipod_touch_user_guide.pdf`
- iPod Touch Tech Specs is available at `http://support.apple.com/specs/#ipodtouch`
- How to find the serial number, IMEI, MEID, CDN, and ICCID number for iOS can be viewed at `http://support.apple.com/kb/ht4061`
- Back up and restore your iOS device with iCloud or iTunes can be viewed at `http://support.apple.com/kb/HT1766`
- Information about iOS backups (iTunes) is available at `http://support.apple.com/kb/ht4946`
- Protect your iOS device using the information available at `http://support.apple.com/kb/HT5874`
- Forgot passcode or device disabled (iOS) information is available at `http://support.apple.com/kb/HT1212`
- iCloud storage and backup overview is available at `http://support.apple.com/kb/PH12519`
- Information about troubleshooting and creating an iCloud backup is available at `http://support.apple.com/kb/TS3992`
- HFS Plus Volume Format is available at `https://developer.apple.com/legacy/library/technotes/tn/tn1150.html`

Device security and data protection

If the reader is interested in learning more about the security of iOS devices, the following are the most interesting researches carried out:

- *Identifying Back Doors, Attack Points, and Surveillance Mechanisms in iOS Devices, Jonathan Zdziarski, Digital Investigation, Volume 11, Issue 1, March 2014* is available at `http://www.sciencedirect.com/science/article/pii/S1742287614000036`. A related presentation is available at `https://pentest.com/ios_backdoors_attack_points_surveillance_mechanisms.pdf`.
- *iPhone security model & vulnerabilities, Cedric Halbronn, Jean Sigwald, Sogeti Lab, 2010* is available at `http://esec-lab.sogeti.com/dotclear/public/publications/10-hitbkl-iphone.pdf`.
- *iPhone data protection in depth, Jean-Baptiste Bédrune, Jean Sigwald, Sogeti Lab, 2012* is available at `http://esec-lab.sogeti.com/static/publications/11-hitbamsterdam-iphonedataprotection.pdf`.
- *Forensics iOS, Jean-Baptiste Bédrune, Jean Sigwald* is available at `https://www.sstic.org/media/SSTIC2012/SSTIC-actes/forensicsios/SSTIC2012-Slides-forensicsios-sigwald_bedrune.pdf`.
- *Overcoming data protection to re-enable iOS forensics, Andrey Belenko, Black Hat USA, 2011* is available at `https://media.blackhat.com/bh-us-11/Belenko/BH_US_11_Belenko_iOS_Forensics_Slides.pdf`.
- *Handling iOS encryption in a forensic investigation, Jochem van Kerkwijk, Universiteit van Amsterdam, 2011* is available at `http://www.delaat.net/rp/2010-2011/p26/report.pdf`.
- *iOS Keychain Weakness FAQ, Jens Heider, Rachid El Khayari, Fraunhofer Institute for Secure Information Technology (SIT), 2012* is available at `http://sit.sit.fraunhofer.de/studies/en/sc-iphone-passwords-faq.pdf`.

- *Lost iPhone? Lost Passwords!, Jens Heider, Matthias Boll, Fraunhofer Institute for Secure Information Technology (SIT), 2011* is available at `https://www.sit.fraunhofer.de/fileadmin/dokumente/studien_und_technical_reports/Whitepaper_Lost_iPhone.pdf`.
- *iOS Encryption Systems, Peter Teufl, Thomas Zefferer, Christof Stromberger, Christoph Heckhenblaikner, Institute for Applied Information Processing and Communications, 2014* is available at `http://www.a-sit.at/pdfs/Technologiebeobachtung/ios-encryption-systems.pdf`.

Device hardening

Information on how to harden an iOS device can be found in the following papers:

- *End User Devices Security Guidance, Apple iOS 9, 2015* is available at `https://www.gov.uk/government/publications/end-user-devices-security-guidance-apple-ios-9/end-user-devices-security-guidance-apple-ios-9`
- *CIS Apple iOS 9 Benchmark, Center for Internet Security, 2016* is available at `https://benchmarks.cisecurity.org/downloads/show-single/?file=appleios9.100`
- *CIS Apple iOS 7 Benchmark, Center for Internet Security, 2014* is available at `https://benchmarks.cisecurity.org/downloads/show-single/?file=appleios7.110`
- *iOS Hardening Configuration Guide, Australian Government-Department of Defence, 2012* is available at `http://www.asd.gov.au/publications/iOS5_Hardening_Guide.pdf`
- *Security Configuration Recommendations for Apple iOS 5 Devices, National Security Agency, 2012* is available at `https://cryptome.org/2012/06/apple-ios5-sec.pdf`

iTunes backup

Among the papers and articles related to the iTunes backup structure and analysis the most interesting are the following:

- Information about MBDB and MBDX formats can be found at `http://code.google.com/p/iphonebackupbrowser/wiki/MbdbMbdxFormat`
- *iPhone 3GS Forensics*: *Logical analysis using Apple iTunes Backup Utility*, *Mona Bader, Ibrahim Baggili*, *Small Scale Digital Device Forensics Journal, 2010* is available at `http://securitylearn.net/wp-content/uploads/iOS%20Resources/iPhone%203GS%20Forensics%20Logical%20analysis%20using%20Apple%20iTunes%20Backup%20Utility.pdf`
- *Forensic Analysis of iPhone backups* is available at `http://www.exploit-db.com/wp-content/themes/exploit/docs/19767.pdf`
- Information about Encrypted iTunes backups by Hal Pomeranz in the video *Forensic Lunch*, 2014 is available at `http://www.youtube.com/watch?v=mNLOokxME5A`
- Information about iTunes backup analysis by *Vladimir Katalov, 2013, Elcomsoft Blog* can be found at `http://blog.crackpassword.com/2013/09/itunes_backup_analysis/`
- *Advanced Smartphone Forensics, Vladimir Katalov, ElcomSoft Co. Ltd, 2014* is available at `http://elcomsoft.com/presentations/nullcon2014.pdf`
- *Looking to iPhone backup files for evidence extraction, Clinton Carpene, School of Computer and Security Science, Edith Cowan University* is available at `http://ro.ecu.edu.au/cgi/viewcontent.cgi?article=1091&context=adf`
- *iPhone Backup Files. A penetration Tester's Treasure, Darren Manners, The SANS Institute, 2011* is available at `http://www.sans.org/reading-room/whitepapers/testing/iphone-backup-files-penetration-testers-treasure-33859`
- *Mac OS X: iOS device backup locations* is available at `https://forensicartifacts.com/2015/10/mac-os-x-ios-device-backup-locations/`

iCloud

Various presentations about iCloud illustrate the most interesting concepts from a security and forensics point of view:

- *Ubiquity Forensics*: *Your iCloud and You* by Sarah Edwards is available at `https://github.com/mac4n6/Presentations/blob/master/Ubiquity%20Forensics%20-%20Your%20iCloud%20and%20You/Ubiquity_Forensics-Your_iCloud_and_You.pdf`
- *Advanced Smartphone Forensics, Vladimir Katalov, ElcomSoft Co. Ltd, 2014* is available at `http://elcomsoft.com/presentations/nullcon2014.pdf`
- *iCloud Keychain and iOS 7 Data Protection, Andrey Belenko, ViaForensics, 2013* is available at `https://speakerdeck.com/belenko/icloud-keychain-and-ios-7-data-protection`
- *Modern Smartphone Forensics, Vladimir Katalov, HITBSecConf, 2013* is available at `http://conference.hitb.org/hitbsecconf2013kul/materials/D2T2%20-%20Vladimir%20Katalov%20-%20Cracking%20and%20Analyzing%20Apple's%20iCloud%20Protocol.pdf`
- *Apple iCloud Inside out, Vladimir Katalov, HITBSecConf, 2013* is available at `https://deepsec.net/docs/Slides/2013/DeepSec_2013_Vladimir_Katalov_-_Cracking_And_Analyzing_Apple_iCloud_Protocols.pdf`
- *Cracking and Analyzing Apple iCloud backups, Find My iPhone, Document Storage, Oleg Afonin, REcon, 2013* is available at `https://www.elcomsoft.com/PR/recon_2013.pdf`
- *A Digital Forensic Analysis on the iCloud and its Synchronization to Apple® Devices, Marshall University, 2013* is available at `http://www.marshall.edu/forensics/files/FRIEDMANRACHEL-Research-Paper-08242012.pdf`

Application data analysis

Dedicated articles, presentations, and papers on specific applications data analysis are provided in the following list:

- *Changes Examiners Face with iOS 8.3* is available at `https://www.blackbagtech.com/blog/2015/05/06/changes-examiners-face-with-ios-8-3/`
- *iOS 8.3: the end of iOS Forensics?* is available at `http://blog.digital-forensics.it/2015/06/ios-83-end-of-ios-forensics.html`
- *iOS 8 and its Impact on Investigations* is available at `https://www.blackbagtech.com/blog/2014/09/24/ios-8-and-its-impact-on-investigations/`
- *The iOS of Sauron*: *How iOS Tracks Everything You Do* is available at `https://github.com/mac4n6/Presentations/blob/master/iOS%20of%20Sauron%20-%20How%20iOS%20Tracks%20Everything%20You%20Do/iOS_of_Sauron_04162016.pdf`
- *Parsing iOS "Frequent Locations"* is available at `http://www.mac4n6.com/blog/2015/12/20/parsing-the-ios-frequent-locations`
- *Apple iOS*: *Recently Deleted images* is available at `http://forensenellanebbia.blogspot.it/2015/10/apple-ios-recently-deleted-images.html`
- *Kik Messenger Forensics* is available at `http://www.xploreforensics.com/blog/kik-messenger-forensics.html`
- *Times a' Ticking…to Forensicate the Apple Watch!* presentation is available at `https://github.com/mac4n6/Presentations/blob/master/Apple%20Watch%20-%20Times%20a'%20Tickin'/Apple_Watch_Times_a_Tickin.pdf`
- *iOS Application Forensics* is available at `http://www.scribd.com/doc/57611934/CEIC-2011-iOS-Application-Forensics`
- *Third Party Application Forensics on Apple Mobile Devices, Alex Levinson, Bill Stackpole, Daryl Johnson* is available at `http://www.researchgate.net/publication/224221519_Third_Party_Application_Forensics_on_Apple_Mobile_Devices`

- The *Investigation iOS Phone Images, File Dumps & Backups* article is available at `http://www.magnetforensics.com/investigating-ios-phone-images-file-dumps-backups/`
- The *Analysis Of iOS Notes App* article is available at `http://articles.forensicfocus.com/2013/11/02/analysis-of-ios-notes-app/`
- *Forensic Artifacts of the ChatOn Instant Messaging application, Iqbal A, Marrington A, Baggili I, IEEE* is available at `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=6911538`
- *Forensic analysis of social networking applications on mobile devices, Noora Al Mutawa, Ibrahim Baggili, Andrew Marrington, Elsevier Ltd.* is available at `https://github.com/mac4n6/Presentations/blob/master/Apple%20Watch%20-%20Times%20a'%20Tickin'/Apple_Watch_Times_a_Tickin.pdf`
- The *From iPhone to Access Point* article is available at `http://articles.forensicfocus.com/2013/09/03/from-iphone-to-access-point/`
- *Analysis of WeChat on iPhone, Feng Gao, Ying Zhang, Atlantis Press* can be downloaded from `http://www.atlantis-press.com/php/download_paper.php?id=10185`
- *Know Your Suspect – Uncovering Hidden Evidence from Mobile Devices with Oxygen Forensics* is available at `http://www.forensicfocus.com/c/aid=74/webinars/2014/know-your-suspect---uncovering-hidden-evidence-from-mobile-devices-with-oxygen-forensics/`
- Information about iPhone Call History Database is available at `http://avi.alkalay.net/2011/12/iphone-call-history.html`
- *iPhone Call History, Detective Richard Gilleland* is available at `http://cryptome.org/isp-spy/iphone-spy2.pdf`
- The *Who's Texting? The iOS6 sms.db* article is available at `http://linuxsleuthing.blogspot.com/2012/10/whos-texting-ios6-smsdb.html`
- The *Parsing the iPhone SMS Database* article is available at `http://linuxsleuthing.blogspot.it/2011/02/parsing-iphone-sms-database.html`
- The *Addressing the iOS 6 Address Book and SQLite Pitfalls* article is available at `http://linuxsleuthing.blogspot.it/2012/10/addressing-ios6-address-book-and-sqlite.html`

- The *iOS 6 Photo Streams: "Recover" Deleted Camera Roll Photos* article is available at `http://linuxsleuthing.blogspot.it/2013/05/ios6-photo-streams-recover-deleted.html`
- The *Recovering Data from Deleted SQLite Records: Redux* article is available at `http://linuxsleuthing.blogspot.it/2013/09/recovering-data-from-deleted-sqlite.html`
- The *SQLite Data Parser to Recover Deleted Records* blog is available at `http://az4n6.blogspot.com/2013/11/python-parser-to-recover-deleted-sqlite.html`
- *Forensic Acquisition and Analysis of Tango VoIP, Nhien-An Le-Khac, Christos Sgaras, M-Tahar Kechadi* is available at `https://www.insight-centre.org/sites/default/files/publications/icciet-2014.pdf`
- *Challenges in Obtaining and Analyzing Information from Mobile Devices, Davydov, 2014* is available at `http://computerforensicsblog.champlain.edu/wp-content/uploads/2014/05/Challenges-in-Obtaining-and-Analyzing-Information-from-Mobile-Devices-DavydovO-5-20-2014.pdf`
- The *Advanced Smartphone Forensics* poster by *SANS DFIR* is available at `https://digital-forensics.sans.org/media/DFIR-Smartphone-Forensics-Poster.pdf`

Related books

Other previous books on the same topic are as follows:

- Bommisetty, Satish, Tamma, Rohit, Mahalik Heather, *Practical Mobile Forensics*, by *Packt Publishing, 2014*
- Zdziarski, Jonathan, *Hacking and Securing iOS Applications*, by *O'Reilly, 2012*
- Miller, Charlie, Blazakis, Dyonysus, Dai Zovi, Dino, Esser, Stefan, Iozzo, Vincenzo, Weinmann, Ralf-Philip, *iOS Hacker's Handbook*, by *John Wiley & Sons, 2012*
- Hogg, Andrew, Strzempka, Katie, *iPhone and iOS Forensics*: *Investigation, Analysis and Mobile Security for Apple iPhone, iPad and iOS Devices*, by *Syngress, 2011*
- Casey, Eoghan, *Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet Third Edition*, by *Academic Press, 2011*

- Morrissey, Sean, *iOS Forensic Analysis: for iPhone, iPad, and iPod touch*, by *Apress, 2010*
- Jonathan, Zdziarski, *iPhone Forensics*, by *O'Reilly*, *2008*
- Kubasiak, Ryan, Morrissey, Sean, *Mac OS X, iPod, and iPhone Forensics Analysis Toolkit*, by *Syngress*, *2008*
- Casey, Eoghan, *Digital Evidence and Computer Crime First Edition*, by *Academic Press*, *2000*

B
Tools for iOS Forensics

Acquisition tools

The list of physical acquisition tools (iPhone 2G/3G/3GS/4, iPad 1, iPod touch 1/2/3/4) is as follows:

- **UFED Physical Analyzer**: `http://www.cellebrite.com`
- **Elcomsoft iOS Forensic Toolkit**: `http://www.elcomsoft.com/`
- **AccessData Mobile Phone Examiner Plus**: `http://www.accessdata.com/solutions/digital-forensics/mobile-phone-examiner`
- **Lantern**: `https://katanaforensics.com/`
- **XRY**: `http://www.msab.com/`
- **iXAM forensics**: `http://www.ixam-forensics.com/`
- **iPhone data protection tools**: `https://code.google.com/p/iphone-dataprotection/`
- **Paraben's Device Seizure**: `https://www.paraben.com/device-seizure.html`

For physical acquisition tools on jailbroken iPhone 4s/5/5c, iPad 2/3/4, iPad mini 1 you can use Elcomsoft iOS Forensic Toolkit.

For advanced logical acquisition tools (all models) you can choose UFED Physical Analyzer and Oxygen Forensic.

A list of backup acquisition tools (all models) is as follows:

- **iTunes**: `https://www.apple.com/itunes/download/`
- **Libimobiledevice**: `http://www.libimobiledevice.org/`
- **iMobileDevice**: `http://quamotion.mobi/iMobileDevice/Download`
- **UFED Physical Analyzer/UFED 4PC/Ufed Touch**: `http://www.cellebrite.com`
- **Oxygen Forensic® Suite Standard/Analyst**: `http://www.oxygen-forensic.com/en/`
- **Mobiledit Forensic**: `http://www.mobiledit.com/forensic`
- **AccessData Mobile Phone Examiner Plus**: `http://www.accessdata.com/solutions/digital-forensics/mobile-phone-examiner`
- **Lantern**: `https://katanaforensics.com/`
- **XRY**: `http://www.msab.com/`
- **Mobilyze**: `https://www.blackbagtech.com/software-products/mobilyze-9/mobilyze.html`
- **SecureView**: `http://secureview.us/`
- **Paraben's Device Seizure**: `https://www.paraben.com/device-seizure.html`

iDevice browsing tools and other non-forensic tools

A list of iDevice browsing tools and other non-forensic tools is as follows:

- **iFunBox**: `http://www.i-funbox.com/`
- **iBackupBot**: `http://www.icopybot.com/itunes-backup-manager.htm`
- **iTools**: `http://sale.itools.cn/`

- **Wondershare Dr.Fone iOS**: `http://www.wondershare.com/data-recovery-mac/mac-iphone-data-recovery.html`
- **iSkysoft iPhone Data Recovery**: `http://www.iskysoft.com/iphone-data-recovery/`
- **iMazing**: `http://imazing.com/`
- **iExplorer**: `http://www.macroplant.com/iexplorer/`
- **Syncios**: `http://www.sync-droid.com/`
- **PhoneView**: `http://www.ecamm.com/mac/phoneview/`

iDevice backup analyzer

A list of iDevice backup analyzers is as follows:

- **UFED Physical Analyzer/UFED 4PC/Ufed Touch**: `http://www.cellebrite.com`
- **Oxygen Forensic® Suite Standard/Analyst**: `http://www.oxygen-forensic.com/en/`
- **Elcomsoft Phone Viewer**: `http://www.elcomsoft.com/epv.html`
- **Mobiledit Forensic**: `http://www.mobiledit.com/forensic`
- **AccessData Mobile Phone Examiner Plus**: `http://www.accessdata.com/solutions/digital-forensics/mobile-phone-examiner`
- **iPhone Backup Analyzer**: `https://github.com/PicciMario/iPhone-Backup-Analyzer-2`
- **iPhone Analyzer**: `https://sourceforge.net/projects/iphoneanalyzer/`
- **iPhone Backup Browser**: `https://code.google.com/p/iphonebackupbrowser/`
- **Super Crazy Awesome iPhone Backup Extractor**: `http://supercrazyawesome.com/`
- **Apple iTunes Backup Parser EnScript**: `http://www.proactivediscovery.com/apple-itunes-backup-parser/`
- **iBackupBot**: `http://www.icopybot.com/itunes-backup-manager.htm`
- **iPhone Backup Extractor**: `http://www.iphonebackupextractor.com/`

- **iPhone Backup Viewer**: `http://www.imactools.com/iphonebackupviewer/`
- **iBackup Extractor**: `http://www.wideanglesoftware.com/ibackupextractor/`
- **Smsiphone.org**: `http://www.smsiphone.org/`
- **Logical iOS Forensic Examiner**: `http://www.unhcfreg.com/#!datasetsandtools/c18k6`
- **iTunes Backup Extractor**: `http://www.backuptrans.com/itunes-backup-extractor.html`

iDevice encrypted backup

A list of tools to analyze an iDevice encrypted backup is as follows:

- **Elcomsoft Phone Password Breaker**: `http://www.elcomsoft.com/eppb.html`
- **iPhone Backup Unlocker**: `http://www.windowspasswordsrecovery.com/product/iphone-backup-unlocker.htm`
- **Mbdb file parser**: `https://github.com/halpomeranz/mbdbls`

iCloud Backup

A list of tools to analyze an iCloud Backup is as follows:

- **Elcomsoft Phone Password Breaker**: `http://www.elcomsoft.com/eppb.html`
- **Passware Kit Forensic**: `https://www.passware.com/kit-forensic/`
- **iPhone Backup Extractor**: `http://www.iphonebackupextractor.com/`
- **Wondershare Dr.Fone iOS**: `http://www.wondershare.com/data-recovery-mac/mac-iphone-data-recovery.html`
- **iSkysoft iPhone Data Recovery**: `http://www.iskysoft.com/iphone-data-recovery/`
- **iPhone Data Recovery**: `http://www.tenorshare.com/products/iphone-data-recovery-win.html`
- **InflatableDonkey**: `https://github.com/horrorho/InflatableDonkey`
- **iLoot**: `https://github.com/hackappcom/iloot`

Jailbreaking tools

For more information on jailbreaking tools, refer to the iPhone Wiki jailbreaking tools page at `http://theiphonewiki.com/wiki/Jailbreak.`

iOS 9

For iOS 9, refer to the following list:

- **Pangu**: `http://en.pangu.io/`

iOS 8

For iOS 8, refer to the following list:

- **Pangu**: `http://en.8.pangu.io/`
- **Taig**: `http://www.taig.com/en/`

iOS 7

For iOS 7, refer to the following list:

- **Pangu**: `http://en.7.pangu.io/`
- **Evasi0n7**: `http://evasi0n.com/`
- **Geeksn0w**: `http://geeksn0w.it/`

iOS 6

For iOS 6, refer to the following list:

- **Evasi0n**: `http://evasi0n.com/iOS6/`
- **Redsn0w**: `http://blog.iphone-dev.org/tagged/redsn0w`
- **Sn0wbreeze**: `http://ih8sn0w.com/`
- **P0sixspwn**: `http://p0sixspwn.com/`

Data analysis

All the acquisition tools previously illustrated also have analysis features; for this reason here we list the tools only dedicated to data analysis/parsing.

Forensic toolkit

A list of forensic toolkits is as follows:

- **AccessData FTK**: `http://accessdata.com/solutions/digital-forensics/forensic-toolkit-ftk`
- **GuidanceSoftware Encase Forensic**: `https://www.guidancesoftware.com/products/Pages/encase-forensic/overview.aspx`
- **X-Ways Forensics**: `http://www.x-ways.net/forensics/index-m.html`
- **WinHex**: `http://www.x-ways.net/winhex/`
- **BlackBag Blacklight**: `https://www.blackbagtech.com/software-products/blacklight-7/blacklight.html`

SQLite viewer

The tools to analyze SQLite databases are as follows:

- **SQLite Database Browser**: `http://sqlitebrowser.org/`
- **SQLite Expert**: `http://www.sqliteexpert.com/`
- **SQLite Studio**: `http://sqlitestudio.pl/`
- **SQLite Manager**: `https://addons.mozilla.org/en-US/firefox/addon/sqlite-manager/`
- **SQLite Spy**: `http://www.yunqa.de/delphi/doku.php/products/sqlitespy/index`
- **SQLite Forensic Reporter**: `http://www.filesig.co.uk/sqlite-forensic-reporter.html`

SQLite record carver

The tools for SQLite record carver are as follows:

- **SQLite Recovery Python Parser**: `http://az4n6.blogspot.it/2013/11/python-parser-to-recover-deleted-sqlite.html` and `https://github.com/mdegrazia/SQLite-Deleted-Records-Parser`
- **Epilog**: `http://www.cclgroupltd.com/product/epilog-sqlite-forensic-tool/`
- **Oxygen Forensics SQLite Viewer**: `http://www.oxygen-forensic.com/en/features/analyst/data-viewers/sqlite-viewer`
- **SQLite Recovery**: `http://sandersonforensics.com/forum/content.php?190-SQLite-Recovery`
- **Undark**: `http://pldaniels.com/undark/`

Plist viewer

The tools to analyze the Plist files are as follows:

- **Plist Editor Pro for Windows**: `http://www.icopybot.com/plist-editor.htm`
- **Oxygen Forensics Plist Viewer**: `http://www.oxygen-forensic.com/en/features/analyst/data-viewers/plist-viewer`
- **PlistEdit Pro**: `http://fatcatsoftware.com/plisteditpro/`
- **Pip**: `http://www.cclgroupltd.com/product/pip-xml-and-plist-parser/`

iOS analysis suite

The most interesting iOS analysis suites are as follows:

- **Internet Evidence Finder**: `http://www.magnetforensics.com/`
- **Belkasoft Evidence Center**: `http://belkasoft.com/ec`
- **BlackBag Blacklight**:

`https://www.blackbagtech.com/software-products/blacklight-6/blacklight.html`

- **iPhone Tools**: `https://code.google.com/p/linuxsleuthing/downloads/list`

App analysis tools

The app analysis tools are listed as follows:

- **Elcomsoft Explorer for WhatsApp**: `https://www.elcomsoft.com/exwa.html`
- **SkypeExtractor**: `http://www.skypextractor.com/`
- **SkypeLogView**: `http://nirsoft.net/utils/skype_log_view.html`
- **Safari Forensic Tools**: `http://jafat.sourceforge.net/files.html`
- **Andriller**: `http://andriller.com/`
- **iPhone History Parser**: `http://az4n6.blogspot.it/2014/07/safari-and-iphone-internet-history.html`
- **iThmb Converter**: `http://www.ithmbconverter.com/`
- **Ultra File Opener**: `http://www.ultrafileopener.com/formats/ithmb/`
- **class-dump-z**: `https://code.google.com/p/networkpx/wiki/class_dump_z`
- **Keychain Dumper**: `https://github.com/ptoomey3/Keychain-Dumper`

Consolidated.db

The tools for `Consolidated.db` are as follows:

- **iStalkr**: `http://www.evigator.com/free-apps/`
- **iPhone Tracker**: `http://petewarden.github.io/iPhoneTracker/`
- **iOS Tracker**: `http://tom.zickel.org/iostracker/`

App reverse engineering tools

The app reverse engineering tools are as follows:

- **class-dump-z**: `https://code.google.com/p/networkpx/wiki/class_dump_z`
- **Keychain Dumper**: `https://github.com/ptoomey3/Keychain-Dumper`
- **Dump Decrypted**: `https://github.com/stefanesser/dumpdecrypted`
- **Read Mem**: `https://github.com/gdbinit/readmem`
- **iOS Reverse Engineering Toolkit (iRET)**: `https://github.com/S3Jensen/iRET`
- **Idb**: `https://github.com/dmayer/idb`

C
Self-test Answers

Chapter 1: Digital and Mobile Forensics

Question No.	Correct option
1	3
2	4
3	2
4	3

Chapter 2: Introduction to iOS Devices

Question No.	Correct option
1	3
2	3
3	4
4	3
5	4
6	3

Chapter 3: Evidence Acquisition from iDevices

Question No.	Correct option
1	3
2	1
3	1
4	2
5	2

Chapter 4: Evidence Acquisition and Analysis from iTunes Backup

Question No.	Correct option
1	1
2	2
3	4
4	2

Chapter 5: Evidence Acquisition and Analysis from iCloud

Question No.	Correct option
1	3
2	2
3	4
4	3

Chapter 6: Analyzing iOS Devices

Question No.	Correct option
1	1
2	4
3	2
4	3
5	2
6	3
7	1
8	3

Chapter 7: Applications and Malware Analysis

Question No.	Correct option
1	3
2	3
3	4
4	1
5	2

Index