Elcomsoft iOS Forensic Toolkit

This document contains information about using Elcomsoft iOS Forensic Toolkit for Microsoft Windows and Mac OS X and technical information required to understand the internal working of the Toolkit.

1. Disclaimer

Elcomsoft iOS Forensic Toolkit is based on the work of many people, both within and outside of Elcomsoft. In particular, loading the Toolkit Ramdisk onto the iOS device and Ramdisk general structure are based on tools which are publicly available (or can be obtained with publicly available tools). Authors of those tools may or may not permit commercial usage or they can impose other restrictions. Therefore it is End Users' ultimate responsibility to ensure that they are not violating any license agreements. For full license terms see **`EULA.pdf`** provided with the Toolkit.

2. Requirements

Elcomsoft iOS Forensic Toolkit requires a computer running Windows XP or Windows 7 or Mac OS X from 10.7 (Lion) to 10.11 (El Capitan), as well as iTunes 10.6 or later installed.

Note: Elcomsoft iOS Forensic Toolkit does not work with old non-jailbroken devices (iPhone 4 and older) through the DFU mode on OS X 10.10.5 and 10.11 (El Capitan) at this time. To work with these devices, you have to use Windows version, or OS X 10.10.4 or older.

3. General Description

Elcomsoft iOS Forensic Toolkit is a set of tools aimed at making the acquisition of iOS devices easier. It consists of Toolkit Ramdisk and set of tools to load this Ramdisk onto the iOS device.

When loaded onto the iOS device, Toolkit Ramdisk will allow an analyst to:

- obtain physical (dd-style) dump of root (system) and user (data) partitions
- obtain logical snapshot of user partition
- access device file system
- extract from the iOS device all keys required to decrypt user (data) filesystem as well as keychain items
- run passcode recovery attack

4. Usage

Toolkit can be used in two modes: guided and manual. Guided mode features menu-based user interface for typical tasks and while manual mode lets you interact with tools directly using command-line interface.

Toolkit is shipped with USB protection dongle. This dongle must be connected to the PC while Toolkit is running.

Please read this document carefully before attempting acquisition of an iOS device. Also, you are at all times advised to carefully read output produced by the script driver (in Guided Mode) or by manually executed commands (in Manual Mode).

4.1. Guided Mode

You can start Guided mode by double-clicking on **Toolkit.cmd** (Windows) or **Toolkit.command** (Mac OS X) file in the directory where you have unpacked or copied Toolkit files. This should open console/terminal window and present text-based menu:

```
1  ENTER DFU       - Help putting device into DFU mode
2  LOAD RAMDISK    - Load tools onto the device
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
```

You can accomplish different tasks by selecting corresponding menu items.

When running in Guided Mode, Toolkit logs all related activity to a text file. Each time Toolkit is started, a new log file is created in current directory and output of all invoked commands as well as user choices is written to that file. File name is created based on current universal coordinated time and date and is of the following form: **YYYYMMDD_hhmmssZ.log**.

Enter DFU Mode

The first step in iOS device acquisition is to put the device into DFU (Device Firmware Upgrade) mode. You can do this by selecting menu item ‘1’ and following onscreen instructions:

- Make sure iOS device is connected to the computer with USB cable and is switched off. If it is not connected – connect it and if it switches on switch it off.
- Press Sleep/Power (right-top corner) and Home (bottom center) buttons and hold them for about 10 seconds.
- Release Sleep/Power (right-top corner) button but continue to hold Home (bottom center) button for about 10 seconds.
- The iOS device should be in DFU mode. Please note that there is no visual cue on the device indicating that it has entered DFU mode.

When the device is in DFU mode its screen is blank and the device looks like it is switched off. If the device screen shows the iTunes or Apple logo then the device is not in DFU mode. You should reboot the device and try putting it into DFU once again.

If you have problems putting the device into DFU mode please make sure that it is the only Apple mobile device connected to the computer and that iTunes is not running. You can also try the alternative method:

- Turn off the device.
- Hold the Sleep/Power button for 3 seconds
- Hold the Home button without releasing the Power button for 10 seconds
- Release the Sleep/Power Button but keep holding the Home button for 10 seconds

Load the Toolkit Ramdisk

Note: In order to load the Ramdisk on the device, device must be put into DFU mode first, and USB dongle must be connected to the computer.

After the iOS device has been put into DFU mode you can load the Toolkit Ramdisk with the acquisition tools. To do so select menu item '2' from the main menu or answer 'y' to the prompt following the DFU procedure.

The script automatically detects the type of device connected to the computer, and the Ramdisk loading process starts. The process itself relies on several publicly available tools which are distributed along with the Toolkit for your convenience. While the Ramdisk is loading, the screen of the device may turn white. When Ramdisk is loaded, the device screen will show Elcomsoft logo. This means that Ramdisk is loaded and device is ready for further acquisition.

Note: sometimes (for some very old models) device type cannot be detected automatically; in that case, Toolkit prompts you to select from a few.

Recover device passcode

Note: In order to recover device passcode of the iOS device, the Ramdisk must be first loaded on the iOS device.

You may use iOS Forensic Toolkit to attempt recovery of the device passcode. To do this select menu item '3' from the main menu. After confirming that Ramdisk has already been loaded on the iOS device, you get the following menu:

```
1  Only show passcode type
2  Check 4-digit PINs
3  Perform a wordlist attack
4  Set custom passcode recovery parameters
```

If device is protected with a passcode, it stores passcode type to determine what kind of keyboard to present to user when unlocking the device. There are three possible passcode types:

- Type 0 corresponds to Simple passcode. Passcode is exactly 4 digits
- Type 1 corresponds to passcode composed only of digits and with length not equal to 4
- Type 2 corresponds to alphanumeric passcode of any length

So if you are not sure what passcode is set on the device, use the first option to check.

With option [2], program will attempt to recover 4-digit passcode (so-called 'Simple passcode'). Recovering of 4-digit passcode should take no more than 10 to 40 minutes depending on the device type.

Option [3] allows you to test the passcodes from the file you supply, so it is in fact the "dictionary atrtack". No other parameters should be set, just the wordlist file.

With option [4], you can perform the brute-force attack; just set the password length (you cannot currently set the length range, but just the fixed length; if you do not know it, you

will have to try all possibilities) and the character set, i.e. the symbols the passcode may contain.

Finally you should select the name of the file to save the password to (when/if found), and the recovery starts.

Note: if the device has the simple passcode set (type 0), then the Toolkit always checks for all 4-digit combinations, regardless the options you set.

Extract encryption keys and keychain data from the iOS device

Note: In order to extract encryption keys from the iOS device, the Ramdisk must be first loaded on the iOS device.

You may use the iOS Forensic Toolkit to extract encryption keys required to decrypt files stored on the user partition and Keychain contents. To begin key extraction process select menu item '4' from the main menu. After confirming that Ramdisk has already been loaded on the iOS device, you will be prompted for the following information:

- iOS device passcode. If you know the passcode for the device, you can enter it now. If you don't know it, or the device is not protected with the passcode, you can leave this field blank.

- Escrow file (**iOS 4.x only**). If you have access to computer(s) to which iOS device has been connected or synced then you can provide escrow file acquired from one of those computers. Having correct escrow file allows you to decrypt all data stored on the iOS device even without correct passcode. Thus, if you already have passcode (or device is not protected with passcode) then you do not really need escrow file. Vice versa is also true: if you have valid escrow file then you do not need to recover device passcode to be able to decrypt data stored on the device. The escrow files are stored in the following folder:

  Windows XP
  %ALLUSERSPROFILE%\Application Data\Apple\Lockdown

  Windows 7
  %ALLUSERSPROFILE%\Apple\Lockdown

  Mac OS X
  /var/db/lockdown

  This folder is typically hidden. To open it in Windows Explorer press **Win-R** (in Finder, use Go - Go to Folder... menu item) and type in the path. Each file in this directory corresponds to one iOS device, file name being the device UUID. If you do not have escrow file or would like not to use it you can leave this field blank as well. Please note that this will not work on devices running iOS 5-9 due to improved security.

- File name to save keys to. Created file will be in Apple Property List (plist) format.

After the key extraction utility has completed you are advised to study the output it has produced. Pay particular attention on the following messages:

- Passcode. Utility reports if the correct passcode has been supplied as well as if device is not passcode protected.
- Backup password. Utility analyzes device keychain and shows iTunes backup password, if any.
- Escrow records. If you have provided an escrow file you should check the program output to see if a matching pairing record has been found. If it was not found, you might want to try another escrow file (possibly obtained from another computer).
- Apple ID and password. The key extraction utility tries to get them from the device, though they are not always available (it depends on the system configuration and some settings, i.e. whether iMessage and FaceTime are configured on the device, and whether iCloud account has been set up).

Decrypt device keychain data

Note: In order to decrypt device keychain data you need to provide device keys. Please refer to previous sections for information about obtaining them. Decryption is done off-line and does not require iOS device to be connected.

To decrypt device keychain data using the Toolkit, please select menu item '5' from the main menu. You will then be asked to provide device keys file and name of the file where decrypted keychain data should be written.

Acquire physical image(s) of iOS device filesystem(s)

Note: In order to acquire iOS filesystem image(s), the Ramdisk must be first loaded on the iOS device as described above.

You may use iOS Forensic Toolkit to obtain image(s) of iOS device filesystem(s). To do this select menu item '6' from the main menu.

You will be presented with the list of partitions depending on the current device.

Most iOS devices have two partitions (System and User) and their names differ between iOS versions:

- *iOS 4: System is* ***disk0s1*** *and User is* ***disk0s2s1***
- *iOS 5..9: System is* ***disk0s1s1*** *and User is* ***disk0s1s2***

System partitions are typically around 1-1.5 Gb in size and contain system files. User partitions sizes depend on the device model and are usually around 15 Gb, 31 Gb or 64 Gb. The User partition holds most of user-generated data and thus is the primary acquisition objective. Most of the files on the User partition are encrypted and to decrypt them keys must be obtained from the device (see next section).

Toolkit also supports acquiring physical image of older devices, such as original iPhone, iPhone 3G, and iPod Touch 1st and 2nd generations. Please not that due to absence of hardware encryption module, User partitions of those devices are not encrypted, even if

device is running iOS 4.x. Partition names for those devices are **`disk0s1`** (System) and **`disk0s2`** (User).

After you select which partition to image you will be prompted for file name to save image to. If you do not provide full path the file will be stored in current directory (Windows) or in current user's home directory (i.e. **`/Users/`*`username`***) (Mac OS X). Created file is a raw disk image and you would typically use **`.dmg`** file extension for it. Please make sure that the target volume or directory where you are storing the image is large enough to accommodate the image file which can be up to 128 Gb for some device models.

After specifying the file name the imaging process begins. This can take significant time depending on the selected partition and device model. Approximate partition duplication and transfer times are as follows:

Device	Duration
iPhone 4 32Gb	30 min
iPhone 3GS 32Gb	40 min
iPad 16Gb	20 min
iPad 64Gb	55 min

Imaging the system partition usually takes no more than 5-7 minutes. During the imaging process you will see some basic progress information.

Decrypt encrypted partition image

Note: In order to decrypt partition image you need to provide encrypted image and device keys. Please refer to previous sections for information about obtaining them. Decryption is done off-line and does not require iOS device to be connected.

To decrypt image using the Toolkit, please select menu item '7' from the main menu. You will then be asked to provide file name of encrypted image file, device keys file and name of the file where decrypted image should be written. After decryption is complete you will be provided with SHA1 hash of decrypted image file. The decryption program will also create three log files with the lists of the files that are not encrypted, that could be decrypted and that couldn't be decrypted. Log files are stored in the current user's home directory (in Mac version of the Toolkit) or in the same directory where decrypted image is stored (in Windows version of the Toolkit).

Acquire users' files from iOS device as a tarball

Note: In order to acquire users' files from the device, the Ramdisk must be first loaded on the iOS device.

During logical acquisition only actual files are copied to the computer (retaining the

directory structure). The process is generally significantly faster than physical acquisition because data residing in unallocated areas of the partition does not have to be transferred.

Logical acquisition currently can not access files with protection classes requiring encryption based on user-supplied passcode. Such files are not included in logical image.

You may use iOS Forensic Toolkit to obtain files stored on the User partition of the iOS device (logical image). To do this select menu item '8' from the main menu.

You will be prompted for file name to save logical image to. If you do not provide full path the file will be stored in current directory (Windows), or in current user's home directory (Mac OS X). Created file is a TAR archive and you would typically use **`.tar`** file extension for it. Please make sure that the target volume or directory where you are storing the logical image is large enough to accommodate the file which can be up to 128 Gb depending on device model and amount of data stored by the user on the device.

After specifying the file name the imaging process begins. It can take noticeable time depending on the device model and the amount of data stored on the device. During the process you will be presented some basic progress information.

Reboot the iOS device

Note: In order to reboot the iOS device, the Ramdisk must be first loaded on the iOS device.
To reboot the iOS device, select menu item '9' from the main menu.

4.2. Manual Mode

Manual mode allows for greater flexibility and thus is the recommended way of doing iOS device acquisition if you are comfortable with using command-line tools.

Enter DFU mode

The first step in iOS acquisition is to put device into DFU (Device Firmware Upgrade) mode, the same way as described in the appropriate section under the **Guided Mode**.

Load the Toolkit Ramdisk

Note: In order to load the Ramdisk on the device, device must be put into DFU mode first, and USB dongle must be connected to the computer.

To load the Ramdisk to the device, open Command Prompt (Start - All Programs - Accessories - Command Prompt) or Terminal.app (Applications - Utilities - Terminal), navigate to the directory where you have unpacked or copied Toolkit files and start the following commands (replace ***`ibss`***, ***`ibec`***, ***`kernel`*** and ***`devicetree`*** according to the table):

iPhone 3GS and later devices (iPhone 4, iPod Touch 3/4, iPad)

Windows:

```
.\win32\tetheredboot.exe -d -x -i common/ibss -j common/ibec
.\win32\itunnel_mux.exe --decrypt --kernelcache common/kernel -- devicetree
```

```
common/devicetree --ramdisk common/ramdisk-5.dmg
```

Mac OS X:

```
./macosx/tetheredboot -d -x -i common/ibss -j common/ibec
./macosx/itnl --decrypt --kernelcache common/kernel --devicetree
common/devicetree --ramdisk common/ramdisk-5.dmg
```

	iPhone 3Gs	iPhone 4 (GSM)	iPhone 4 (CDMA)
	iPhone2,1	iPhone3,1	iPhone3,3
ibss	iBSS.n88	iBSS.n90	iBSS.n92
ibec	iBEC.n88	iBEC.n90	iBEC.n92
kernel	kernelcache.n88	kernelcache.n90	kernelcache.n92
devicetree	DeviceTree.n88	DeviceTree.n90	DeviceTree.n92

	iPod Touch 3	iPod Touch 4	iPad 1
	iPod3,1	iPod4,1	iPad1,1
ibss	iBSS.n18	iBSS.n81	iBSS.k48
ibec	iBEC.n18	iBEC.n81	iBEC.k48
kernel	kernelcache.n18	kernelcache.n81	kernelcache.k48
devicetree	DeviceTree.n18	DeviceTree.n81	DeviceTree.k48

iPod Touch 2nd Generation

```
.\win32\tetheredboot.exe -d -x -i common/iBSS.n72
.\win32\itunnel_mux.exe --decrypt --kernelcache common/kernelcache.n72 --
devicetree common/DeviceTree.n72 --ramdisk common/ramdisk-4.dmg

./macosx/tetheredboot -d -x -i common/iBSS.n72
./macosx/itnl --decrypt --kernelcache common/kernelcache.n72 --devicetree
common/DeviceTree.n72 --ramdisk common/ramdisk-4.dmg
```

Original iPhone and iPod Touch 1st Generation

```
.\win32\itunnel_mux.exe --decrypt --wtf common/WTF.8900 --ibss common/ibss --
kernelcache common/kernelcache.8900 --devicetree common/devicetree --ramdisk
common/ramdisk-3.dmg --ramdisk-command "0x90000000"

./macosx/dfu-util common/WTF.8900
./macosx/dfu-util common/ibss
(Wait for 15 seconds to let device boot)
```

```
./macosx/itnl --decrypt --kernelcache common/kernelcache.8900 -- devicetree
common/devicetree --ramdisk common/ramdisk-3.dmg -- ramdisk-command "0x90000000"
```

	iPhone	iPod Touch 1st
	iPhone1,1	iPod1,1
ibss	iBSS.m68	iBSS.n45
devicetree	DeviceTree.m68	DeviceTree.n45

iPhone 3G

```
./win32/itunnel_mux.exe --decrypt --wtf common/WTF.8900 --ibss common/iBSS.n82 -
-kernelcache common/kernelcache.n82 --devicetree common/DeviceTree.n82 --ramdisk
common/ramdisk-4.dmg

./macosx/dfu-util common/WTF.8900
./macosx/dfu-util common/iBSS.n82
(Wait for 15 seconds to let device boot)
./macosx/itnl --decrypt --kernelcache common/kernelcache.n82 -- devicetree
common/DeviceTree.n82 --ramdisk common/ramdisk-4.dmg
```

As soon as Ramdisk has been loaded, the device screen will show Elcomsoft logo.

Enable connection forwarding to the device

To enable communication to the device, a network connection forwarding must be established. This can be accomplished by issuing following command:

```
.\win32\itunnel_mux.exe --iport 22 --lport 2022

./macosx/itnl --iport 22 --lport 2022
```

This will establish connection forwarding from port 2022 on `localhost` to port 22 on the iOS device. Port 22 is where the SSH server is expecting connections on the iOS device and thus should be fixed and port 2022 can be replaced with any other available port number.

This command will not terminate and you should not interrupt it – it must continue running until you have completed acquisition tasks. Should the connection be interrupted for any reason (such as the host computer entering power save mode, etc.) then you must re-establish the connection by beginning the entire process from the Section above.

Recover device passcode

Note: In order to recover device passcode of the iOS device, the RAMdisk must be first loaded on the iOS device.

Elcomsoft iOS Forensic Toolkit uses **`passcode`** utility to perform brute-force attack on device passcode.

Before using **passcode** you must mount User partition so that utility could read system keybag. To mount the user partition, execute the following command:

On device running iOS 4:

```
.\win32\ssh.exe -p 2022 root@localhost mount -t hfs /dev/ disk0s2s1 /mnt2

ssh -p 2022 root@localhost mount -t hfs /dev/disk0s2s1 /mnt2
```

On device running iOS 5..9:

```
.\win32\ssh.exe -p 2022 root@localhost mount -t hfs /dev/ disk0s1s2 /mnt2

ssh -p 2022 root@localhost mount -t hfs /dev/disk0s1s2 /mnt2
```

On iPhone 3G use the following commands instead:

```
.\win32\ssh.exe -p 2022 root@localhost fsck_hfs /dev/disk0s2
.\win32\ssh.exe -p 2022 root@localhost mount -t hfs /dev/disk0s2 /mnt2

ssh -p 2022 root@localhost fsck_hfs /dev/disk0s2
ssh -p 2022 root@localhost mount -t hfs /dev/disk0s2 /mnt2
```

This only needs to be done once per iOS device reboot because partition remain mounted until device is switched off or rebooted.

Quick help on using **passcode** can be obtained by issuing:

```
.\win32\ssh.exe -p 2022 root@localhost passcode -h

ssh -p 2022 root@localhost passcode -h
```

Which will produce the following result:

```
Usage: passcode [-h] [-t] -l length -c charset
-h                    - Display help (you are looking on it right now
-t                    - Show passcode type and quit. Possible values:
                        0: Simple passcode (4 digits)
                        1: Digits-only passcode (length is not 4)
                        2: Alphanumeric passcode
-i                    - Read passcodes from stdin
-nc                   - Do not check for common simple passcodes
-l length             - Passcode length
-c charset             - Charset to use
```

If device is protected with a passcode, it stores passcode type to determine what kind of keyboard to present to user when unlocking the device. There are three possible passcode types:

- Type 0 corresponds to Simple passcode. Passcode is exactly 4 digits
- Type 1 corresponds to passcode composed only of digits and with length not equal to 4
- Type 2 corresponds to alphanumeric passcode of any length

Launching **passcode** utility with **-t** parameter will show passcode type for current device:

```
.\win32\ssh.exe -p 2022 root@localhost passcode -t

ssh -p 2022 root@localhost passcode -t
```

Once passcode type is known you can launch actual bruteforce command by issuing:

```
.\win32\ssh.exe -p 2022 root@localhost passcode -l length –c charset

ssh -p 2022 root@localhost passcode -l length -c charset
```

For example, this command will check all 5-digit passcodes composed of digits:

```
.\win32\ssh.exe -p 2022 root@localhost passcode -l 5 -c 1234567890

ssh -p 2022 root@localhost passcode -l 5 -c 1234567890
```

Please note that passcode recovery can't currently be done off the device and is thus quite slow. Typical recovery speeds are less than 3 passcodes/second for iPhone 3GS, 6-7 passcodes/second for iPhone 4 and iPad, and up to 20 passcodes/second on iPhone 5.

Extract encryption keys and keychain data from the iOS device

Note: In order to extract encryption keys from the iOS device, the Ramdisk must be first loaded on the iOS device and connection forwarding must be established.

Elcomsoft iOS Forensic Toolkit uses **dumpkeys** utility to extract encryption keys required to decrypt files stored on user partition and keychain contents.

Before using **dumpkeys** you must mount User partition so that utility could read system keybag and escrow pairing records. To mount user partition, start the following command:

On device running iOS 4:

```
.\win32\ssh.exe -p 2022 root@localhost mount -t hfs /dev/disk0s2s1 /mnt2

ssh -p 2022 root@localhost mount -t hfs /dev/disk0s2s1 /mnt2
```

On device running iOS 5-9:

```
.\win32\ssh.exe -p 2022 root@localhost mount -t hfs /dev/disk0s1s2 /mnt2

ssh -p 2022 root@localhost mount -t hfs /dev/disk0s1s2 /mnt2
```

On the iPhone, iPhone 3G and iPod Touch 1st and 2nd Generation use the following commands instead:

```
.\win32\ssh.exe -p 2022 root@localhost fsck_hfs /dev/disk0s2
.\win32\ssh.exe -p 2022 root@localhost mount -t hfs /dev/disk0s2 /mnt2

ssh -p 2022 root@localhost fsck_hfs /dev/disk0s2
ssh -p 2022 root@localhost mount -t hfs /dev/disk0s2 /mnt2
```

This only needs to be done once per iOS device reboot because partition remain mounted until device is switched off or rebooted.

Quick help on using **dumpkeys** can be obtained by issuing:

```
.\win32\ssh.exe -p 2022 root@localhost dumpkeys -h

ssh -p 2022 root@localhost dumpkeys -h
```

Which will produce the following result:

```
Usage: dumpkeys [-h] [-p passcode] [-e]
-h                     - Display help
-p passcode            - Use specified passcode (optional)
-e                     - Read escrow keybag from stdin (optional)
```

To provide passcode to **dumpkeys**:

```
.\win32\ssh.exe -p 2022 root@localhost dumpkeys -p passcode
```

To provide escrow keybag to **dumpkeys**:

```
.\win32\ssh.exe -p 2022 root@localhost dumpkeys -e < escrow-file

ssh -p 2022 root@localhost dumpkeys -p passcode
```

dumpkeys uses `stderr` for errors and informational messages and writes Apple XML Property List containing extracted keys to `stdout`, therefore you can use output redirection operators to save exported keys to predetermined file on your computer, e.g.:

```
.\win32\ssh.exe -p 2022 root@localhost dumpkeys -p passcode -e < escrow-file > out-file

ssh -p 2022 root@localhost dumpkeys -p passcode -e < escrow-file > out-file
```

This will redirect `stdout` to file and keep `stderr` (error and information) in the console.

Decrypt device keychain data

Note: In order to decrypt device keychain data you need to provide device keys. Please refer to previous sections for information about obtaining them. Decryption is done off-line and does not require iOS device to be connected.

To decrypt device keychain data using the Toolkit you can either use Guided mode (select menu item '5' in the main menu) or you can invoke decryption program directly by supplying three mandatory arguments:

```
.\win32\keychain.exe keys-file

./macosx/keychain keys-file
```

The program will read device keys and keychain data from the supplied file, decrypt device keychain entries, and output the result to `stdout` in plain text format.

Acquire physical image(s) of iOS device filesystem(s)

Note: In order to acquire iOS filesystem image(s), the Ramdisk must be first loaded on the iOS device and connection forwarding must be established.

Elcomsoft iOS Forensic Toolkit uses **`dd`** to obtain filesystem image(s). To create an image of the System partition, start one of the following commands:

On device running iOS 4:

```
.\win32\ssh.exe -c null -m hmac-md5-96 -p 2022 root@localhost dd bs=1M
if=/dev/rdisk0s1 | .\win32\dd.exe bs=1M of=output-file -- progress
```

```
./macosx/ssh -c null -m hmac-md5-96 -p 2022 root@localhost dd bs=1M
if=/dev/rdisk0s1 | dd of=output-file
```

On device running iOS 5..9:

```
.\win32\ssh.exe -c null -m hmac-md5-96 -p 2022 root@localhost dd bs=1M
if=/dev/rdisk0s1s1 | .\win32\dd.exe bs=1M of=output-file -- progress
```

```
./macosx/ssh -c null -m hmac-md5-96 -p 2022 root@localhost dd bs=1M
if=/dev/rdisk0s1s1 | dd of=output-file
```

To create an image of the User partition, execute one of the following commands:

On device running iOS 4:

```
.\win32\ssh.exe -c null -m hmac-md5-96 -p 2022 root@localhost dd
bs=1M if=/dev/rdisk0s2s1 | .\win32\dd.exe bs=1M of=output-file --
progress
```

```
./macosx/ssh -c null -m hmac-md5-96 -p 2022 root@localhost dd
bs=1M if=/dev/rdisk0s2s1 | dd of=output-file
```

On device running iOS 5..9:

```
.\win32\ssh.exe -c null -m hmac-md5-96 -p 2022 root@localhost dd bs=1M
if=/dev/rdisk0s1s2 | .\win32\dd.exe bs=1M of=output-file -- progress
```

```
./macosx/ssh -c null -m hmac-md5-96 -p 2022 root@localhost dd bs=1M
if=/dev/rdisk0s1s2 | dd of=output-file
```

On the iPhone, iPhone 3G and iPod Touch 1st and 2nd Generation use the following command instead:

```
.\win32\ssh.exe -c null -m hmac-md5-96 -p 2022 root@localhost dd bs=1M
if=/dev/rdisk0s2 | .\win32\dd.exe bs=1M of=output-file -- progress
```

```
./macosx/ssh -c null -m hmac-md5-96 -p 2022 root@localhost dd bs=1M
if=/dev/rdisk0s2 | dd of=output-file
```

These commands will connect to the iOS device using SSH, start **dd** on the device and on the computer and send output provided by the **dd** on the device side to the **dd** running on the computer so that it can be saved in the *output-file*.

Please note that on first SSH connection to the iOS device you may get a warning about unknown key fingerprint for host `localhost`. This warning can be ignored and you should add the key fingerprint to the list of known hosts if SSH requires you to do so.

You may also get a warning about a key fingerprint mismatch for host `localhost`. This may happen if you have connected with SSH to localhost before (possibly redirecting connections to elsewhere). If this error is fatal (i.e. SSH refuses to connect) please consult with SSH manual or Google on how to resolve the issue.

Decrypt encrypted partition image

Note: In order to decrypt partition image you need to provide encrypted image and device keys. Please refer to previous sections for information about obtaining them. Decryption is done off-line and does not require iOS device to be connected.

To decrypt image using the Toolkit, you can either use Guided mode (select menu item '7' in the main menu) or you can invoke decryption program directly by supplying three mandatory arguments:

```
.\win32\hfsdecrypt.exe encrypted-image keys-file decrypted-image

./macosx/hfsdecrypt encrypted-image keys-file decrypted-image
```

After decryption is complete you will be provided with SHA1 hash of decrypted image file.

Acquire users' files from iOS device as a tarball

Note: In order to acquire users' files from the device, the Ramdisk must be first loaded on the iOS device and connection forwarding must be established.

Elcomsoft iOS Forensic Toolkit uses **tar** and **dd** commands to obtain logical filesystem image(s). Before trying to obtain logical image of User partition, it must be mounted. To mount the User partition, start the following command:

On device running iOS 4:

```
.\win32\ssh.exe -c null -m hmac-md5-96 -p 2022 root@localhost mount -t hfs
/dev/disk0s2s1 /mnt2

ssh -p 2022 root@localhost mount -t hfs /dev/disk0s2s1 /mnt2
```

On device running iOS 5..9:

```
.\win32\ssh.exe -c null -m hmac-md5-96 -p 2022 root@localhost mount -t hfs
/dev/disk0s1s2 /mnt2
```



```
ssh -p 2022 root@localhost mount -t hfs /dev/disk0s1s2 /mnt2
```

On iPhone 3G use the following commands instead:

```
.\win32\ssh.exe -c null -m hmac-md5-96 -p 2022 root@localhost fsck_hfs
/dev/disk0s2s1 /mnt2
.\win32\ssh.exe -c null -m hmac-md5-96 -p 2022 root@localhost mount -t hfs
/dev/disk0s2s1 /mnt2

ssh -p 2022 root@localhost fsck_hfs /dev/disk0s2
ssh -p 2022 root@localhost mount -t hfs /dev/disk0s2 /mnt2
```

This only needs to be done once per iOS device reboot because partition remain mounted until device is switched off or rebooted.

To create tarball of files on User partition, start the following command:

```
.\win32\ssh.exe -c null -m hmac-md5-96 -p 2022 root@localhost tar
-c /mnt2 | .\win32\dd.exe of=output-file --progress

./macosx/ssh -c null -m hmac-md5-96 -p 2022 root@localhost tar - c /mnt2 | dd
of=output-file
```

Those commands will connect to the iOS device using SSH, start **`tar`** on the device and send output provided by it to the **`dd`** running on the computer so that it can be saved in the *`output-file`*.

Please note that on first SSH connection to the iOS device you may get a warning about unknown key fingerprint for host `localhost`. This warning can be ignored and you should add the key fingerprint to the list of known hosts if SSH requires you to do so.

You may also get a warning about a key fingerprint mismatch for host `localhost`. This may happen if you have connected with SSH to localhost before (possibly redirecting connections to elsewhere). If this error is fatal (i.e. SSH refuses to connect) please consult with SSH manual or Google on how to resolve the issue.

Please note also that there is no progress information provided by **`dd`** while running. You can monitor progress by watching actual file size of *`output-file`*.

Reboot the iOS device

Note: This section covers rebooting of the iOS device to which the Ramdisk have been loaded. For instructions to reboot iOS devices running other software refer to other sources. To reboot the iOS device, execute the following command:

```
.\win32\ssh.exe -p 2022 root@localhost kill 1

ssh -p 2022 root@localhost kill 1
```

Appendix A. Supported devices

Toolkit currently works with the following devices:

- iPhone
- iPhone 3G
- iPhone 3GS
- iPhone 4 (GSM and CDMA models)
- iPad
- iPod Touch (1st, 2nd, 3rd and 4th generations)

Support for the following models is limited to jailbroken devices only (so the device should be already jailbroken with OpenSSH installed, or you should be able to install the jailbreak yourself, so the passcode is not set or known, and iOS version is compatible with the jailbreak):

- iPhone 4S
- iPhone 5
- iPhone 5C
- iPod Touch (5th generation)
- iPad 2
- iPad with Retina display (3rd and 4th generations)
- iPad Mini

Support for following (64-bit) models is very limited: device should be jailbroken (or can be jailbroken), and only file system (logical) acquisition is possible (TAR FILES):

- iPhone 5S
- iPhone 6, iPhone 6 Plus
- iPhone 6S, iPhone 6 Plus
- iPad Air, iPad Air 2
- iPad Mini 2-4
- iPad Pro

		Physical imaging	Logical imaging	Passcode recovery	Keychain decryption	Disk decryption
iPhone iPhone 3G iPod Touch 1 iPod Touch 2	**iOS 1..3**	+	+	instant	+	N/A
	iOS 4	+	+	+	+	N/A
iPhone 3GS iPod Touch 3 iPad 1	**iOS 3**	+	+	instant	+	N/A
	iOS 4/5	+	+	+	+	+
iPhone 4 iPod Touch 4	**iOS 4..7**	+	+	+	+	+
iPhone 4S iPhone 5/5C iPad 2-4 iPad Mini iPod Touch 5	**iOS 5..9**	+	+	+	+	+
iPhone 5S iPhone 6/6S (Plus) iPad Mini 2-4 iPad Air (2) iPad Pro	**iOS 7..9**	-	+	-	-	-

Appendix B. Acquisition of jailbroken devices

IMPORTANT

Performing acquisition of iPhone 4S and newer devices is currently not forensically sound, meaning that it may and will introduce artifacts into the system. This may affect the admissibility of the obtained evidence. Please document all your actions extensively.

Due to improved security mechanisms in iOS 5..9 it is important that you do not reboot seized device and keep it switched-on if possible (if it was seized in switched on state, of course). This applies to all devices running iOS 5..9 (especially to iPhone 4S, iPad 2 and newer devices) with device passcode set. Doing so will potentially allow to bypass passcode by using escrow keys obtained from the paired PC or Mac (iOS 4), or at least keep the pairing records to be able to perform iTunes backups. Faraday bags and/or airplane mode should be used to prevent device from communicating with cellular or wireless networks.

Jailbreak for iPhone 4S to 6S Plus, iPad, iPad Mini, iPad Pro and iPod Touch (5th gen) running iOS 5 to 9 is public. Jailbreaking allows, among other things, to run acquisition tools. In order to run them, SSH server must be running on the device. Device may already have OpenSSH installed by its owner, or you can install OpenSSH package on the jailbroken device using Cydia package manager. Jailbreak required:

- iOS 6.0-6.1.2: evasi0n
- iOS 6.1.3-6.1.6: p0sixspwn
- iOS 7.0: evasi0n
- iOS 7.1: PanGu
- iOS 8.0-8.4: TaiG
- iOS 9.0-9.0.2: PanGu

Note: jailbreak installation requires the passcode to be not set or known; some jailbreaks also require the *Find My Phone* function to be disabled on the device, and iTunes backup password not set. Refer to jailbreak documentation for more details.

Checking if SSH server is already running on the device

Start the iOS Forensic Toolkit for jailbroken devices (`Toolkit-JB.command` on Mac OS X or `Toolkit-JB.cmd` on Windows). This will automatically establish a tunnel between SSH port (22) on the device and port 3022 on the `localhost`. Now use SSH client to connect to `localhost` on port 3022, e.g. using the following command:

```
ssh -p 3022 root@localhost
```

If SSH session is established, or if you are asked for a password, or if you receive a key fingerprint mismatch error then SSH server is already running on the device. If connection is not established or refused then no SSH server is running on the device and it must be installed prior to performing the acquisition. You can do this using the Cydia package manager installed on the device.

Changing `root` password

The default password for user `root` on the iOS devices is `alpine`. If it does not work, you may need to change it: using any of the tools to access iOS file system (such as i- FunBox or iExplorer) edit file `/private/etc/master.passwd` so that the line corresponding to `root` account looks

like the following:

```
root:/smx7MYTQIi2M:0:0::0:0:System Administrator:/var/root:/bin/sh
```

Saving modified `master.passwd` file back to the device will restore the default `root` password which is `alpine`. You should now be able to establish SSH session with the device.

Running acquisition tools

In previus versions of the Toolkit, you had to use SFTP (SSH File Transfer Protocol) to copy dumpkeys and passcode utilities to the device. With the current version, it is being done automatically by the script when you extract the keys or recover the passcode. The only notable difference is that you will be asked to provide root password (default is 'alpine') at least once, or in some cases every time you perform operation on the device. All acquisition tasks are being performed as usual (you just do not need to enter the device into the DFU mode and load the Ramdisk).

Toolkit functions are available on jailbroken devices

iOS Forensic Toolkit for jailbroken devices includes the same acquisition options as iOS Forensic Toolkit: it can be used to obtain image device disks(s), to obtain archive of users' files, to recover the passcode, and to extract device keys and keychain data. Disk image and keychain decryption also work as usual.

Using the Toolkit with non-jailbroken iPhone 4S and other newer devices

Acquisition of non-jailbroken iPhone 4S, iPad 2 and newer devices is not possible at this time. However, you can jailbreak the device and perform acquisition as described in previous section if the device is running iOS 5 to 9. Please note that jailbreak is currently possible if there is no passcode and no backup password set on the device. If the device has backup password, you can try to recover it using Elcomsoft Phone Password Breaker (https://www.elcomsoft.com/eppb.html).

Using the Toolkit with jailbroken iPhone 4 (and earlier) devices

Acquisition of older jailbroken devices (iPhone 4 etc) is performed by the 'usual' version of the Toolkit (`Toolkit.cmd` or `Toolkit.command`) through the DFU mode.

Using the Toolkit with 64-bit devices (iPhone 5S, iPhone 6(S) (Plus), iPad Air iPad Mini 2-4, iPad Pro

For 64-bit devices, Toolkit functionality is even more limited. All you can do is the logical acquisition using the TAR FILES function (menu item '8').

Appendix C. Troubleshooting

Compatibility

Error: Device is not compatible with this exploit

- **You are working with iPhone 4S+ device, trying to load RAMdisk (after entering the DFU mode). You should just use the second script (*Toolkit-JB*) instead. Of course, the device should be jailbroken (see Appendix A).**
- **You have entered the device into the DFU mode "manually" – without the help of the Toolkit, so skipping the first menu item and going directly to the second one. This is OK, but in this case you should take care yourself that iTunes is not running (Toolkit checks whether the iTunes is running and close it only when selecting [1] from the menu).**

Error: Keys are not valid for this encrypted image (-8)

This message may appear on some jailbroken iPhone 4S and iPhone 5C devices when trying to decrypt the disk image. Unfortunately, there is no fix at the moment; all you can do wil this device is:

- **GET PASSCODE**
- **GET KEYS**
- **DECRYPT KEYCHAIN**
- **TAR FILES**

libusb: 0.000000 error [darwin_transfer_status] transfer error: timed out

This error may appear on Mac version of the Toolkit only, on OS X 10.10.5 or 10.11, when working with old (iPhone 4 and below) devices through the DFU mode. The is currently no solution; you can work with such devices only with Windows version of the Toolkit, or on OS X 10.10.4 or older.

Using Wi-Fi

It is strongly recommended to turn off the Wi-Fi on the device you are working with -- before using the Toolkit. When you start the Toolkit, it quickly shows (after the license validation) the message like "Device connected: {UDID}", and if you see more than one device there (even just two with the same ID, so it looks like duplicate – it happens when you have Wi-Fi connected), the Toolkit may fail.

Reliability/connection

Please make sure that your computer never goes to sleep/hibernate during the acquisition process; even more – if you are using the Toolkit on the laptop, do NOT change from the battery to the external source (or back) until the device is completely acquired. Otherwise, the Toolkit may loose the connection to the device during the acquisition process (especially the one that takes time, such as passcodce recovery or disk imaging), and you will have to start the process from scratch.

IMPORTANT

Performing acquisition of jailbroken devices is currently not forensically sound, meaning that it will produce some artifacts into the system. This may affect the admissibility of the obtained evidence.

Please document all your actions extensively.