

**Investigation of Cryptocurrency Wallets on
iOS and Android Mobile Devices for Potential Forensic Artifacts**

Angelica Montanez

Department of Forensic Science, Marshall University

Summer 2014

The author thanks Michael Younger and Breandan Gleason from Stroz Friedberg LLC, Terry Fenger, PhD, from the Department of Forensic Science, Marshall University, and Christopher Vance from Access Data, Inc. for their insight, guidance, and mentorship during this research project.

Abstract

As their use no doubt increases in the coming years, it is important for those in law enforcement and forensics to be familiar with systems of digital currency. Although the infamous “Silk Road”—described by some as a black-market Amazon or eBay—was shut down by the FBI in late 2013, cryptocurrencies are still being used in illegal transactions. The purpose of this research was to examine the most popular wallet applications for the cryptocurrencies Bitcoin, Litecoin, and Darkcoin on mobile devices for potential forensic artifacts. Using various forensic extraction tools, the data generated from controlled trading was extracted from an Android and an iOS device, parsed, and then analyzed for any data that could potentially link a cryptocurrency wallet, whether active or deleted, to a specific device.

Upon completion of this research, it was determined that the Universal Forensic Extraction Device (UFED) Physical Analyzer successfully harvested data indicating present cryptocurrency wallet application presence on both the iOS and Android devices, but past wallet indicators were extracted only for the Android device. Specifically for the iOS, the iFunBox tool was determined to be useful only for confirmation of active wallet application presence on an iOS device. Specific to Android devices, the Android Debug Bridge (ADB) pull command-line tool could successfully extract a wealth of valuable transaction information for active cryptocurrency wallet applications. In addition to transaction data, the ADB pull was also capable of extracting information indicating present and past wallet presence on the mobile device, but only if the wallet had been installed via a downloaded APK file.

Ultimately, the results of this research may serve to aid law enforcement in connecting unlawful transactions involving these cryptocurrency wallets on Android devices to implicated individual(s) and devices. Further research is still needed to discover a more reliable method for extracting cryptographic wallet data from iOS devices.

I. Introduction

It has been argued that a cover of anonymity increases the likelihood of participation in illegal activities. After recent media illustrations of Bitcoin and alt-coins (all coin alternatives to Bitcoin), most discussions of these cryptographic currencies bring up this connection between criminal intent and anonymity. It is under the false supposition that these cryptocurrencies offer true anonymity in electronic transactions that trading websites such as the Silk Road, essentially a black-market Amazon or eBay, were created and became popular for criminal dealings. Although the Silk Road was shut down by the FBI in 2013 for its part in facilitating the exchange of illegal goods for the cryptocurrency Bitcoin, the existence of similar websites supporting illegal business resolutely persists.

Because these currency systems have been affiliated with illegal activities, it is necessary for them to be forensically researched. Digital forensics is predominantly concerned with user-generated data—to search for signs of user activity amid the software and memory of digital devices. With peer-to-peer transactions as the fundamental purpose of cryptocurrencies, these digital currency systems offer a plethora of user activity. While many have conducted studies on the deanonymisation of a currency's public transaction ledger, less has been done to investigate the electronic wallets that users download to hold their coins. Since an electronic wallet is for many the prominent access point into the cryptocurrency's transaction network, a user's electronic cryptocurrency wallet should be an ample store for user-generated data.

Figure 1 (see Appendix) is a pictorial summary of how a cryptocurrency transaction is performed. First, a user installs a wallet onto his computer or mobile device and, either through a third-party exchange or a donation from another user, he accumulates a sum of coins (1). To send some of these coins to another user, he goes into his wallet application and submits a request to

transfer a sum of coins to the next user (2). The payment information is gathered into a block and the block is broadcast to the entire user network for verification (3). If the verification is successful, the new block is added to the block chain, which is a public ledger of all past transactions in the network (4). Finally, the transferred coins are delivered to the new owner's wallet and the transaction is complete (5) ("Virtual Currency," 2014).

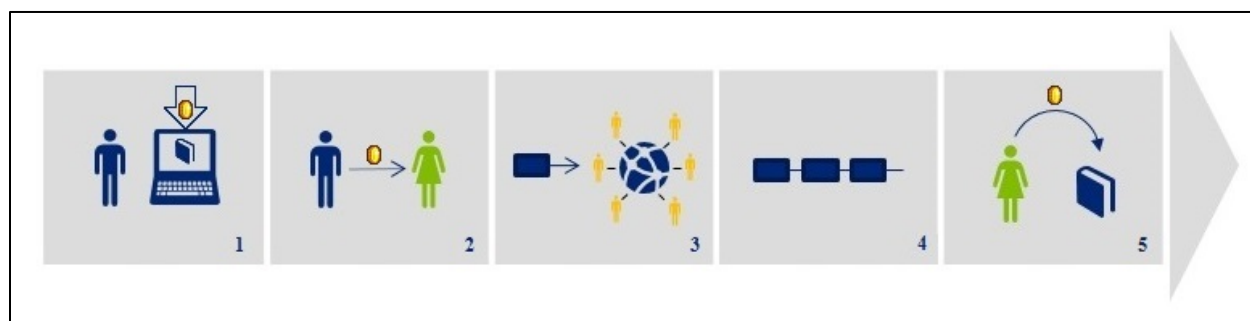


Figure 1. A Basic Cryptocurrency Transaction

Cryptocurrencies were initially created to blend the features of physical money into existing forms of electronic payments. The use of cryptography in these payments is what ensures protection from theft or fraud. Like an exchange of physical money, a cryptocurrency transaction does not explicitly identify the parties involved (Meiklejohn, Pomarole, Jordan, Levchenko, McCoy, Voelker, and Savage, 2013). Unlike cash, however, cryptocurrency transactions also work like an electronic payment in that an outside third-party intermediary is required to safeguard honesty from both sides during the transfer. While cryptocurrencies do require mediation, they are unique in that the responsibility of validating transactions rests with the entire user network instead of an outside financial institution. So while a real-world identity is never tied to a transaction or an address, every transaction that occurs is visible to every user in the network. It is presumably the misinterpretation of these pseudo-anonymous transactions as a truly anonymous process that sparks criminal interest in these currencies.

Although used currently by only a small portion of the national and world populations, cryptocurrencies are growing in prevalence and thus must be researched and understood. In this particular research, the cryptocurrency systems of interest are Bitcoin, Litecoin, and Darkcoin. Although each currency has its unique features, which are discussed in detail in Section II, their general protocol is the same given that the latter two are based upon the open-source code of Bitcoin. The theory driving this research is that, as a digital system, a cryptographic wallet will leave artifacts related to its presence and activity in the memory of a mobile device that can be found after forensic investigation. The work here will essentially be a discovery procedure to investigate the potential wealth of user information generated by the existence of cryptocurrency wallet software on iOS and Android mobile devices.

The methodology for this research involves the trading of the three cryptocurrencies Bitcoin, Litecoin, and Darkcoin. These cryptocurrencies were researched using a physical Apple iOS mobile device, a physical Samsung Galaxy S4 Android device, and an emulated Android mobile device in a controlled lab setting. As is described more fully in Section III, wallet application data was forensically extracted from the devices at four different stages in the testing process, throughout which the presence and use of the wallet applications on the devices changed. Upon completion of trading and imaging, the collected data of the mobile devices was analyzed with a variety of forensic software for information that could potentially link a cryptocurrency wallet or transaction to a specific device or real identity. The results of these analyses are detailed in Section IV. Section V provides a conclusive overview of this research, while Section VI offers suggestions of future research prospects and ways in which digital forensic examiners can use the results of this research in the investigation of cases involving cryptocurrency use on mobile devices.

II. Background

Provided here is an in-depth overview of the unique features that distinguish Bitcoin, Litecoin, and Darkcoin cryptocurrencies. Given that Litecoin and Darkcoin grew out of the operational skeleton of the Bitcoin protocol, Section II.A describes the underlying causes for the creation of a cryptocurrency system, as well as how “gold standard” Bitcoin functions, while Sections II.B and II.C detail why Litecoin and Darkcoin were subsequently created from the Bitcoin protocol and how they are unique. Describing these differences is important in further understanding how each of these currencies and their components can be used and potentially manipulated. Since it is possible that forensic analysis may also vary with each type of currency, it is crucial for investigators to be familiar with the distinct features of these digital systems.

II. A. Bitcoin Features

Bitcoin is a virtual currency that makes use of cryptography in the creation and management of its digital currency system. Created by the pseudonymous Satoshi Nakamoto and launched in early 2009, the main drive behind the creation of Bitcoin was to replace existing virtual commerce methods, which relied heavily on financial institutions to process electronic transactions in a trust-based model, with a system in which payments are moderated by an unaffiliated and more reliable third party: cryptographic proof (Nakamoto, 2008). To accomplish this, Bitcoin operates in a decentralized peer-to-peer network, meaning that instead of a single entity holding responsibility for payment verification, consensus of the entire Bitcoin network determines the validity of each payment (de la Porte, 2012).

There are a numbers of ways in which an individual can become involved in the Bitcoin network. On either a personal computer or a mobile device, a user can trade Bitcoins for goods

and services with other users or participating vendors; he can trade in coins for traditional currencies on a Bitcoin exchange platform; he can donate to a political party, charity, or a friend; or he can take on the role of a miner and lend his resources to the verification of other users' transactions (Luther, 2013). To begin, a user installs the open-source Bitcoin client—a Bitcoin wallet—onto his computer(s) to manage his account. There are two types of wallets available for download: a software wallet, which is installed straight onto a device, or a web wallet, which is a wallet hosted by a third party (“How to Set Up a Wallet,” 2013). Through a wallet, a user is able to receive, store, and send bitcoins.

An individual bitcoin is actually a chain of digital signatures (Nakamoto, 2008). These signatures are hash values that represent each sequential transfer of a coin from one user's public key to another. As a coin is transferred from user to user, the coin develops a unique chain of signatures that starts with the creation of the coin and has an end that changes as the coin continues to be passed along. A user's coins are held in an encrypted wallet, where his sets of private and public key pairs are also stored (Luther, 2013). When a user goes into his electronic wallet to send coins, his unique secret signing key (private key) is used to generate a SHA-256 hash composed of past and future transaction information. This first hash is the sender's digital signature. The combination of this signature with the hash of the previous transaction and the recipient address is collectively called a coin transaction. Because each transaction includes a reference to the previous one, the sequence of transactions forms a chain, which is recorded in the coin (Luther, 2013). The coin transaction is next combined with the public key of the receiver to ensure it can only be opened by the receiver. Finally, the bundle is cryptographically hashed once more into what becomes a new block (Luther, 2013). A block is synonymous for a

transaction. This process of generating a Bitcoin block is pictographically summarized in Figure 2 (red denotes sender components and green denotes receiver).

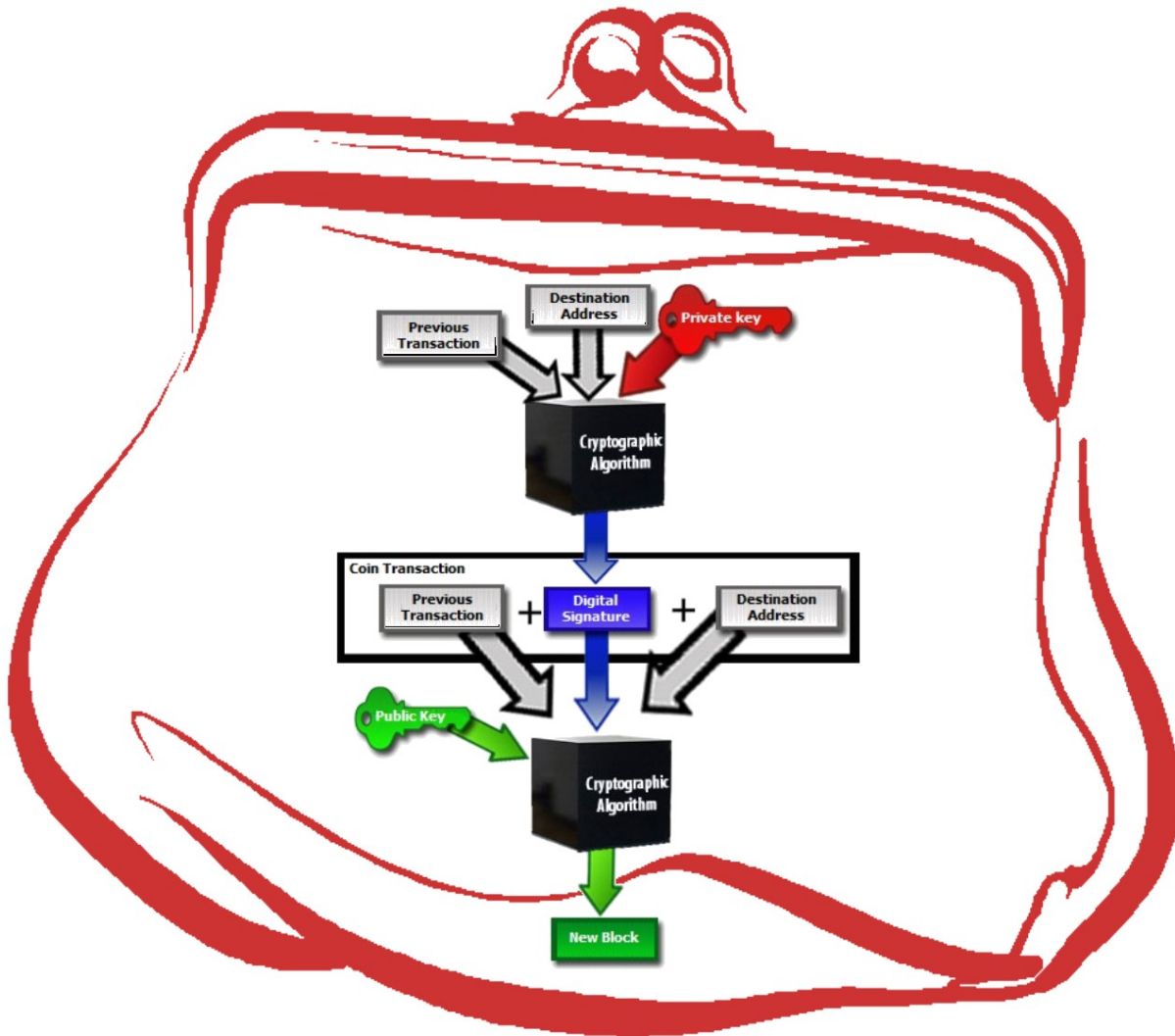


Figure 2. Cryptographic Processing of a Bitcoin Transaction

To ensure that a bitcoin is legitimate in terms of ownership, a block must be verified against the public ledger (Luther, 2013). This verification is not performed by one user, however. The information in the new block is broadcast to the entire network of users for verification (Luther, 2013). To check the validity of the block, and ultimately complete the transaction, it must be added to the official block chain—a globally visible and accepted ledger consisting of all past transactions—by a Bitcoin miner (Luther, 2013). Put simply, mining is the term used for

the process of computing complex hash calculations in order to verify Bitcoin transactions (“Mining,” 2013). During mining, the new block is compared to the historical list of transactions on the ledger. In order for the block to be confirmed, the current end of the transaction chain in the submitted coins must point to the sender as the rightful, current owner.

If ownership, and thus the block, is confirmed to be valid, the block is created and the reward—a sum of new bitcoins—for finding that block goes to the miner whose work produced the correct hash (Meiklejohn, Pomarole, Jordan, Levchenko, McCoy, Voelker, and Savage, 2013). The new block is then broadcast to the network so that the public ledger can be updated. Finally, the payment is permitted to pass on to the receiver. With her wallet, the receiver decodes the hash information using her own private key and thus ownership of the coin is successfully transferred to her (Luther, 2013). Once an entirely new payment is submitted and a newly created block references this block as the previous one in the historical chain, the block is fully accepted as part of the public block chain (Meiklejohn, Pomarole, Jordan, Levchenko, McCoy, Voelker, and Savage, 2013).

The uniformity required of a currency system mediated by a widespread network of peers depends on a sound method of communication. Though simple, the Bitcoin protocol effectively provides reliable interconnectedness and harmony to its dynamic peer-to-peer network. Peers in the Bitcoin network connect to one another over an unencrypted TCP channel. The Bitcoin protocol implements propagation and discovery mechanisms through the use of what are called messages. A message is essentially a communication flag whose name is specific intent. Nodes send and receive these messages from one another in order to spread a variety of requests through the entire network. To circulate addresses, for example, a node can request a list of neighboring peers in the network using GETADDR messages, as well as convey its own list of

known addresses using the ADDR message. Collectively, these messages are used for peer discovery, connecting to peers, block broadcasting, and transaction broadcasting. Of particular interest to this research is the TX message, which is used to describe transactions. There are multiple steps from the start to finish of a transaction broadcast. First, the sender submits an INV (inventory) message with the transaction hash. If the hash passes the validity checks on the receiver's end, the receiver sends back a GETDATA message. The sender then transmits the real transaction in a TX message. When the transaction reaches the receiver, he broadcasts an INV message to his peers so that the block chain can be updated (Birukov, Khovratovich, and Pustogarov, 2014; "Bitcoin Developer Guide," 2009).

II. B. Litecoin Features

Marketed as the "silver to Bitcoin's gold" cryptocurrency status, Litecoin protocol predominantly mirrors that of Bitcoin. While accepting and even promoting its second place status since its debut in October 2011, Litecoin's creator, Charles Lee, did create Litecoin to have features which he believed to be better than those offered by Bitcoin. His aspiration for Litecoin was to be a faster and more energy efficient cryptocurrency. To distinctly differentiate Litecoin from Bitcoin, three parameters were altered: the number of coins to introduce into circulation, the rate of block generation, and the hashing algorithm utilized (Sprankel, 2013).

Instead of a cap of 21 million coins like Bitcoin, the limited supply of Litecoin sits at 84 million coins, which is four times the number of coins than the Bitcoin network will issue (Sprankel, 2013). By having more coins available for circulation, smaller divisions of coins can theoretically be made and thus smaller transaction values become more feasible. This is

particularly attractive for merchants that operate on many small transactions being made at a high rate (“Main Page,” 2011).

The quicker transaction time offered by Litecoin is another feature that aids in the practicality of small value trades. In just two and a half minutes, as opposed to ten minutes with Bitcoin, a transaction will be put out to the network, undergo verification, and complete the transfer from sender to recipient. A faster confirmation time means less anxiety for both trading parties and also a much smaller window for potential hackers to interrupt the transaction chain (“Main Page,” 2011).

Finally, Litecoin makes itself available and useable to a broader scope of users with its mining capabilities. By using the hashing algorithm called scrypt to provide the cryptographic security and complexity to mining efforts, coins can be generated and transactions verified by standard computers with consumer-grade hardware which most individuals already own (“Main Page,” 2011). In addition to its capacity for attracting more miners, Litecoin’s use of scrypt means that this mining system does not interfere with Bitcoin’s system. If a capable user so desires, he may put his machines to mine both cryptocurrencies simultaneously. In this way, Litecoin aligns itself more as a currency complimentary to Bitcoin than as a true competitor (“Litecoin,” 2014).

II. C. Darkcoin Features

Released in March 2013, while Darkcoin is also an offshoot of Bitcoin, it is radically different than Bitcoin and any other cryptocurrency in its privacy design. Formulated with the purpose of providing true anonymity to transactions, Darkcoin eliminates the capacity to even link a transaction with the correct corresponding source or destination addresses. As creator Evan

Duffield writes in the white paper of Darkcoin, his currency can solve the blockchain's inherent inability for true privacy with the implementation of DarkSend (Duffield and Hagan, 2014).

The basic Darkcoin system operates just as Bitcoin does; however, the verification method of Darkcoin transactions is unique with the availability of DarkSend. It is an optional implementation available to Darkcoin users and can be deactivated at any time per the user's wishes (Duffield and Hagan, 2014). The innovation of DarkSend lies in its ability to mix the inputs and outputs of transactions traveling through the DarkSend payment system ("What is Darkcoin?", 2014). With DarkSend enabled, a payment request is automatically split into smaller coin denominations and then combined with other split transactions of similar size into pools of larger transactions. A master node for each pool is then randomly elected to serve as the transaction mediator for that session. This master node is a user who has volunteered their resources and 1000 Darkcoins for the privilege of assuming the role. The steep requirement aims to keep the master nodes honest and a reward of 20% of the newly mined coins provides incentive for miners to continue offering their services ("DarkSend," 2014).

Once the master node is established, the mixing begins. Since all inputs in the pool are equal in size, all outputs are virtually the same and can therefore be shuffled around interchangeably. A scheme involving blind signatures is utilized, though, to ensure the outputs of the pool will belong only to participants in that pool. This tactic additionally ensures that no user, not even the master node, knows which outputs are linked to which inputs. Once the obfuscation process is complete, the master node presents the finalized transactions to the participants. If each participant confirms their respective inputs and outputs amounts as correct, the transactions are signed and the master node broadcasts the new blocks to the network, then resigns its

position. Anyone who views the block chain thereafter will see the completed payments but will not be able to match senders to receivers (Duffield and Hagan, 2014).

Other unique features of Darkcoin include the use of a highly secure hashing algorithm X11, a total transaction time of 2.5 minutes, and a maximum supply of approximately 22 million coins (“What is Darkcoin?”, 2014). In totality, Darkcoin is the first—and currently only—cryptocurrency to boast of privacy-centered operations since the creation of Satoshi Nakamoto’s Bitcoin (Duffield and Hagan, 2014).

III. Methodology

To investigate the potential forensic artifacts left behind in mobile device memory by cryptocurrency wallets after controlled trading, the most popular wallet applications available for Bitcoin, Litecoin, and Darkcoin were considered for iOS and Android mobile devices. The wallets were tested in the most basic, user-only condition, so blockchains were never downloaded, addresses were not saved, coins were not publicly requested, only one address was used per wallet, and the wallets were not backed up.

The basic investigative premise was to extract the application data of each wallet from the two types of mobile devices at four different points in the installation and trading processes: (1) after a factory reset of the phone and prior to the installation of the wallet applications, (2) after wallet installation and before trading, (3) after completion of controlled trading of the currencies, and (4) after the wallet applications had been deleted from the device. Comparison of the data from stages 1 and 2 allow wallet-specific data to be distinguished from other software and application data native to the mobile device. Comparison of 2 and 3 permits the distinction of any transaction-specific alterations to the devices’ memory. Comparing data from stage 3 to 4

will reveal if any indicators of the wallet applications' presence on the device still exist after deletion of the wallet applications.

All extracted data was saved onto a Seagate Barracuda 1 (one) terabyte hard drive that had previously been forensically wiped and verified. A DiskCypher SATA 256 with an encryption key was attached to the hard drive to protect the contents of the hard drive. Using current forensic analysis tools and software, these extractions were gleaned for any links between user-related information and the data pulled from the application software.

III. A. Physical iOS Device

To test the iOS operating system, an Apple iPhone 4 (Model A1332) running iOS version 7.1.1 was used.

III. A. 1. iOS Extraction

In each of the four stages of the experiment, two methods of data extraction from the iOS device were used. The first was Cellebrite UFED Physical Analyzer (version 3.9.8.7) software. After opening the application on a desktop computer, the mobile device was plugged into one of the USB ports using the Cellebrite Tip T-110 attached to Cable A. In the application Graphic User Interface (GUI), the Advanced Logical extraction was selected and both extraction Method 1 and 2 were performed. The extracted images were saved for later analysis.

The second method for extraction was iFunBox, an open-source app/file manager for iOS devices. Immediately after extraction in UFED Physical Analyzer for each stage, iFunBox was used to actively examine and extract application data from the still-connected iPhone. The File System folder was extracted using the iFunBox's Dump feature and saved to the encrypted hard

drive. Since iFunBox is fundamentally an active file/app managing tool, analysis of the wallet applications could only be performed for Stages 2 and 3, when the wallets were still installed on the device.

For both methods of extraction, all collected data was saved via the Dump option (if possible) or by screenshot to the encrypted hard drive.

III. A. 2. iOS Stage 1: Device Set-Up

To ensure all data found on the mobile device was exclusively from the experiment, the phone was reset to factory settings. During the set-up process after resetting the phone, the device was configured to connect to the lab's private wireless network and a test email address was saved for future use in the iTunes App Store. Once the set-up process was complete, the phone was imaged with UFED Physical Analyzer and examined iFunBox.

III. A. 3. iOS Stage 2: Installation of Wallet Applications

Opening the iTunes App Store on the home screen of the device, the following Bitcoin wallet applications were installed onto the device: bitWallet (v. 1.5) by Sollico Software and Coin Pocket (v. 1.1.0) by Enriquez Software LLC. Litecoin- and Darkcoin-specific wallets could not be tested on the iOS in this project due to the ban Apple had placed on apps involving the transmission of virtual currencies in early 2014. This ban was lifted the first week of June 2014, less than a month before the beginning of this research, and so the availability of wallets for any currency that allowed the transference of coins and not simple fund-monitoring was scarce.

Both wallets were installed and operated as user-only; therefore, the block chains were never downloaded to the devices and no mining was performed. Once the wallets were successfully installed, each application was tapped to start. Having been opened once, the mobile device was then imaged by UFED Physical Analyzer and examined in iFunBox.

III. A. 4. iOS Stage 3: Trading

Trading occurred over a private wireless Internet connection. Trades were made with both a known user running desktop wallet applications and the physical Android device maintained in this experiment. A log in Microsoft Word was kept to record important transaction information such as sender/recipient addresses, transferred amount, the transaction hash, and the date and time stamps. After each wallet had a minimum of two received and two sent transactions, the device underwent extraction and examination.

III. A. 5. iOS Stage 4: Wallet Application Deletion

After each wallet application completed two send and two receive requests, the wallets were emptied in a final transaction to send the coins back to their owner. Once successfully emptied, the applications were deleted by pressing down on the icon until all of the app icons begin to wiggle and an encircled “X” appeared in the top right corner of each icon. The “X” for both the bitWallet and Coin Pocket wallet applications were selected and confirmed in the pop-up window for uninstallation. Once uninstalled, the mobile device data was extracted with UFED Physical Analyzer and examined in iFunBox.

III. A. 6. iOS Analysis

To analyze the collected memory data from the iOS, UFED Physical Analyzer was used to examine the extracted .ufd files. From the encrypted hard drive, the files were opened in UFED Physical Analyzer to decode and parse the collected data. Once parsed, all extracted information listed in the left project tree pane in the GUI were examined for potential forensic artifacts. In particular, the Analyzed Data, the Data Files, Timeline, and File Systems tree items and all sub-items under each were assessed. Any extracted databases were examined both in the data view in Physical Analyzer and in the open-source tool DB Browser for SQLite. All relevant data found was dumped as a file onto the Seagate hard drive and/or was documented with a screen capture.

As described previously, iFunBox is more of an active examination tool. For each stage, all items listed in the left project tree pane in the GUI were examined for potential forensic artifacts while the phone was connected to the computer. As for extracted data, the File System folder from all four stages along with the specific wallet application files from Stages 2 and 3 were later analyzed using Notepad++ and DB Browser for SQLite.

III. B. Physical Android Device

To test the Android operating system, a Samsung Galaxy S4 (Model GT-I9500) running Android OS version 4.4.2 was used. No SIM card was inserted.

III. B. 1. Physical Android Extraction

In each of the four stages of the experiment, data extraction from the Android device was performed using a Cellebrite UFED Touch Ultimate hardware device. The mobile device was connected to the UFED unit using Cable A with Tip T-100. Once plugged in, the UFED

automatically detected the device and a prompt window appeared listing extraction options for CDMA and GSM models of the Samsung GT-I9500 Galaxy S4 device. Ultimately, all possible extraction types targeting all possible content types for both CDMA and GSM versions of the phone were performed to ensure all potential data was collected. For the CDMA model, the only extraction option was a Logical Extraction. For the GSM version, extraction options included a Logical Extraction, a File System Extraction with “Android Backup – NO Shared,” and a File System Extraction with “Android Backup-with Shared.” The options of “no shared” and “with shared” refers to the exclusion or inclusion of existing shared preferences for applications in the backup.

Since the mobile device was running above Android version 4.2, the settings on the device had to be changed to activate Developer Options, allow Android/USB Debugging, and configure the screen to “Stay Awake” while on. The instructions for making these changes were provided by the UFED device before the extraction began. They can also be found on the Android Developers website.

All extractions were initially saved to a removable USB flash drive inserted into the UFED target port. Once all extractions for a stage were completed, the USB was plugged into the desktop computer and the extracted folders were cut from their place on the flash drive and pasted onto the encrypted Seagate hard drive for later analysis in UFED Physical Analyzer.

III. B. 2. Physical Android Stage 1: Device Set-Up

To ensure all data found on the mobile device was exclusively from the experiment, the phone was reset to factory settings. During the set-up process after resetting the phone, the device was configured to connect to the lab’s private wireless network and a test email address

was saved for future use of the Google Play Store. Once the set-up process was complete, the phone was imaged with UFED Touch.

III. B. 3. Physical Android Stage 2: Installation of Wallet Applications

From the Google Play Store icon on the home screen, the following wallet applications were installed onto the device: Bitcoin Wallet (version 3.53) by Andreas Schildbach, Hive Bitcoin Wallet (v. 0.3.3.3.46) by Hive Labs, Litecoin Wallet (v. 3.30.9) by Litecoin Project, and Darkcoin Wallet (beta) (v. 1.0.1) by Hash Engineering Solutions. Each wallet was installed and operated as user-only; therefore, the block chains were never downloaded to the devices and no mining was performed. Once the wallets were successfully installed, each was tapped to start. Having been opened once, the mobile device was again imaged using the UFED Touch.

III. B. 4. Physical Android Stage 3: Trading

Trading occurred over a private wireless Internet connection. Trades were made with both a known user running desktop wallet applications and the physical iOS device maintained in this experiment. A log in Microsoft Word was kept to record important transaction information such as sender/recipient addresses, transferred amount, the transaction hash, and the date and time stamps. After each wallet had a minimum of two received and two sent transactions, the device underwent extraction.

III. B. 5. Physical Android Stage 4: Wallet Application Deletion

After each wallet application completed two send and two receive requests, the wallets were emptied in a final transaction to send the coins back to their owner. Once successfully

emptied, the applications were deleted using Android's built-in Application Manager. To access it, the Settings app was opened, the More tab was tapped, and Application Manager was selected. In Application Manager was a list of the currently installed apps. To uninstall the wallet applications, the icon of each was tapped and then the Uninstall button at the top of the subsequent screen was selected. Once all four wallet applications were uninstalled, the mobile device data was extracted with the UFED Touch.

III. B. 6. Physical Android Analysis

To analyze the collected memory data from the Android device, UFED Physical Analyzer was used to examine the extracted .ufd files. From the encrypted hard drive, the files were opened in UFED Physical Analyzer to decode and parse the collected data. Once parsed, all extracted information listed in the left project tree pane in the GUI were examined for potential forensic artifacts. In particular, the Analyzed Data, the Data Files, Timeline, and File Systems tree items, as well as all sub-items under each. Any extracted databases were examined in both the Data view pane in Physical Analyzer and in the open-source tool DB Browser for SQLite. All relevant data found was dumped as a file onto the Seagate hard drive and/or was documented with a screen capture.

III. C. Virtual Android Device

To run the second method of extraction for the Android operating system, a physical mobile device was no longer available and so a virtual Android device was used. The open-source tools Genymotion (v. 2.2.2) and Oracle VM VirtualBox (v. 4.3.14) were downloaded to create this virtual device (emulator). Mirroring the device used in the first extraction process, two

virtual Samsung Galaxy S4 devices running Android OS version 4.4.2 were created for this portion of the experiment.

III. C. 1. Virtual Android Extraction

A component of the Genymotion Android Emulator that proved useful for extracting application data in this project was the Genymotion Android tool package. Automatically downloaded and available with the installation of Genymotion, these command-line tools are similar to those used for Android SDK, which are a set of development and debugging tools. The specific tool used in this project was the Android Debugging Bridge (ADB) pull command.

In order to use ADB pull on the virtual devices, the emulators had to be open and running on the computer. Additionally, the unique Google ID for each wallet was needed in order to direct the tool to pull the data from a specific wallet application. This ID was found by searching the wallet application name on the Google Play Store website and then seen in the URL of the application details page, as shown in Figure 3.



Figure 3. Location of the Google ID of an application (red) within its Google Play Store URL.

To access the ADB tools, a command prompt was opened and navigated to the C:\Program Files\Genymobile\Genymotion\tools directory. This directory is the default location selected for storing the Genymotion tools during the installation process. The data unique to each wallet application was then pulled from the emulator and directed to a designated output directory on the local computer using the command syntax displayed in Figure 4. A successful ADB pull was confirmed by comparing the number and names of the files seen in the specified output directory open in a Windows Explorer GUI to what was printed in the command line window output (Figure 5).

A screenshot of a Windows Command Prompt window titled "Administrator: Command Prompt". The window has a black background with white text. The command entered is: `C:\Program Files\Genymobile\Genymotion\tools>adb pull /data/data/de.schildbach.wallet C:\Users\AJ\Documents\Research2\Android-ADB-Pulls\1stDay-Pull\Bitcoin-Wallet`. The command is split across three lines: the first line is the prompt and the command up to the path, the second line is the path, and the third line is the final part of the path. The window also shows standard Windows window controls (minimize, maximize, close) in the top right corner.

```
C:\Program Files\Genymobile\Genymotion\tools>adb pull /data/data/de.schildbach.w
allet C:\Users\AJ\Documents\Research2\Android-ADB-Pulls\1stDay-Pull\Bitcoin-Wall
et
```

Figure 4. ADB pull command syntax to extract Android Bitcoin Wallet application data.

```

Administrator: Command Prompt
C:\Program Files\Genymobile\Genymotion\tools>adb pull /data/data/de.schildbach.w
allet C:\Users\AJ\Documents\Research2\Android-ADB-Pulls\1stDay-Pull\Bitcoin-Wall
et
pull: building file list...
skipping special file 'lib'
pull: /data/data/de.schildbach.wallet/app_log/wallet.log -> C:\Users\AJ\Documen
s\Research2\Android-ADB-Pulls\1stDay-Pull\Bitcoin-Wallet/app_log/wallet.log
pull: /data/data/de.schildbach.wallet/files/key-backup-protobuf.93 -> C:\Users\AJ
\Documents\Research2\Android-ADB-Pulls\1stDay-Pull\Bitcoin-Wallet/files/key-bac
kup-protobuf.93
pull: /data/data/de.schildbach.wallet/files/wallet-protobuf -> C:\Users\AJ\Docum
ents\Research2\Android-ADB-Pulls\1stDay-Pull\Bitcoin-Wallet/files/wallet-protobu
f
pull: /data/data/de.schildbach.wallet/files/key-backup-protobuf -> C:\Users\AJ\D
ocuments\Research2\Android-ADB-Pulls\1stDay-Pull\Bitcoin-Wallet/files/key-backup
-protobuf
pull: /data/data/de.schildbach.wallet/shared_prefs/de.schildbach.wallet_preferen
ces.xml -> C:\Users\AJ\Documents\Research2\Android-ADB-Pulls\1stDay-Pull\Bitcoin
-Wallet/shared_prefs/de.schildbach.wallet_preferences.xml
pull: /data/data/de.schildbach.wallet/app_blockstore/blockchain -> C:\Users\AJ\D
ocuments\Research2\Android-ADB-Pulls\1stDay-Pull\Bitcoin-Wallet/app_blockstore/b
lockchain
pull: /data/data/de.schildbach.wallet/databases/address_book-journal -> C:\Users
\AJ\Documents\Research2\Android-ADB-Pulls\1stDay-Pull\Bitcoin-Wallet/databases/a
ddress_book-journal
pull: /data/data/de.schildbach.wallet/databases/address_book -> C:\Users\AJ\Docu
ments\Research2\Android-ADB-Pulls\1stDay-Pull\Bitcoin-Wallet/databases/address_b
ook
8 files pulled. 0 files skipped.
1643 KB/s (862919 bytes in 0.512s)
C:\Program Files\Genymobile\Genymotion\tools>_

```

Figure 5. Command prompt output of a successful ADB pull of the Bitcoin Wallet application data.

Since application-specific data could only be pulled from an installed application, however, no wallet application data was extracted in Stage 1. All other application data present on the emulator, however, was pulled after each of the four stages and saved to the encrypted hard drive.

III. C. 2. Virtual Android Stage 1: Device Set-Up

After successful installation of Genymotion and Oracle VirtualBox, Genymotion was started. Two new virtual devices were created with the specifications listed in Table 1.

Specification	Setting
Android version	4.2.2
Device model	Samsung Galaxy S4

Memory size	2048 MB
Data storage capacity	16,394 MB

Table 1. Genymotion Android Emulator Set-Up Specifications

Once the emulators were successfully created, they were ready to use. After powering on an emulator, it was necessary to check that wireless capabilities were enabled in order to successfully download the wallet applications and make trades. If the gray wireless symbol was visible in the top right corner of the emulation window, it was confirmed that the emulator was properly set up. To confirm connectivity in a second way, the browser application was opened and a Google search was made. If the search completed and displayed results, it was determined that the device was properly configured.

As with the physical Android device, the emulator required activation of Developer Options so that USB Debugging could be enabled to successfully extract application data. Navigation through the settings to make this change was the same as with the physical device and is available for reference on the Android Developers website

III. C. 3. Virtual Android Stage 2: Installation of Wallet Applications

Before the wallet applications could be installed, a Security setting in the emulator had to be changed. Clicking on the Settings icon within the emulator, the Security option was selected from the Options menu. Under Device Administration, the box next to “Unknown Sources” was checked to allow installation of apps from unknown sources onto the device.

To install the applications onto the emulators, a third-party website was used to download the wallets’ Android application package, or APK file, which contained the application software. Clicking on the Browser icon on the emulator home screen, the following URL was typed into the address bar: apps.evozi.com/apk-downloader. The Google ID of a wallet (obtained in Section III. C. 1.) was typed into the input box on the page and then the “Generate Download Link” button was clicked. After the package details appeared for the designated Google ID, the “Click here to Download” button was clicked. This process is illustrated in Figure 6. Once the application .apk file was successfully downloaded, the application was installed onto the device from the Downloads menu.

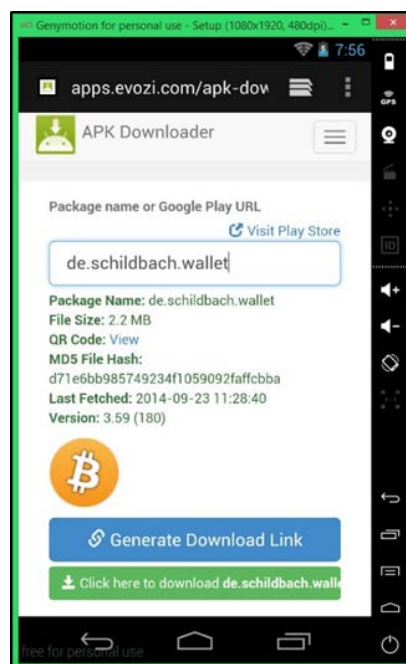


Figure 6. APK Downloader: Details of the Bitcoin Wallet APK File.

After all four wallet applications were downloaded and installed onto the device, the applications were clicked open since it was discovered that the ADB pull command could not extract any wallet application data from the emulator if the application had never been opened. With the emulator open and running, the first ADB pull extraction of each wallet application was made and saved onto the Seagate hard drive.

III. C. 4. Virtual Android Stage 3: Trading

Trading occurred over a private wireless Internet connection and was made between the two Samsung Galaxy S4 emulations running simultaneously. A log in Microsoft Word was kept to record important transaction information such as sender/recipient addresses, transferred amount, the transaction hash, and the date and time stamps. After each wallet had a minimum of two received and two sent transactions, the device underwent extraction.

III. C. 5. Virtual Android Stage 4: Wallet Application Deletion

After each wallet application completed two send and two receive requests, the wallets were emptied in a final transaction to send the coins back to their owner. Once successfully emptied, the applications were deleted in the same way that the apps were deleted on the physical Android device: using Android's built-in Application Manager. To access it, the Settings app was opened, the More tab was tapped, and Application Manager was selected. In Application Manager was a list of the currently installed apps. To uninstall the wallet applications, the icon of each was tapped and then the Uninstall button at the top of the subsequent screen was selected. Once all four wallet applications were uninstalled, all application data was extracted with ADB pull.

III. C. 6. Virtual Android Analysis

To analyze the collected memory data from the ADB pull extractions, Notepad++, an open-source code editor, and DB Browser for SQLite were used. All files extracted from the ADB pull were opened in Notepad++ to review and keyword search if the file was human-readable. All extracted database files were opened in DB Browser for SQLite to assess the tables for entries relating to the wallet applications. If relevant data was found, the database was subsequently queried for specific wallet application data. All relevant data found in any program was saved onto the Seagate hard drive and/or was documented with a screen capture.

IV. Data and Results

With both the physical iOS and Android mobile devices, after loading the UFED extraction files in Physical Analyzer, all extracted components visible in the Project Tree were analyzed. In particular, the Analyzed Data, the Data Files, Timeline, and File Systems tree items and all sub-items under each. In addition to examination in UFED Physical Analyzer, select database files were also analyzed in DB Browser for SQLite.

For the physical iOS device, active analysis using iFunBox was performed while the phone was still attached to the computer for each stage. In these examinations, information listed in the left project tree pane in the GUI was examined for potential forensic artifacts. From Stages 2 and 3, the File System folder and all wallet application files dumped from iFunBox were analyzed in Notepad++.

For the virtual Android device, all files extracted with ADB pull were analyzed in either Notepad++ or DB Browser for SQLite. The data pulled from the specific wallet applications in Stages 2 and 3, along with all of the application data on the emulator in all four stages, were the

items analyzed by these methods. It was found that the content of these pulls only pertained to the activity made in the active session during which the pull was made and was erased after the emulator was shut down.

IV. A. 1. Physical iOS Results – UFED Physical Analyzer

Under the Analyzed Data tree item in Physical Analyzer, the only sub-item with any relevant artifacts was Installed Applications. In Stages 2 and 3, the Extraction table in the data display contained entries for bitWallet and CoinPocket (see Figure 7). A particularly key value in the table for each application was its Identifier. In extraction tables encountered further on in the analysis, these identifiers were found again. While these data entries were extracted in Stages 2 and 3, they did not persist after the wallet applications were deleted in Stage 4.

Installed Applications										
#	Name	Description	Version	Identifier	Application ID	Purchase Date	Copyright	Deleted Date	Permissions	Del?
2	bitWallet		1.5	com.sollico.bitwallet	E8AA0587-CB19-4E51-B662-1E0DFA8F1925					
5	CoinPocket		1.2.0	me.enriquez.CoinPocket	3BCB1663-674C-4C28-8B0D-B4E0DDF83818					

Figure 7. Installed Applications data table entries for bitWallet and CoinPocket from UFED iOS Advanced Logical Extraction.

Within the Data Files tree item, two categories were of interest: Configurations and Databases. In both Method 1 and 2 of the Advanced Logical extractions, the Configurations sub-item contained an entry with the details of a .plist file for the CoinPocket wallet, as shown in Figure 8. A plist is a “property list” file used by iOS to store a user’s settings or information about bundles and applications. These entries list the storage path of the .plist configuration files, as well as the Created, Last Modified, and Last Accessed date and time stamps, which were congruent with the true installation action of the wallet applications in the test. No configuration

files existed for bitWallet in any of the four stages. As with the Installed Application data entries, these CoinPocket entries were extracted after wallet installation, but not after deletion of the wallets.

Configurations			
#	File Info	Additional file info	Del?
108	Name: me.enriquez.CoinPocket.plist Path: /Library/Preferences/me.enriquez.CoinPocket.plist	Size (Bytes): 464 Created: 7/18/2014 3:20:21 PM(UTC+0) Modified: 7/18/2014 3:20:21 PM(UTC+0) Accessed: 7/18/2014 3:20:21 PM(UTC+0)	Method 1
108	Name: me.enriquez.CoinPocket.plist Path: /Applications/me.enriquez.CoinPocket/Library/Preferences/me.enriquez.CoinPocket.plist	Size (Bytes): 464 Modified: 7/18/2014 3:20:21 PM(UTC+0)	Method 2

Figure 8. Device configuration files for CoinPocket from UFED iOS Advanced Logical Extractions.

Under the Databases category, two databases with the CoinPocket identifier were extracted after installation of the wallets (Figure 9). These database files were only seen in the Method 2 extraction, however. The first is a cache database and the second is a local storage file. After inspection in UFED and DB Browser for SQLite, neither database file was found to contain relevant data. The Last Modified date and time stamps seen in the entries were updated from Stage 2 to 3. Again, no database files for bitWallet were found and these database files for CoinPocket were removed after deletion of the wallet applications from the mobile device.

Databases			
#	File Info	Additional file info	Del?
25	Name: Cache.db Path: /Applications/me.enriquez.CoinPocket/Library/Caches/me.enriquez.CoinPocket/Cache.db	Size (Bytes): 4096 Modified: 7/23/2014 7:02:07 PM(UTC+0)	
42	Name: file_0.localstorage Path: /Applications/me.enriquez.CoinPocket/Library/Caches/file_0.localstorage	Size (Bytes): 12288 Modified: 7/18/2014 3:25:20 PM(UTC+0)	

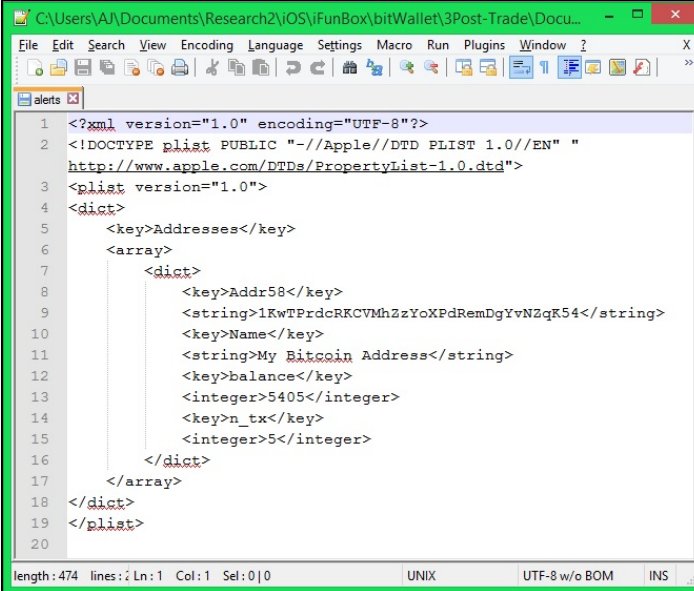
Figure 9. Database files for CoinPocket from Method 2 UFED iOS Advanced Logical Extraction.

Physical Analyzer was not able to extract any data for the Time Line tree item in any of the iOS .ufd extraction files. Additionally, analysis of the File Systems tree item revealed no further artifacts of forensic relevance.

IV. A. 2. Physical iOS Results – iFunBox

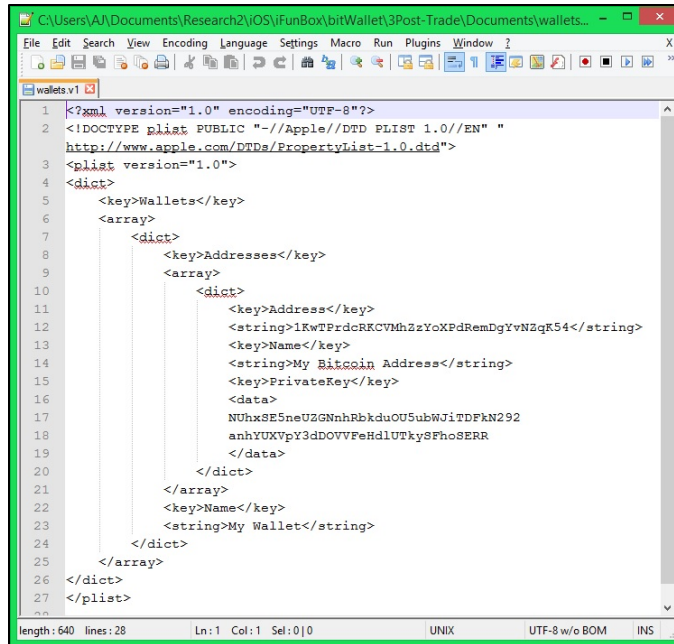
While active examination of the iOS mobile device after each stage in iFunBox yielded no relevant data, analysis of the extracted File System and wallet-specific files in Notepad++ did turn up some pertinent data.

In the wallet application dump of bitWallet, five folders were extracted—bitWallet.app, Documents, Library, StoreKit, and tmp. In the Documents folder was an alerts.file file. When opened in Notepad++, this file was found to contain the public address string of the wallet installed on the device (see Figure 10). In the same folder was another file: wallets.v1. This file listed not only the public address (key) of the wallet, but the private key as well (see Figure 11). The other folders and files in the extractions of bitWallet did not hold relevant forensic data.



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "
3 http://www.apple.com/DTDs/PropertyList-1.0.dtd">
4 <plist version="1.0">
5 <dict>
6 <key>Addresses</key>
7 <array>
8 <dict>
9 <key>Addr58</key>
10 <string>1KwTPrdcRKCVMhZzYoXPdRemDgYvNZqK54</string>
11 <key>Name</key>
12 <string>My Bitcoin Address</string>
13 <key>balance</key>
14 <integer>5405</integer>
15 <key>n_tx</key>
16 <integer>5</integer>
17 </dict>
18 </array>
19 </dict>
20 </plist>
```

Figure 10. bitWallet: Content of the Alerts.file File Extracted by iFunBox.



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "
3 http://www.apple.com/DTDs/PropertyList-1.0.dtd">
4 <plist version="1.0">
5 <dict>
6 <key>Wallets</key>
7 <array>
8 <dict>
9 <key>Addresses</key>
10 <array>
11 <dict>
12 <key>Address</key>
13 <string>1KwTPrdcRKCVMhZzYoXPdRemDgYvN2qK54</string>
14 <key>Name</key>
15 <string>My Bitcoin Address</string>
16 <key>PrivateKey</key>
17 <data>
18 NUhxSE5neUZGNhhRbkduOU5SubWjiTDFkN292
19 anhYUXVpY3dDOVVFehdlUTkysFhoSERR
20 </data>
21 </dict>
22 </array>
23 <key>Name</key>
24 <string>My Wallet</string>
25 </dict>
26 </array>
27 </dict>
28 </plist>
```

Figure 11. bitWallet: Content of the Wallet.v1 File Extracted by iFunBox.

In the wallet application dump of CoinPocket, again five folders were extracted: CoinPocket.app, Documents, Library, StoreKit, and tmp. None of the files in in these folders held relevant forensic data.

In the Raw File System dump of the mobile device from Stage 3, two folders containing data related to the wallet applications were extracted: Purchases and Downloads. The Purchases folder contained a handful of .plist files, but none of them were related to the wallet applications. The Downloads folder contained a SQLite database file labeled downloads.28.sqlitedb. When opened in DB Browser for SQLite, the database file contained a table labeled Purchase with two entries. Within this table was encoded_data and encoded_response columns whose cells contained binary data which, when interpreted, listed the names of the ID of the wallet application downloaded, as well as the iTunes name of the individual who purchased the wallet application, respectively (see Figures 12 and 13). No relevant date and time stamps were found. These table entries were still extracted after the wallet application was deleted from the mobile device.

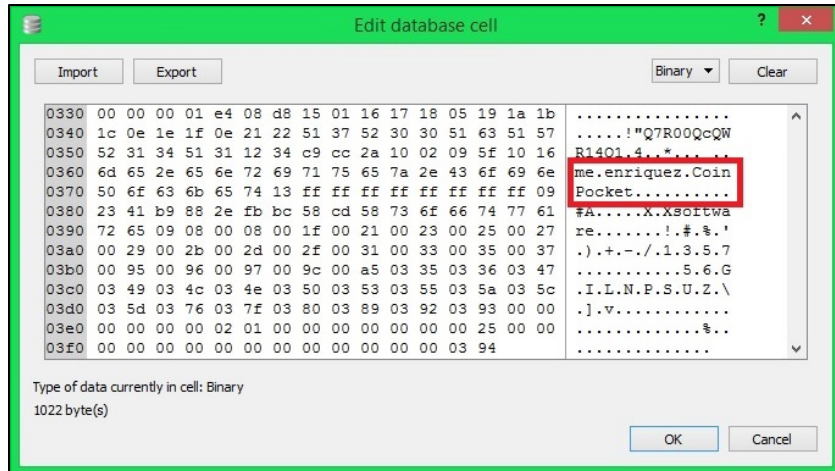


Figure 12. Wallet Application Name in the Binary Interpretation of the Downloads.28.sqlitedb Encoded_Data Column.

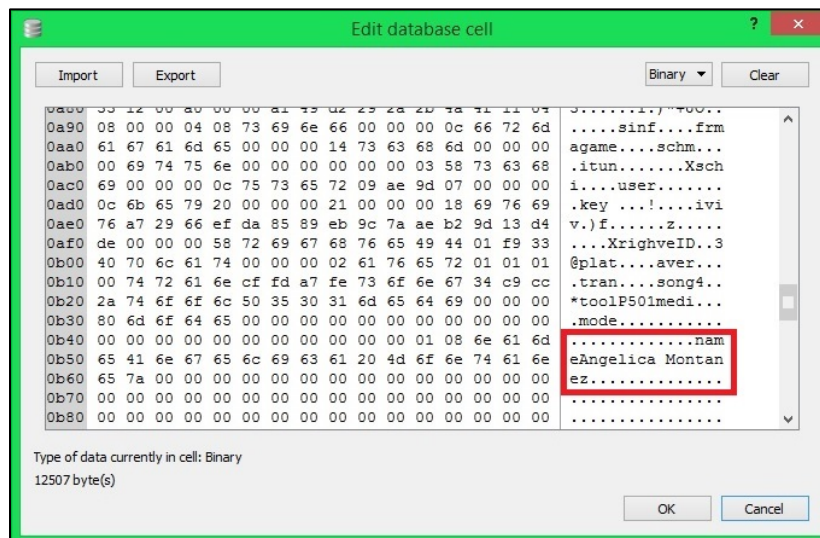


Figure 13. iTunes Name in the Binary Interpretation of the Downloads.28.sqlitedb Encoded_Response Column.

IV. B. Physical Android Device Results

The Logical extractions yielded no data when opened in UFED Physical Analyzer.

The Shared File System extraction did not contain any relevant information that was not already included in the No Shared File System extraction. Thus, all results obtained were from analysis of the No Shared File System extractions.

Under the Analyzed Data tree item in Physical Analyzer, the only sub-item with any relevant artifacts was Installed Applications. After installation of the wallets, the extraction table in the data display contained entries for the Bitcoin Wallet, Hive Wallet, Litecoin Wallet, and Darkcoin Wallet applications (Figure 14). An especially key value in the table for each application was its Identifier. As the analysis process progressed, these identifiers were found in other extracted tables. Another column in this table containing pertinent data was the Purchase Date. The date and time stamps in these cells correspond to the actual installation date and time of the wallets onto the mobile device. These data entries for the wallets remained present in the table after trading in Stage 3 and were still extracted after deletion of the wallet applications in Stage 4.

Installed Applications										
#	Name	Description	Version	Identifier	Application ID	Purchase Date	Copyright	Deleted Date	Permissions	Del?
13	Bitcoin Wallet			de.schildbach.wallet		6/27/2014 21:21:57(UTC+0)				
16	Darkcoin Wallet (beta)			hashengineering.darkcoin.wallet		6/27/2014 21:26:23(UTC+0)				
30	Hive Bitcoin Wallet			com.hivewallet.androidclient.wallet		6/27/2014 21:22:54(UTC+0)				
31	Litecoin Wallet			de.schildbach.wallet_ltc		6/27/2014 21:23:57(UTC+0)				

Figure 14. Installed Applications data table entries from the UFED Touch Extraction of the Android device.

Within the Data Files tree item, the Databases category was the only one found to contain data relating to the wallet applications after their installation. The launcher.db file, when opened in DB Browser for SQLite, displayed two tables labeled AppOrder and Favorites. These two tables contained a column listing the wallet identifiers, as found in the Installed Applications table. The Favorites table also had a column describing the application's Intent (see Figure 15). An intent is defined by the Android Developers webpage as an abstract description of an operation to be performed. More basically, it is a component involved in the launching of

activities on the mobile device (“Intent”). For each of the four wallets, there was a Favorites table entry listing the identifier and intent for the applications when clicked to launch. These data entries for the wallets remained present in the table after trading, but were not extracted after deletion of the wallet applications.

android_metadata (1)	appOrder (56)	favorites (24)	_id	itemType	container	screen	cellX	cellY	spanX	spanY	title	intent
			21	1	-100	2	0	2	1	1	Bitcoin	#intent:action=android.intent.action.MAIN;category=android.intent.category.LAUNCHER;launchFlags=0x10000000;package=de.schildbach.wa
			22	1	-100	2	1	2	1	1	Hive	#intent:action=android.intent.action.MAIN;category=android.intent.category.LAUNCHER;launchFlags=0x10000000;package=com.hivewallet.a
			23	1	-100	2	2	2	1	1	Litecoin	#intent:action=android.intent.action.MAIN;category=android.intent.category.LAUNCHER;launchFlags=0x10000000;package=de.schildbach.wa
			24	1	-100	2	3	2	1	1	Darkcoin (beta)	#intent:action=android.intent.action.MAIN;category=android.intent.category.LAUNCHER;launchFlags=0x10000000;package=hashengineering;

Figure 15. Favorites table of Launcher Database file in Physical Analyzer from the UFED Touch Extraction of the Android device.

A second database file named localappstate.db was found in the Android extractions which contained a table labeled Appstate that held data relating to the wallet applications. When opening in DB Browser for SQLite, this table listed the wallet application identifiers, as well as the first download timestamps of the wallets, the most recent data delivery time stamp, and the user account under which the wallet applications were downloaded. Figure 16 displays these entries, which were selected by a SQL query for easier viewing. The timestamps, once converted from Unix Epoch (Unix timestamp) to the correct time zone, were congruent with the application install date and time, and the date and time when the wallet application was active on the mobile device. These table entries were extracted after both trading and deletion of the applications.

package_name	o_upd	red_ver	nloa	delivery_data	delivery_data_timestamp_ms	aller_st	first_download_ms	eferre	account	title	flags	timeue	last_notified_version	last_update_timestamp_ms	st_for_1	auto_acquire_tag
1 de.schildbach.wallet_ltc	1	-1		(BLOB)	1403904236969	0	1403904237470		khaldrogo018@gmail.com	Litecoin Wallet	0	163	0			
2 de.schildbach.wallet	1	-1		(BLOB)	1405448381688	0	1403904117971		khaldrogo018@gmail.com	Bitcoin Wallet	0	174	1405448386488			
3 hashengineering.darkcoin.wallet	1	-1		(BLOB)	1405454681207	0	1403904383725		khaldrogo018@gmail.com	Darkcoin Wallet (beta)	0	11001	1405454690398			
4 com.hivewallet.androidclient.wallet	1	-1		(BLOB)	1405696335311	0	1403904174827		khaldrogo018@gmail.com	Hive Bitcoin Wallet	0	16	1405696392792			

Figure 16. Appstate table in DB Browser of LocalAppState Database file from the UFED Touch Extraction of the Android device.

The Timeline tree item was quite informative for the Android device. After installation of the wallets, two entry types of interest were extracted (see Figure 17). The first were the

Installed Application entries—there was one for each of the four wallets, with date and time stamps that matched the actual purchase date. The second was Searched Items entries. These entries denoted terms typed into the search bar of the Google Play Store, which was, in fact, how these wallets were located in the Play Store for download onto the device. These table entries were extracted after trading and after deletion of the wallet applications.

Timeline							
#	Type	Direction	Timestamp	Party	Description	Del?	
8	Installed Applications		6/27/2014 21:21:57(UTC+0)		Bitcoin Wallet		
9	Searched Items		6/27/2014 21:22:26(UTC+0)		bitcoin wallet		
10	Installed Applications		6/27/2014 21:22:54(UTC+0)		Hive Bitcoin Wallet		
11	Searched Items		6/27/2014 21:23:34(UTC+0)		litecoin wallet		
12	Installed Applications		6/27/2014 21:23:57(UTC+0)		Litecoin Wallet		
13	Searched Items		6/27/2014 21:24:13(UTC+0)		darkcoin		
14	Installed Applications		6/27/2014 21:26:23(UTC+0)		Darkcoin Wallet (beta)		

Figure 17. Timeline data table entries for the four cryptocurrency wallets from the UFED Touch Extraction of the Android device.

The final tree item analyzed for the physical Android device was the File System. The only file found to contain relevant data not previously discovered was the finsky.xml file in the com.android.vending sub-item (see Figure 18, boxed in yellow). When selected in Physical Analyzer, the text view of the file displayed three of the four wallet application names amid a list of other device applications, which are marked in the red boxes in Figure 18. Based on the code present (orange underlines), this finsky.xml file was determined to be a record of application update notifications. It was also deduced that the wallet applications were present in this list because the default setting for the mobile device was to automatically update installed applications when connected to a WiFi network and that this update process occurred during testing. After the installation of the wallet applications, this file and its contents were visible in the File System tree item in all subsequent stage extractions.

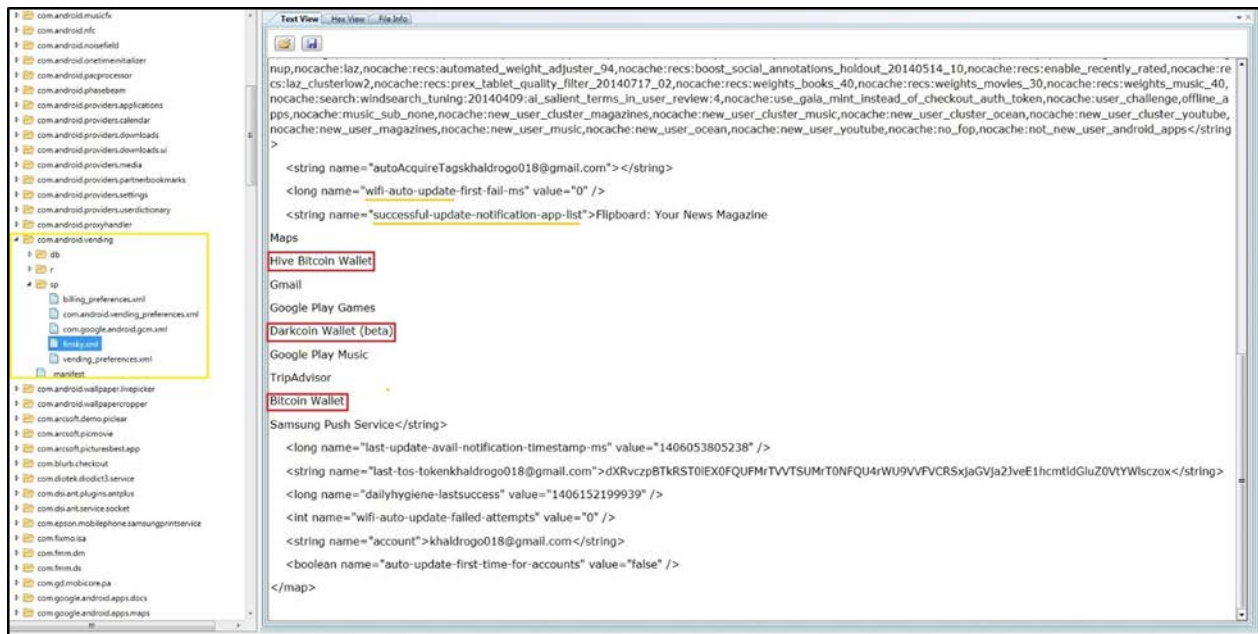


Figure 18. Finsky.xml file within the File System tree item in Physical Analyzer from the UFED Touch Extraction of the Android device.

IV. C. Virtual Android Device Results

Given that five separate extractions were necessary after each stage to collect the data specific to the four wallets, as well as the collective application data in memory, the results of each are likewise discussed separately.

IV. C. 1. Bitcoin Wallet Results

From the ADB pull of the Bitcoin Wallet application, five folders were extracted: app_blockstore, app_log, databases, files, and shared_prefs. The app_blockstore folder contained one file, blockchain.file, whose data was not human readable when opened in Notepad++. The app_log folder contained a file labeled wallet.log which was found to be a historical record of the wallet application's activity for the most recent emulator session. Of particular interest in this log file were the blocks of entries for a transaction, which involved the TX message of the Bitcoin

Protocol (see Figure 19). After comparing the activity in a log containing both a sent and received transaction, it was found that all pertinent entries relating to a transaction included the transaction hash value. Searching the transaction hash in Notepad++ brought up a Results box listing all entries containing the hash value (see Figure 20), which, after referring back to the full log file, were surrounded by the other entries describing the transaction. Searching “tx” similarly helped with narrowing the area of focus for relevant information within the extensive log file. In addition to Bitcoin network protocol messages and transaction hashes, the wallet.log file also provided a variety of IP addresses of network peers. Based on the structure of the Bitcoin network, it was concluded that these IP addresses are most likely linked to random Bitcoin nodes which were involved in passing along the information and/or in verifying this singular transaction.

```

514 037429 | 16:56:13.821 [NioClientManager] MemoryPool - [67.22.244.206]:8333: Peer announced new transaction [1
515 037529 | ] 92022193ff5061b4136387806f8fc857a782a971c1af6c59098dfe29fc13c3ff
516 037596 | 16:56:14.147 [NioClientManager] Peer - [67.22.244.206]:8333: Downloading dependencies of 92022193ff5
517 037696 | 061b4136387806f8fc857a782a971c1af6c59098dfe29fc13c3ff
518 037750 | 16:56:14.341 [NioClientManager] Peer - [67.22.244.206]:8333: Bottomed out dep tree at 828d661a446b0d
519 037850 | d65613b73840ff5f26f4c039d8e355fdbcb5b82930dfb9e4cf7
520 037901 | 16:56:14.343 [NioClientManager] Peer - [67.22.244.206]:8333: Dependency download complete!
521 037992 | 16:56:14.351 [NioClientManager] Wallet - Received a pending transaction 92022193ff5061b4136387806f8f
522 038092 | c857a782a971c1af6c59098dfe29fc13c3ff that spends 0.00 BTC from our own wallet, and sends us 0.0011 B
523 038192 | TC
524 038195 | 16:56:14.356 [NioClientManager] Wallet - commitTx of 92022193ff5061b4136387806f8fc857a782a971c1af6c5
525 038295 | 9098dfe29fc13c3ff
526 038313 | 16:56:14.360 [NioClientManager] Wallet - --pending: 92022193ff5061b4136387806f8fc857a782a971c1af6c59
527 038413 | 098dfe29fc13c3ff
528 038430 | 16:56:14.379 [NioClientManager] WalletFiles - Saving wallet, last seen block is 315100/00000000000000
529 038530 | 0002144ea796b242d064de34deb773a7d86e30e442f44705b41
530 038582 | 16:56:14.387 [NioClientManager] WalletFiles - Save completed in 7msec
531 038652 | 16:57:00.039 [main] BlockchainServiceImpl - History of transactions/blocks: 1/0
532 038732 | 16:58:00.039 [main] BlockchainServiceImpl - History of transactions/blocks: 0/0, 1/0
533 038817 | 16:59:00.041 [main] BlockchainServiceImpl - History of transactions/blocks: 0/0, 0/0, 1/0
534 038907 | 17:00:00.041 [main] BlockchainServiceImpl - History of transactions/blocks: 0/0, 0/0, 0/0, 1/0
535 039002 | 17:00:02.640 [NioClientManager] AbstractBlockChain - 8 blocks per second
536 039075 | 17:00:02.646 [NioClientManager] Wallet - Received tx for 0.0011 BTC: 92022193ff5061b4136387806f8fc85
537 039175 | 7a782a971c1af6c59098dfe29fc13c3ff [0] in block 000000000000000366d6e030f504399f4319796a4d80e9d86e3c
538 039275 | 33c2976a269
539 039287 | 17:00:02.649 [NioClientManager] Wallet - <-pending
540 039340 | 17:00:02.652 [NioClientManager] Wallet - tx 92022193ff5061b4136387806f8fc857a782a971c1af6c59098dfe
541 039440 | 29fc13c3ff -->unspent
542 039461 | 17:00:02.665 [NioClientManager] Wallet - Balance is now: 0.0011
543 039525 | 17:00:02.671 [NioClientManager] WalletFiles - Saving wallet, last seen block is 315100/00000000000000
544 039625 | 0002144ea796b242d064de34deb773a7d86e30e442f44705b41
545 039677 | 17:00:02.714 [NioClientManager] WalletFiles - Save completed in 29msec
546 039748 | 17:00:03.715 [main] BlockchainServiceImpl - service start command: Intent { act=de.schildbach.wallet
547 039848 | .service.cancel_coins_received cmp=de.schildbach.wallet/.service.BlockchainServiceImpl }

```

Figure 19. Transaction entries for the first received payment in the wallet.log file of the Bitcoin Wallet application.


```
Find result - 11 hits
C:\Users\A\Desktop\Research\Android\ADB-Extractions\Test_Emulation\Pull_Day\Bitcoin-Wallet-Pull\app_log\wallet.log (11 hits)
Line 364: 16:56:13.821 [NioClientManager] MemoryPool - [67.22.244.206]:8333: Peer announced new transaction [1] 92022193ff5061b4136387806f8fc857a782a971c1af6c59098dfe29fc13c3ff
Line 365: 16:56:14.147 [NioClientManager] Peer - [67.22.244.206]:8333: Downloading dependencies of 92022193ff5061b4136387806f8fc857a782a971c1af6c59098dfe29fc13c3ff
Line 368: 16:56:14.351 [NioClientManager] Wallet - Received a pending transaction 92022193ff5061b4136387806f8fc857a782a971c1af6c59098dfe29fc13c3ff that spends 0.00 BTC from our own wallet, and
Line 369: 16:56:14.356 [NioClientManager] Wallet - commitTx of 92022193ff5061b4136387806f8fc857a782a971c1af6c59098dfe29fc13c3ff
Line 370: 16:56:14.360 [NioClientManager] Wallet - -pending: 92022193ff5061b4136387806f8fc857a782a971c1af6c59098dfe29fc13c3ff
Line 378: 17:00:02.644 [NioClientManager] Wallet - Received tx for 0.0011 BTC: 92022193ff5061b4136387806f8fc857a782a971c1af6c59098dfe29fc13c3ff [0] in block 0000000000000036d6e030f504399f431
Line 380: 17:00:02.652 [NioClientManager] Wallet - tx 92022193ff5061b4136387806f8fc857a782a971c1af6c59098dfe29fc13c3ff ->unspent
Line 903:      outpoint:92022193ff5061b4136387806f8fc857a782a971c1af6c59098dfe29fc13c3ff:0 hash160:3a0e686cb78b029e3a33f6cbe9a2757905f9600
Line 907: 12:57:34.366 [backgroundThread] Wallet - marked 92022193ff5061b4136387806f8fc857a782a971c1af6c59098dfe29fc13c3ff:0 as spent
Line 908: 12:57:34.370 [backgroundThread] Wallet - 92022193ff5061b4136387806f8fc857a782a971c1af6c59098dfe29fc13c3ff prevtx <-unspent ->spent
Line 929: 13:01:48.550 [NioClientManager] Wallet - marked 92022193ff5061b4136387806f8fc857a782a971c1af6c59098dfe29fc13c3ff:0 as spent
```

Figure 20. Results for a search of the first received transaction hash in Notepad++ within the Bitcoin Wallet wallet.log file.

Switching back to the ADB pulled files, in the databases folder was address_book.file and address_book-journal.file, two files which were unreadable in Notepad++ and contained no data in their tables when opened in DB Browser for SQLite. Into the files folder, three files were extracted: key-backup-protobuf.file, key-backup-protobuf.p3, and wallet-protobuf.file. None of these files were readable when opened in Notepad++. Finally, in the shared_prefs folder was a single file labeled de.schildbach.wallet_preferences.xml. The data in this file was just a short code script for some of the wallet application preferences. After deletion of the Bitcoin Wallet application from the emulator, its application data folder ceased to exist and thus could not be pulled.

IV. C. 2. Hive Wallet Results

From the ADB pull of the Hive Wallet application, six folders were extracted: app_blockstore, app_log, cache, databases, files, and shared_prefs. The contents of the first two folders were the same as those found for Bitcoin Wallet. In the cache folder, two files were extracted: wallet.-267244447.log and wallet-dump.1791589373.txt. The log file, like the wallet.log file in the app_log folder, was a historical record of wallet activity. Unlike the wallet.log file which includes communication information with external sources, this log file appeared to list activity exclusively internal to the application. There were, however, entries with

similar TX information for a transaction seen in this file as was found in the wallet.log file (Figure 21).

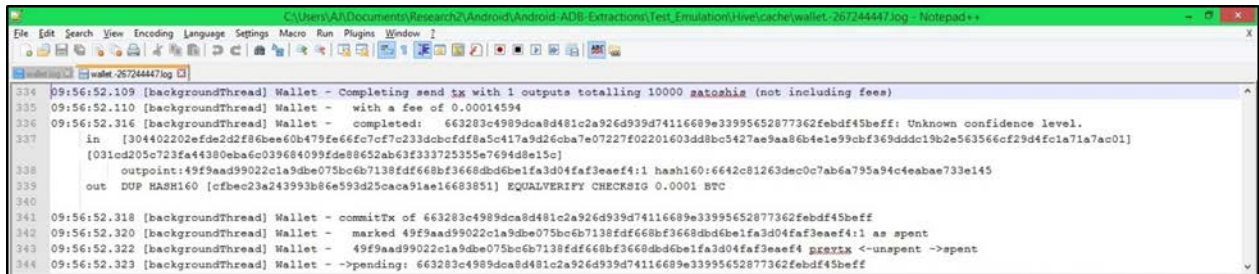


Figure 21. Entries within wallet.-26724447.log in Notepad++ listing TX information for the first sent transaction for the Hive Wallet.

The wallet-dump.1791589373.txt was another log of the wallet activity; however, this file was more of a summary of the transactions made within the wallet. This summary included the public wallet key, the amount of bitcoins currently in the wallet and the methods of transactions in which by that value was reached, followed by more detailed descriptions of these sent and received transactions. Figure 22 provides a basic view of this log in Notepad++.

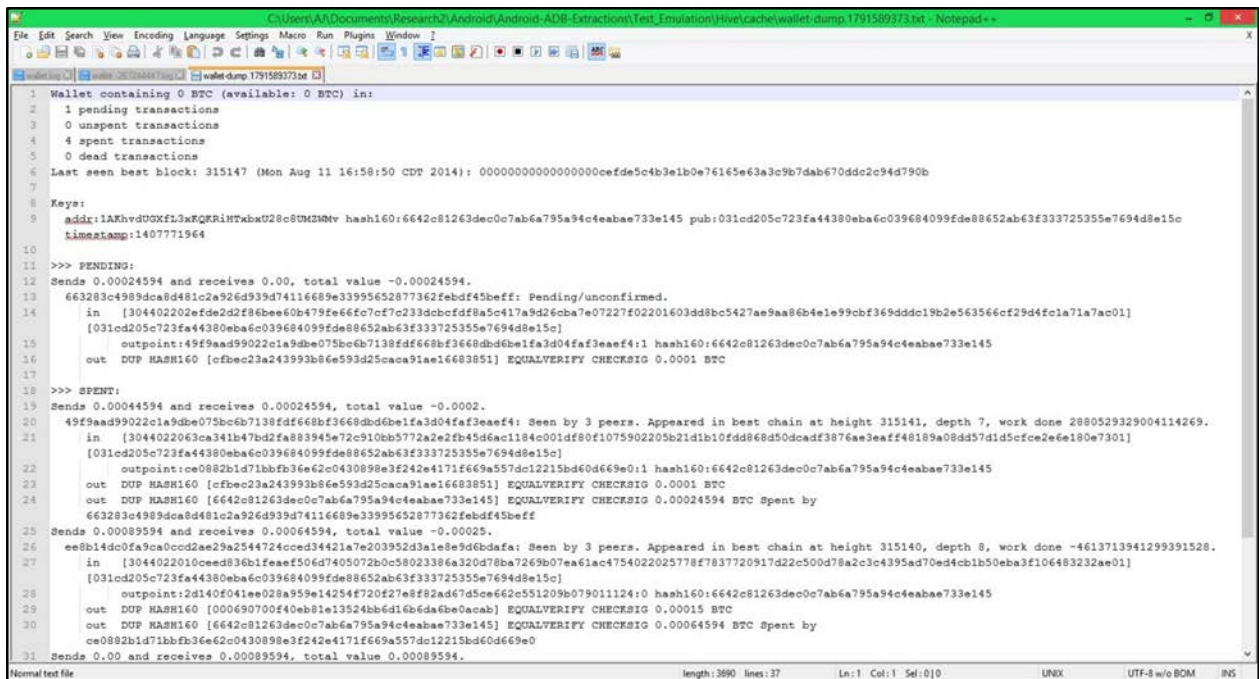


Figure 22. Data within the wallet-dump.1791589373.txt file in Notepad++ constituting Hive Wallet transaction summaries.

The files in the databases folder included address_book.file, address_book-journal.file, manifests.file, and manifests-journal.file. Address_book.file contained no data in its tables when opened in DB Browser for SQLite and address_book-journal.file was unreadable in Notepad++. When opened in DB Browser for SQLite, manifests.file had a table labeled manifests, which contained a singular entry with data pertaining to the Hive Wallet application (see Figure 23). The manifests-journal file was unreadable when opened in Notepad++.

The screenshot shows a window titled 'Database Browser for SQLite - C:/Users/Al/Documents/Research2/Android/Android-ADB-Extractions/Test_Emulation/Hive/databases/manifests'. The interface includes a menu bar (File, Edit, View, Help), buttons for 'New Database', 'Open Database', 'Write Changes', and 'Revert Changes', and tabs for 'Database Structure', '@Browse Data', 'Edit Pragma', and 'Execute SQL'. The 'Database Structure' tab is active, showing a table named 'manifests'. The table has columns: id, id, version, name, author, contact, description, icon, accessedHosts, apiVersionMajor, apiVersionMinor, and sortPriority. A single record is displayed with the following values: id=1, id=wei-lu.app-store, version=1.2.2, name=App Store, author=Wei Lu, contact=wei@hivewallet.com, description=A marketplace for Hive apps, icon=file:///android_asset/wei-lu.app-store/images/logo.png, accessedHosts=hive-app-registry.herokuapp.com,hive-app-registry-android.herokuapp.com, apiVersionMajor=0, apiVersionMinor=2, and sortPriority=100.

id	id	version	name	author	contact	description	icon	accessedHosts	apiVersionMajor	apiVersionMinor	sortPriority
1	wei-lu.app-store	1.2.2	App Store	Wei Lu	wei@hivewallet.com	A marketplace for Hive apps	file:///android_asset/wei-lu.app-store/images/logo.png	hive-app-registry.herokuapp.com,hive-app-registry-android.herokuapp.com	0	2	100

Figure 23. Manifests table in the manifests database from the ADB pull of the Hive Wallet application.

Extracted into the files folder were three files: key-backup-protobuf.file, key-backup-protobug.93, and wallet-protobuf.file. The data of all three files was unreadable when opened in Notepad++. Finally, in the shared_prefs folder was a single file labeled com.hivewallet.androidclient.wallet_preferences.xml. The data in this file was just a short code script for some of the wallet application preferences. After deletion of the Hive Wallet application from the emulator, its application data folder ceased to exist and thus could not be pulled.

IV. C. 3. Litecoin Wallet Results

From the ADB pull of the Litecoin Wallet application, five folders were extracted: app_blockstore, app_log, databases, files, and shared_prefs. In the app_blockstore folder, blockchainlitecoin.file was the only file extracted, which contained no readable data when opened in Notepad++. The app_log folder contained the wallet.log file chronologically describing the wallet application's activity. In the databases folder were the files address_book.file and address_book-journal.file, which, respectively, contained no data in its tables when opened in DB Browser SQLite and was unreadable in Notepad++. The files folder contained three extracted files: key-backup-base58litecoin.93, litecoin.peerdb, and wallet-protobuflitecoin.file. The .93 file, when opened in Notepad++ (see Figure 24), contained data that essentially gave a warning to the user to protect the private key of the wallet, along with the actual private key for the wallet and the date and time stamp for when the wallet was created and opened for the first time. The other two files were not readable when opened in Notepad++ and the Litecoin.peerdb file could not be opened in DB Browser for SQLite. Lastly, in the shared_prefs folder was the de.schildbach.wallet_ltc_preferences.xml file, which just contained a short code script for some of the wallet application preferences. After deletion of the Litecoin Wallet application from the emulator, its application data folder ceased to exist and thus could not be pulled.

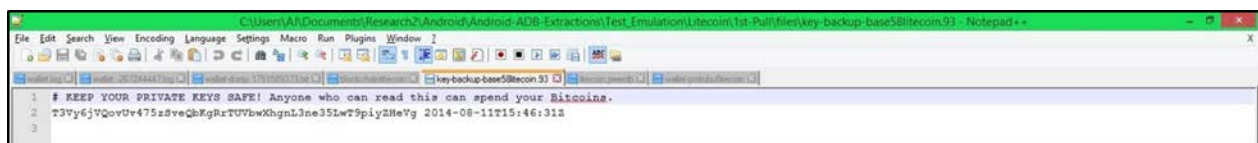


Figure 24. Data within the wallet-dump.1791589373.txt file in Notepad++: a security warning and the private key for the Litecoin Wallet.

IV. C. 4. Darkcoin Wallet Results

From the ADB pull of the Darkcoin Wallet application, five folders were extracted: app_blockstore, app_log, databases, files, and shared_prefs. The app_blockstore folder contained the blockchain file, which again was not readable in Notepad++. The app_log folder held the informative wallet.log file. The databases folder contained the address_book and address_book-journal files, which, respectively, contained no data in its tables when opened in DB Browser for SQLite and was unreadable in Notepad++. Into the files folder, key-backup-protobuf.file, key-backup-protobuf.93, and wallet-protobuf.file were extracted. None of these three files contained readable data. Finally, in the shared_prefs folder was the hashengineering.darkcoin.wallet_preferences.xml file, which only contained a short code script for wallet application preferences. After deletion of the Darkcoin Wallet application from the emulator, its application data folder ceased to exist and thus could not be pulled.

IV. C. 5. All Application Extraction Results

In addition to the data extracted from the ADB pulls of the specific wallet applications, the data extracted from pulling all of the application folders was also analyzed for potential artifacts. Out of all of the files pulled, only two new database files were found to contain data pertinent to the wallet applications. The launcher.db file in the com.android.launcher folder previously discovered in Physical Analyzer was also extracted in the ADB pull and contained the same entries for the wallets in its favorites table. As had also been observed in the analysis of the physical Android device, these entries did not exist in the launcher database once the wallet applications were deleted. The first new file was in the com.android.providers.downloads folder. In the databases sub-folder, the downloads.db file was opened in DB Browser for SQLite, where

the downloads table was found to contain four entries, one for each of the downloaded wallet applications. There were numerous columns in this table, of note were the uri, _data, lastmod, and title columns. URI stands for uniform resource identifier and is a string of characters that identifies a resource (Masinter, Berners-Lee, and Fielding, 1998). The URI column alone contains an abundance of information about the wallet applications, such as the source URL of these downloaded application, the file type of the download, the Google ID of application, and the Unix Epoch (Unix timestamp) for when the application was first opened on the device. The _data column indicates where the downloaded file was stored, the lastmod column indicates the Unix Epoch (Unix timestamp) that the wallet application file was downloaded, and the title provides the title of the file downloaded. Figure 25 shows a SQL query for the data contained in these four columns. These table entries were still present in this extracted database after deletion of the wallet applications from the emulator.

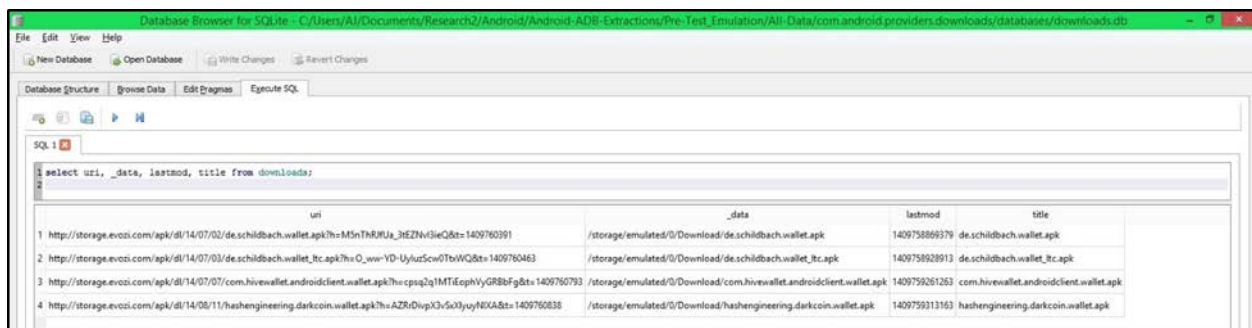


Figure 25. SQL query for uri, _data, lastmod, and title columns in the downloads table of the downloads database file located in the com.android.providers.downloads folder extracted from the ADB pull of All Application Data.

The second file of relevance from the ADB pull of all application data was the external.db file in the extracted com.android.providers.media folder. The files table contained four columns with pertinent data: _data, date_added, date_modified, and title. The data in these columns for the four wallet entries described the path where the downloaded APK file was saved, the Google ID of the wallet application, and the Unix Epoch (Unix timestamp) of the

wallet application installation and last modified time. Figure 26 shows a SQL query for the data contained in these four columns pertaining to the four wallet applications. These table entries were still present in this extracted database after the wallet applications were deleted.

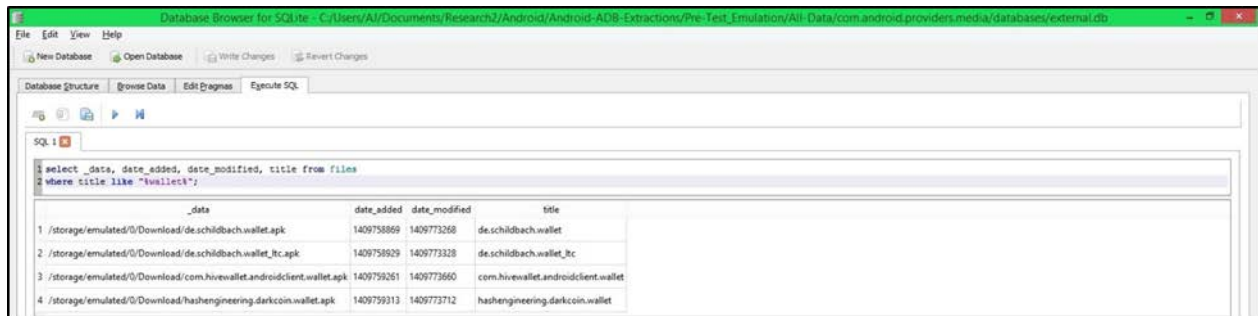


Figure 26. SQL query for `_data`, `date_added`, `date_modified`, and `title` columns in the `files` table of the `files` database file located in the `com.android.providers.media` folder extracted in the ADB pull of All Application Data.

V. Discussion

After analysis of the physical iOS device, it was concluded that UFED Physical Analyzer is a tool capable of determining active wallet application presence on a mobile iOS device. After installation of the bitWallet and CoinPocket wallet applications, many indicators of the wallets' presence could be gleaned from the UFED memory extractions. After trading, all of these indicators could still be seen; however, once the wallet applications were deleted from the device, all references to the wallets and their identifiers ceased to exist. There was also no transaction information of any sort to be found in the UFED extractions. As for iFunBox, this tool did allow more direct access to the applications themselves and the data held in their subfolders. Unfortunately, the data in these folders held nothing of relevance and were no longer present after deletion of the wallet applications. Like Physical Analyzer, iFunBox was deemed as useful only for determining active wallet application presence on an iOS device. In summary, the best approach for harvesting cryptocurrency wallet information on an iOS device would be to simply open up the wallet application itself and browse the transaction data provided there.

After analysis of the physical Android device, it was concluded that UFED is an effective tool for determining both present and past wallet application presence on an Android device. Physical Analyzer was able to parse out a substantial amount of data that indicate wallet presence, such as installation date and time stamps, last modified date and time stamps, and even the email account responsible for the download of the wallet applications onto the device. Unlike the data harvested by the iOS extraction tools, the wallet indicators captured for the Android device were still extracted after deletion of the wallet applications from the device. As for transaction data, again none could be found in the physical Android extractions.

After analysis of the virtual Android device, it was concluded that the ADB pull command-line tool is capable of extracting a wealth of valuable transaction information for active cryptocurrency wallet applications. The most relevant file pulled from each of the emulator wallet applications was the wallet.log file in the app_log folder. As previously described, this file was a historical record of the wallet's activity, the most important of which were transaction-related communications. By searching the term "tx" in a text-editing program like Notepad++, transaction data such as time stamps and transaction hashes could be found within the log file. Whether the transaction was an outgoing or incoming payment could also be established, as well as the final currency total in the wallet after completion of the transaction. While the IP addresses found in the wallet log were confirmed to be involved in the transaction, it could not be definitively said whether they were the source/destination of a transaction or if they were simply intermediate broadcast nodes for the transaction. The transaction hash is still valuable, though, because, once known, it can be searched on the blockchain.info website, where a detailed description of the transaction is publicly available. ADB pull was also capable of extracting information indicating wallet presence on the mobile device, but only if the wallet had

been installed via a downloaded APK file. These indicators were extracted after installation and persisted after deletion of the wallet applications.

VI. Conclusions and Forensic Relevance

Given that cryptocurrencies have been used in illegal transactions, the ability to forensically evaluate these digital currency systems and their cryptographic wallets is imperative. This research was successful in determining UFED Physical Analyzer and iFunBox as suitable tools for extracting wallet application indicators for the bitWallet and CoinPocket Bitcoin wallets off of an iOS mobile device. These tools, however, were unsuccessful in their ability to extract wallet data after the applications had been deleted from the mobile device and to harvest any sort of transaction information.

For Android mobile devices, it was discovered that the use of UFED Physical Analyzer against .ufd memory dump files is a reliable tool for determining past and present cryptocurrency wallet application presence on a device, while ADB pull performs well as an extraction tool for transaction information from a device with an active wallet application. Of particular interest to forensic examiners and law enforcement may be the IP addresses and transaction hashes found in the wallet.log files. If the case is serious enough, the IP addresses could potentially be used as a lead in the search for the source or destination address of an illicit cryptocurrency transaction. Likewise with the transaction hashes—if the hash is discovered or disclosed, it can be searched against the public ledger online. There, transaction data such as the received date and time stamp, the IP address of the first node to broadcast the transaction, and the input and output values are provided. Additionally, one can search a specific address against the public ledger to see all past transactions linked to the address. If an address has been connected to one known illegal

transaction, there is an opportunity to survey past and even monitor future transactions made by the address.

Looking beyond the discoveries of this research, further research into the forensic analysis of cryptocurrency wallets should indubitably be performed. In particular, investigation into an effective method of extraction of cryptocurrency wallet and transaction information from iOS devices would be of high value for digital forensic examiners. Tests of the ADB tools on a physical Samsung Galaxy S4 Android device instead of an emulator should also be conducted, as well as on devices running newer and older Android OS versions than 4.4.2 and devices beyond a Samsung Galaxy S4. Finally, a method of extracting transaction information from an Android device after wallet applications have been deleted would no doubt be of high forensic value.

As previously discussed, while many studies have been made into the public ledger of cryptocurrencies, investigation into the cryptocurrency wallets themselves, while of equal importance, is still deficient. Although the results from this research are a step in that direction, more tests and analyses must be performed in order to allow forensic examiners and law enforcement officers to effectively respond to the criminal web expanding within the rapidly developing realm of cryptographic currencies.

References

1. Virtual Currency: Bitcoin and Beyond, Part 1. [Internet]. Virtual Currency: Bitcoin and Beyond, Part 1. 2014 [cited 2014 July 10]. CIO Journal. Retrieved from http://deloitte.wsj.com/cio/2014/06/24/understanding-virtual-currency-bitcoin-and-beyond-part-1/?mod=wsjcio_hp_deloitte.
2. Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G. M., and Stefan Savage. 2013. A fistful of bitcoins: characterizing payments among men with no names. Proceedings of the 2013 conference on Internet measurement conference (pp. 127-140). ACM.
3. Nakamoto, Satoshi. 2008. Bitcoin: A peer-to-peer electronic cash system. Retrieved from <https://bitcoin.org/bitcoin.pdf>.
4. de la Porte, Lodewijk André. 2012. The Bitcoin transaction system. Utrecht. Netherlands.
5. Luther, William J. 2013. Cryptocurrencies, Network Effects, and Switching Costs. Mercatus Working Paper, Mercatus Center at George Mason University, Arlington, VA, forthcoming. Retrieved from <https://papers.ssrn.com/sol3/papers.cfm>.
6. How to Set Up a Wallet. [Internet]. BTC Gear. 2013 [cited 2014 July 9]. Retrieved from <http://bitcoinsimplified.org/get-started/how-to-set-up-a-wallet/>.
7. Mining. [Internet]. BTC Gear. 2013 [cited 2014 July 9]. Retrieved from <http://bitcoinsimplified.org/learn-more/mining/>.
8. Birukov, A., Khovratovich, D., and Ivan Pustogarov. 2014. Deanonymisation of clients in Bitcoin P2P network. arXiv preprint arXiv:1405.7418. Retrieved from <http://arxiv.org/pdf/1405.7418.pdf>.
9. Bitcoin Developer Guide. [Internet]. Bitcoin Project. 2009 [cited 2014 October 19]. Retrieved from <https://bitcoin.org/en/developer-guide#full-node>.
10. Sprankel, Simon. 2013. Technical Basis of Digital Currencies. Retrieved from <http://www.coderblog.de/wp-content/uploads/technical-basis-of-digital-currencies.pdf>.
11. Main Page. [Internet]. Litecoin Wiki. 2011 [cited 2014 July 02]. Retrieved from <https://litecoin.info/>.
12. Litecoin. [Internet]. Wikipedia. 2014 [cited 2014 July 02]. Retrieved from <http://en.wikipedia.org/wiki/Litecoin>.

13. Duffield, Evan and Kyle Hagan. 2014. Darkcoin: Peer-to-Peer Crypto-Currency with Anonymous Blockchain Transactions and an Improved Proof-of-Work System. Retrieved from <http://www.darkcoin.io/downloads/DarkcoinWhitepaper.pdf>.
14. What Is Darkcoin? [Internet]. Darkcoin. 2014 [cited 2014 June 30]. Retrieved from <http://www.darkcoin.io/intro.html>.
15. DarkSend. [Internet]. 2014. Darkcoin Wiki. 2014 [cited 2014 July 10]. Retrieved from <http://wiki.darkcoin.eu/wiki/DarkSend>.
16. Masinter, L., Berners-Lee, T., and R. T. Fielding. 1998. Uniform resource identifier (URI): Generic syntax. RFC 2396.
17. Intent | Android Developers. [Internet]. Android. [cited 2014 October 15]. Retrieved from <http://developer.android.com/reference/android/content/Intent.html>.