Journal of Cyber Forensics and Advanced Threat Investigations
CFATI OPEN ACCESS

Volume 2(1), 29-49. https://doi.org/10.46386/ijcfati.v2i1.22

Cyber Forensics on Internet of Things: Slicing and Dicing Raspberry Pi

Shuyuan Mary Ho[1], Mike Burmester[2]

Abstract

Any device can now connect to the Internet, and Raspberry Pi is one of the more popular applications, enabling single-board computers to make robotics, devices, and appliances part of the Internet of Things (IoT). The low cost and customizability of Raspberry Pi makes it easily adopted and widespread. Unfortunately, the unprotected Raspberry Pi device—when connected to the Internet—also paves the way for cyber-attacks. Our ability to investigate, collect, and validate digital forensic evidence with confidence using Raspberry Pi has become important. This article discusses and presents techniques and methodologies for the investigation of timestamp variations between different Raspberry Pi ext4 filesystems (Raspbian vs. UbuntuMATE), comparing forensic evidence with that of other ext4 filesystems (i.e., Ubuntu), based on interactions within a private cloud, as well as a public cloud. Sixteen observational principles of file operations were documented to assist in our understanding of Raspberry Pi's behavior in the cloud environments. This study contributes to IoT forensics for law enforcement in cybercrime investigations.

Notes for Practice

- This article serves up a cyber forensics practice example for examining IoT devices interacting in the cloud.
- Procedures and techniques used to examine file operational behaviors captured by Raspberry Pi and Ubuntu ext4 filesystems in the cloud are illustrated.
- Sixteen observational principles of IoT file operations have been derived for practitioners' reference.

Keywords

Timestamp Analysis, Cyber Forensic Techniques, File Systems, Internet of Things, Raspberry Pi, Human-Computer Interaction, Sociotechnical Cybersecurity.

Submitted: 03/02/2021 — **Accepted: 02**/04/2021 — **Published: 23**/05/2021

Corresponding author [1] Email: smho@fsu.edu Address: Florida State University, School of Information, 142 Collegiate Loop, Tallahassee, FL 32306-2100, U.S.A.. ORCID ID 0000-0002-4790-1821.

[2] Email: burmeste@cs.fsu.edu Address: Florida State University, Computer Science, 1004 Academic Way, Tallahassee, FL 32306-4530, U.S.A. ORCID ID 0000-0001-5094-5668.

1. Introduction

Technology is constantly evolving. From the automation of mechanical tools to the introduction of computer chips for ubiquitous application, cloud computing ushers in yet another new era of technology, where the majority of computational processing, storage, and analysis can be performed in centralized locations. This change has left "thin client" workstations and mobile devices with virtually one task—interacting with humans. Meanwhile, new chipset sensor technology enables mobile devices to collect human behavioral data. The more that mobile devices interact with humans, the more behavioral and consumption data can be captured and attuned to each individual, household, community, or organization. The era of the Internet of Things (IoT) allows us to integrate sensor technology with common goods and services, and to collect and profile users' behaviors.

The ability of the Internet of Things (IoT) to interact with humans creates opportunities for digital voice assistants like Alexa[1], Siri[2], and Cortana[3]. Enabled by artificial intelligence (AI), natural language processing (NLP), and natural language

[1] Alexa is a virtual assistant created by Amazon.
[2] Siri is a virtual assistant created by Apple.
[3] Cortana is a virtual assistant created by Microsoft.

generation (NLG), these smart and performance-driven support systems enable technological interaction and communication with humans, incorporating the ability to understand, give/receive commands to programmable IoT devices independently, and possibly fulfill more complex needs. Let's visualize a smart bulb as an IoT device. A smart bulb can refer to a traditional bulb converted by and integrated with a chipset. By connecting the chipset-enabled light bulb to the Internet, the smart light bulb can receive commands from—and transmit data automatically to—an AI-enabled digital voice assistant like Alexa. The chipset and its Raspberry Pi (aka "Pi") motherboard provide a gateway for a traditional bulb to become an IoT device, and thus be accessible online.

Just about any traditional device can be converted by Pi to become an IoT device. The connectivity to the Internet—whether through wired or wireless mechanisms—makes these devices (e.g., robotics, wearables, smart home appliances, or embedded electronic teapots, etc.) accessible from anywhere. The feature of a single chip computer—compatible with various Linux operating systems—makes Pi a highly customizable IoT solution. Moreover, the adoption of Pi is quite cost effective. Pi provides affordable and easy customizability for technology innovators, and thus becomes an ideal substitute for expensive IoT devices, like CCTV smart cameras. As Pi enables the construction of IoT devices, any technically savvy user can now easily convert a traditional camera to a smart camera and create a complete smart home environment. Your refrigerator can now provide a shopping list before you come home from work. You can communicate with—and instruct—Alexa to report near real-time updates about your house or office, through Pi-enabled IoT devices, while you are on vacation.

As the deployment of these Pi-enabled IoT devices increases, the potential for cyber threats impacting these IoT devices also increases. Hackers can now take controls over these IoT devices through the same Internet channel we would use to connect. Moreover, these IoT devices can be controlled as cyber zombies and botnets to flood cyberinfrastructure with distributed denial-of-service attacks. Our ability to understand how the filesystems work on the Pi-enabled IoT devices is critical. This article discusses techniques and methodologies of timestamp investigation of Pi-enabled physical IoT devices and compares the timestamps of these IoT devices with other Linux-based systems in both private and public Cloud environments. More specifically, our research question is: *to what extent do Raspberry Pi-enabled IoT devices differ when compared to other ext4 filesystems in the Cloud*?

2. Related Works

To understand digital forensics and how timestamps work in the Pi, we begin with a review of literature in cloud forensics and IoT forensics.

2.1. Cloud Forensics

Cloud storage has become a necessity in information management practices, for both individuals as well as organizations. Massive user demands have created additional framework options, such as "storage as a service" (StaaS) within cloud computing architecture (Ho et al. 2018). As sensitive or critical information may be stored in the cloud, cloud storage has thus become a target that attracts both curious users and malicious hackers. Chung et al. (2012) suggested the possibility of extracting traces of evidence from activities, such as browsing, uploading, and downloading files, during the usage of cloud storage services (e.g., Amazon S3, Dropbox, and Google Docs) in the local device (e.g., Windows 2000, XP, Vista, 7, or Mac O/S), and analyzing these traces.

Quick and Choo (2013b) analyzed the Dropbox cloud service and gave specific descriptions of the type of terrestrial artifacts and data remnants remaining on client devices such as Windows 7 and Apple iPhone 3G. More specifically, sufficient file references—with a broad range of the file examinations including directory listings, prefetch files, link files, thumbnails, registry, browser history, and memory captures—were concluded as being important in the overall determination of Dropbox usage on a local device.

Martini and Choo (2012) proposed a cloud forensic framework; and moreover, Martini and Choo (2013) adopted and validated this framework in an ownCloud case study, and suggested that an in-depth understanding of the artifacts is required to undertake cloud storage forensics—both client and server forensics. As a result, Martini and Choo (2013) provided technical recommendations of ownCloud StaaS instances, which include metadata analysis, authentication data, cached files, browser artifacts, mobile client artifacts, as well as network data analysis. Quick and Choo (2013a) also proposed another cloud forensics analysis framework, and this framework was adopted and validated in a Google Drive study (Quick and Choo 2014). In the case of the Google Drive study, Quick and Choo (2014) provided a technique for finding the traces of digital forensic evidences over several cloud platforms, and found that a user's password for accounts on Google Drive was stored in *cleartext* within the file (p. 182).

With the growing number of mobile phones and smart devices, pervasive connections via mobile devices and social networks are growing exponentially. The investigate-ability of pervasive social networks and big data concerns law enforcement agencies with regards to the ubiquitous presence of international crime. Quick and Choo (2017) proposed Digital Forensic Intelligence Analysis Cycle (DFIAC), which was adopted from a cloud forensic analysis framework (Quick and Choo 2013a), and can be used in locating information across an increasing volume of forensically extracted data from mobile devices. Based on DFIAC, Quick and Choo (2017) demonstrated how a huge volume of gigabytes data can be reduced to mere megabytes when test data was converted into spreadsheets like *csv*, *xls* or *xlsx*. However, the ability to investigate on, and make sense of, big data from pervasive mobile devices and social networks still lacks significantly in the discipline of cloud forensics.

Regarding Google Docs forensics, Roussev and McCulley (2016) proposed the concept of analyzing cloud-native digital artifacts, and discussed the residual digital artifacts that maintain in the persistent state of web/SaaS applications. Such artifacts can have a completely different structure and their state is often maintained in the form of a complete (or partial) log of user editing actions. Several of Google's APIs were examined, and observations were made to the Docs editor, Slides application and changelog, drawings objects changelog, and Sheets API. "Track changes" functionality within the artifacts were similar to how Microsoft Word would operate, but slightly different due to different formatting and user interface issues (p. S110). Roussev et al. (2016) developed a cloud drive acquisition tool: kumodd, to analyze full API-based acquisition of Google Drive, Dropbox, Box and Microsoft OneDrive. However, because this tool does not acquire cloud-native artifacts in their original form, Roussev and McCulley (2016) developed a new proof-of-concept tool, called kumodocs, to extract and process the history of documents and slides on Google Docs and Slides. This tool can also perform a quick privacy audit, as it can identify all the images, suggestions, comments that have been ostensibly deleted but still can be recovered. Ho et al. (2018) further explored the challenges of cloud forensics investigation into file access, transfer and operations, and identifying file operational and behavioral patterns based on timestamps. Timestamp patterns were observed contributing to the law enforcement community; a reference manual for uncovering data breach incidents occurring between the NTFS and ext4 filesystems in cloud forensics investigations.

2.2. IoT Forensics

Conceptually, any device that can be controlled through remote access connectivity can be considered an IoT device. Raspberry Pi can be easily adopted to the home environment. Casey (2015) suggested that smart home devices can be deployed in a wide range of household appliances—controlled and accessed via the Internet, as well as through voice or text commands. These smart home devices typically collect non-sensitive information such as energy sensor data or device usage; however, they can also store sensitive information such as credit card details, users' browsing history, and social networking activities. If hackers are able to access and eavesdrop on these devices, it could potentially result in fraudulent transactions. Casey (2015) stressed the importance of smart home forensics because of the possibilities of many new forms of burglary.

Raspberry Pi can also be deployed in healthcare and smart city context, due to its cost-effective features. Feng et al. (2017) however suggested that Pi has structural and operations issues regarding the integrity of the device—especially if the device is deployed as a processor. While several different Pi models were discussed, the authors examined and analyzed only the physical media layer, media management layer, file system layer, application layer, network layer, and memory layer for the Pi Model B (rev 2). Exploring the vulnerable side of the Pi, Feng et al. (2017) identified that the Raspbian operating system uses the secure shell protocol (SSH) on port 22 to establish remote connections. However, security issues arise when port 22 continues to remain open in situations where remote access is not required. Also, the default user credentials for Raspbian is well known to be the username 'pi' and password 'raspberry,' and users usually neglect to change this. One of the biggest concerns is with the massive deployment of Pi. When Pi devices are deployed in millions of homes—and especially in a healthcare context—the devices can be easily hijacked, turned into botnets, and used for automated distributed denial-of-service (DDoS) attacks (p. 11).

In a healthcare context, Murray (2017) explored the ability of installing Kali Linux on Pi, and used the Pi device to foresee penetration testing techniques. This study demonstrated how flexible the Pi can be when adopted as master nodes to perform man-in-the-middle attack and denial of service attack, or as sensor nodes to simulate the communication with each other, or as slave nodes (i.e., compromised zombies) to act based on the commands of master nodes. Murray (2017) proposed a revocable key-policy attribute-encryption protocol to control access rights directly on encrypted data. This approach can secure eHealth data sharing in the cloud environment between multiple organizations.

Ubiquitous use and always-on operation mode makes IoT devices a black box of user activities but can also become a great source of potential digital evidence. Similar to the Zawoad and Hasan (2015) multi-level forensics approach, Chung et al. (2017) further proposed a proof-of-concept Cloud-based IoT Forensics Tool (CIFT) in the study of Amazon Alexa ecosystem with the embedded smart speaker system, Amazon Echo. This CIFT—combining cloud-native forensics with client-side forensics—supports identification, acquisition and analysis of both native artifacts from the cloud as well as client artifacts from local devices.

On the other hand, the popularity—and the always-on mode—also enables IoT devices to produce reams of data, which challenges the investigation process in terms of extracting and processing evidence. Quick and Choo (2018) outlined a process for bulk digital forensic data analysis that includes disparate device data. Based on the Digital Forensics Intelligence Analysis Cycle described in Quick and Choo (2013a), Quick and Choo (2018) framed the process for digital forensics of disparate device into multiple phases. This dataset was analyzed for two purposes: (1) to explore the reliability of data from a fitness band device, and (2) to examine a process of analysis of a large volume of disparate data. Through these experiments, Quick and Choo (2018) provided proof that a malicious user's attempts to manipulate the time of activity in order to provide a false alibi cannot succeed, because the true time/date settings will be corrected when files are uploaded to the associated account.

3. Study Framework

Since timestamps are critical components in IoT device forensics, our immediate objective was to collect timestamps from different operating systems (Raspbian 9.14, UbuntuMATE, and Ubuntu 18.04), and to perform file operations from those ext4 filesystems—interacting with both a private cloud as well as the public cloud. We connected Raspbian Pi with Linux—representing a private cloud, while ownCloud and Dropbox represented the public cloud. Our aim was to collect, observe and compare timestamps of file operations, and to derive generalizable observation rules across different ext4 filesystems (***Figure 1***).

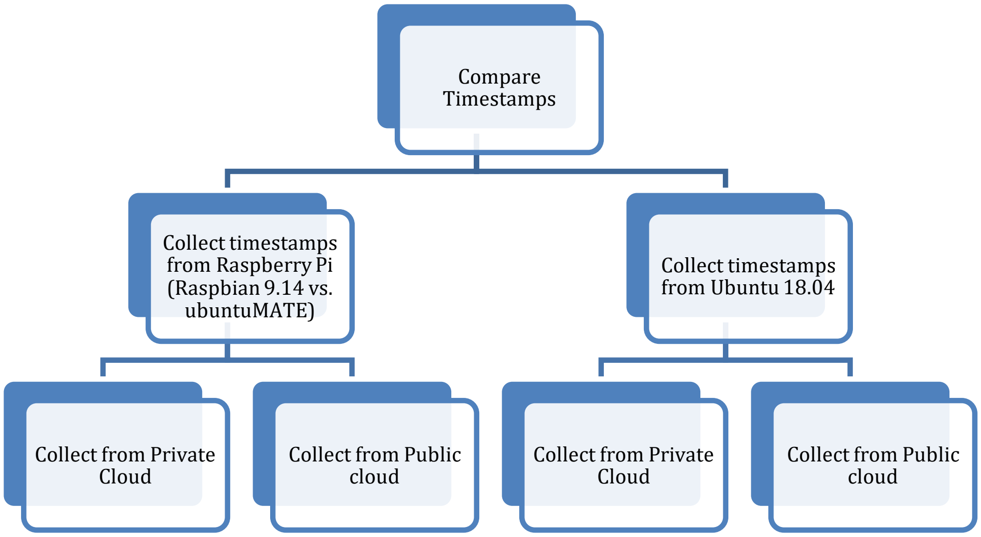
Compare Timestamps
Collect timestamps from Raspberry Pi (Raspbian 9.14 vs. ubuntuMATE)
Collect timestamps from Ubuntu 18.04
Collect from Private Cloud
Collect from Public cloud
Collect from Private Cloud
Collect from Public cloud

Figure 1. Study framework

3.1. Research Design and Experiment Setup

As **Error! Reference source not found.** illustrates, the research design is divided into three phases: 1) file operations on a standalone machine, 2) file transfer with the private cloud (Ubuntu Linux), and 3) file synchronization with the public cloud (ownCloud and Dropbox).

In this experiment, the extracted timestamp attributes include the create time (cr-time), modify time (m-time), access time (a-time), and change time (c-time). To obtain these metadata, we leverage the *debugfs* tool by running the inode number of that particular file in the *debugfs*. Inode is the index value of a file, which is a unique number in a partition. To set up *debugfs*, we mount the *debugfs* program with root permission. After mounting *debugfs*, we run the '*stat*' command, which gives the inode value of the respected file and passes that inode value to the *debugfs* command to provide the timestamps of a file. The same approach is used for the different filesystems, and results are used in comparison.

3.2. Computing Environment

Our computing environment was setup in the iSensor laboratory[4] at Florida State University School of Information. File operations were performed using test files, and timestamps were observed based on different attributes in the file timestamp

[4] iSensor Lab: https://isensoranalytics.com/

metadata. The following illustrates the computing environments adopted in three phases of the experiments. Two sets of standalone physical Pi systems with different operating systems were setup. An Ubuntu Linux system was also set up for timestamp comparison purpose because these three devices utilize the ext4 filesystem.

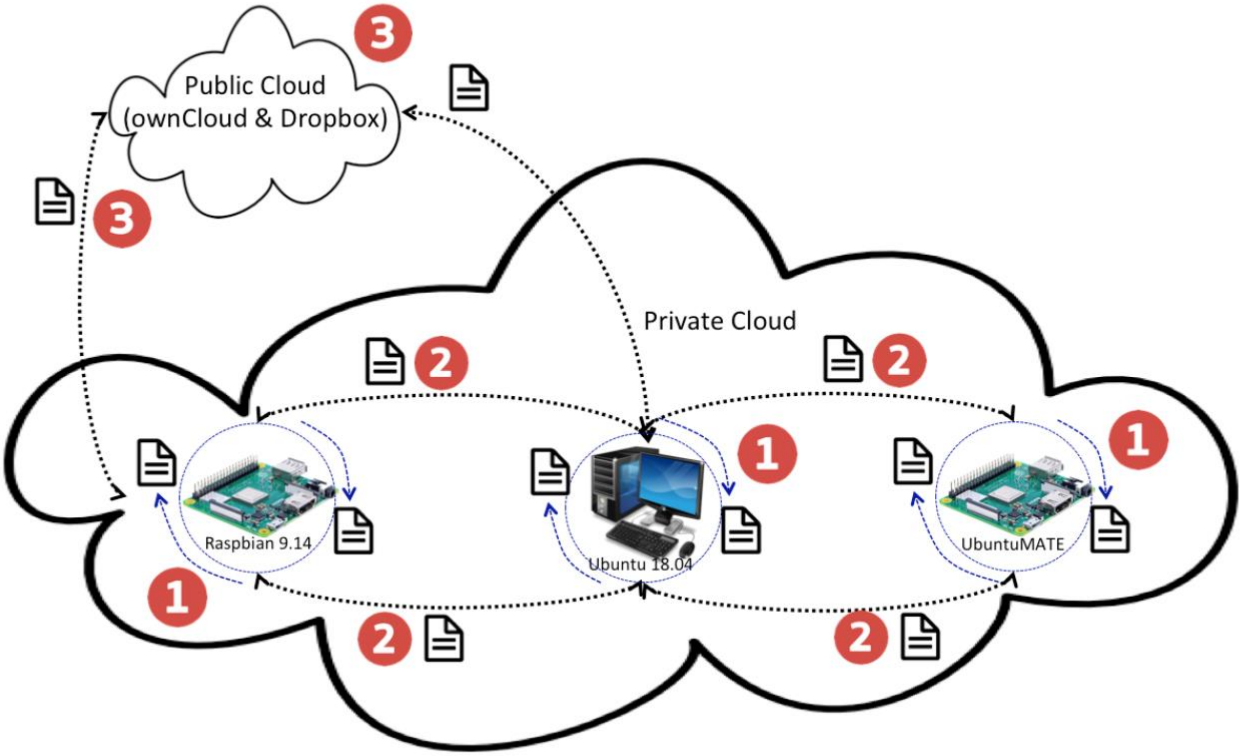

Figure 2. Experimentation design

⇒Raspberry Pi 3B+ with Raspbian 9.14 O/S

- O/S: Raspbian with Debian Stretch v9.14 on 32-bit
- Instruction Set Architecture: arm v7
- Filesystem: ext4
- Browser: Chromium 72.0.3626.121
- Forensic Tool: Command line *debugfs*
- File type: Text file (.txt)

⇒Raspberry Pi 3B+ with UbuntuMATE O/S

- O/S: UbuntuMATE 18.04 on 32-bit
- Instruction Set Architecture: arm v7
- Filesystem: ext4
- Browser: Firefox 67.0.1
- Forensic Tool: Command line *debugfs*
- File type: Text file (.txt)

⇒Ubuntu 18.04 O/S

- O/S: Ubuntu 18.04 LTS on 64-bit
- Instruction Set Architecture: x86
- Filesystem: ext4
- Browser: Firefox 67.0.1
- Forensic Tool: Command line *debugfs*
- File type: Text file (.txt)

In the first phase, file operations were performed on two sets of standalone physical Pi systems with different operating systems. One Ubuntu system was also set up for comparison purposes. Then, connections with both the private and public Cloud were set up. The following file operations were performed and compared: create, compress, decompress, access, modify, move, rename, copy and delete.

In the second phase, file transfer operations and interactions were performed between a Pi and an Ubuntu Linux system within the private cloud (illustrated in ***Figure 2***). The private cloud was also set up at the iSensor laboratory[4]—represented by direct cable connection between the Pi's and the Ubuntu Linux. Files were transferred between Pi's and Ubuntu Linux system. The following file operations were performed via SSH[5] and compared: create, upload, and download.

⇒Private cloud

- RJ45 Cat-6 Ethernet patch cable connection for office networks in a wired LAN

In the third phase, file transfer operations and interaction were performed between a Pi and public cloud (***Figure 2***). Files were transferred from a Raspberry Pi to public cloud using a browser and self-synchronizing command line interface.

⇒Public cloud

- ownCloud cloud service for Pi
- Dropbox cloud service for Ubuntu 18.04 LTS
- Internet connectivity: Florida State University FSUSecure Wi-Fi Networks

3.3. Data Collection

Timestamps were collected and recorded both before and after various file operations from the inode metadata information collected using the *debugfs* tool. In Appendix A, the timestamps recorded during various file operations are given. Tables were aggregated based on the timestamps recorded and compared during identical file operations in different settings.

3.4. Investigation Procedure

Both Pi devices (Raspbian 9.14 and UbuntuMATE) and Ubuntu (18.04) utilized an ext4 filesystem, and generated similar inode reports using *debugfs* procedures. Basically, after each file was created, we accessed the inode number using the stat command for that file. From there, we collected the timestamp metadata (i.e., *cr*-time, *m*-time, *a*-time, and *c*-time) before and after each file operation was performed sequentially. We then recorded the changes on whether or not each file operation had changed or updated the baseline timestamps.

The following steps were iteratively conducted and taken during each phase of the investigations.

Phase 1. Standalone file operations

- Step 1: Set up two Pi's (Raspbian 9.14 and UbuntuMATE) and one Ubuntu Linux (18.04 LTS), and properly install all of the starting software.
- Step 2: Determine the name of the ext4 filesystem used in each environment.
- Step 3: Mount the *debugfs* tool to the ext4 filesystem found in step 2 using root.
- Step 4: Create a test file and use the '*stat*' command to determine the inode numb.
- Step 5: Use the *debugfs* tool to gather the first set of timestamps.
- Step 6: Collect a baseline timestamp before preforming each file operation, then perform the operation and collect the new timestamp afterwards.

Phase 2. File transfer within the private cloud

- Step 1: Collect baseline timestamps on the test file.
- Step 2: Upload the test file to the private cloud environment.
- Step 3: Collect timestamps on the original file.
- Step 4: Download the test file and collect all timestamps again.

Phase 3. File synchronization with the public cloud

- Step 1: Collect baseline timestamps on the test file.
- Step 2: Synchronize the test file to either ownCloud or Drop.
- Step 3: Collect timestamps on the original file.

[5] The '*scp pi@ipaddress:myfile.txt*' command is used.

4. Observations

Our objectives include three phases of observations of the file operations performed on: (1) a standalone physical Pi between two operating systems, (2) file transfer operations within a private cloud, and (3) and file synchronization operations with public cloud (represented by the interaction with either ownCloud or Dropbox).

During the phase one observation, we compared the timestamps on Pi (Raspbian 9.14 vs. UbuntuMATE) and additionally Ubuntu 18.04. During the phase two observations, we compared file transfer operations and the interaction between Pi and Ubuntu 18.04 in a private cloud. During the phase three observations, we compared file synchronization operations and the interaction (1) between Pi and a public cloud (i.e., ownCloud), and (2) between Ubuntu and a public cloud (i.e., Dropbox). Our observations are showcased in ***Table 1*** through ***Table 3***.

4.1. Preliminary Observations

Preliminary observations and baseline data were collected to identify how different Pi operating systems change the way timestamps are collected in comparison with Ubuntu using the same sets of instructions on a text file (*.txt*). We observed that the timestamps of file operations were slightly different between Raspbian 9.14 and UbuntuMATE. Moreover, we observed significant differences between Raspberry Pi (armv7) and Ubuntu standalone system (x86) ext4 filesystems. These might be caused by differences in the instruction set architecture.

4.2. Direct Observations

Based on the experiment, we observed two general differences—when a file is modified and deleted—between Raspberry Pi running on Raspbian 9.14 and UbuntuMATE. The observations below are taken to compare timestamps across Raspbian 9.14, UbuntuMATE and Ubuntu 18.04. Data of direct observations were documented and reported from ***Table 4*** to ***Table 9*** in Appendix A.

4.2.1. Observation 1. File creation

The create time (*cr*-time) refers to the time of creation. When a file is created, all timestamps are recorded as the time of creation depending on the file creation commands.

- Rule 1: Create time (*c*-time) = Access time (*a*-time) = Modify time (*m*-time) = Change time (*c*-time) when a file is created using '*nano*' or '*echo*' commands. But if a file is created using '*cat*' command, it updates the create time and access time to the time of creation whereas it updates the modify time (*m*-time) and change time (*c*-time) to the time of modification across Raspberry Pi and Ubuntu 18.04.

4.2.2. Observation 2. File compression

When a file is compressed, it changes the timestamps with the time of compression. However, the timestamps will vary depending on different compression algorithms and operations.

- Rule 2: When a file is compressed using the '*tar'* [6] command, all timestamps are changed to the time of compression in both Raspberry Pi and Ubuntu 18.04 system.
- Rule 3: When a file is compressed using the '*gzip'* [7] command, the change time (*c*-time) and the create time (*cr*-time) are changed to the compression time, whereas the access time (*a*-time) and modify time (*m*-time) are not changed in both Raspberry Pi and Ubuntu 18.04 system.

[6] 'tar' is a Linux command used for file compression and decompression.

[7] 'gzip' is a Linux command used for file compression and decompression.

Table 1. *File operations on standalone Raspberry Pi and Ubuntu*

File Operation		Raspberry Pi Raspbian 9.14				Raspberry Pi UbuntuMATE				Ubuntu Linux 18.04			
		Create	Modify	Access	Change	Create	Modify	Access	Change	Create	Modify	Access	Change
		cr-time	m-time	a-time	c- time	cr-time	m-time	a-time	c- time	cr-time	m-time	a-time	c- time
1. Create	nano	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change
	echo	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change
	cat	No Change	File modification time	No Change	File modification time	No Change	File modification time	No Change	File modification time	No Change	File modification time	No Change	File modification time
2. Compress	Tar(.tar)	file compression time	file compression time	file compression time	file compression time	file compression time	file compression time	file compression time	file compression time	file compression time	file compression time	file compression time	file compression time
	Gzip (.gz)	file compression time	No Change	No Change	file compression time	file compression time	No Change	No Change	file compression time	file compression time	No Change	No Change	file compression time
3. Decompress	Tar(.tar)	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	file decompression time	No Change
	Gzip (.gz)	No Change	file decompression time	No Change	file decompression time	No Change	file decompression time	No Change	file decompression time	No Change	file decompression time	file decompression time	file decompression time
4. Access	GUI	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	file access time	No Change
	cat	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	file access time	No Change
5. Modify	GUI	No Change	File modification time	No Change	File modification time	File modification time	File modification time	File modification time	File modification time	File modification time	File modification time	File modification time	File modification time
	nano	No Change	File modification time	No Change	File modification time	No Change	File modification time	No Change	File modification time	No Change	File modification time	File modification time	File modification time
6. Move	mv	No Change	No Change	No Change	File move time	No Change	No Change	No Change	File move time	No Change	No Change	No Change	File move time
7. Rename	rename	No Change	No Change	No Change	File rename time	No Change	No Change	No Change	File rename time	No Change	No Change	No Change	File rename time
8. Copy	cp*$_{\text{same directory}}$	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	File copy time	No Change
	cp*$_{\text{different directory}}$	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	File copy time	No Change
	cp*$_{\text{different directory and new name}}$	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	File copy time	No Change
9. Delete	rm	No Change	File delete time	No Change	File delete time	No Change	No Change	No Change	File delete time	No Change	File delete time	File copy time	File delete time

Table 2. *File transfer operations and interaction within the private Cloud*

File Operation	Raspberry Pi 9.14 with Ubuntu 18.04				Raspberry Pi UbuntuMATE with Ubuntu 18.04			
	Create time	Modify time	Access time	Change time	Create time	Modify time	Access time	Change time
	cr-time	m-time	a-time	c-time	cr-time	m-time	a-time	c-time
Uploading a file from Pi to private cloud (Linux)	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change
Downloading a file from private cloud (Linux) to Pi	No Change	File download time	No Change	File download time	No Change	File download time	No Change	File download time

Table 3. *File synchronization operations and interaction with the public Cloud (ownCloud)*

File Operation	Raspberry Pi 9.14 with ownCloud				Raspberry Pi UbuntuMATE with ownCloud			
	Create time	Modify time	Access time	Change time	Create time	Modify time	Access time	Change time
	cr-time	m-time	a-time	c-time	cr-time	m-time	a-time	c-time
Send file from Pi to cloud (OwnCloud) using sync	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change
Send file from Pi to cloud (OwnCloud) using browser	File upload time	No Change	No Change	File upload time	File upload time	No Change	No Change	File upload time

4.2.3. Observation 3. File decompression

The file compression forms the baseline timestamp. When a file is decompressed, both the time of decompression as well as the time before the compression are recorded in timestamps.

- Rule 4: When a compressed file is decompressed using the ‘*tar*’ command, there is no change in the file’s timestamps in Raspberry Pi where the access time (*a*-time) is updated to the time of decompression in Ubuntu 18.04 system.
- Rule 5: When a compressed file is decompressed using the ‘*gzip*’ command, the create time (*cr*-time) and access time (*a*-time) are not changed, but the modify time (*m*-time) and change time (*c*-time) are changed to the time of decompression in Raspberry Pi system whereas the create time (*cr*-time) remains the same while the rest of the three timestamp characteristics are updated to the decompression time in the Ubuntu 18.04 system.

4.2.4. Observation 4. File access

On accessing a file without making any modifications, updates of the timestamp will change based on the latest time of access. The timestamps are recorded depending on the mode of operation.

- Rule 6: When a file is accessed using both GUI and the ‘cat’ command, there is no change in the timestamps of the file in Raspberry Pi system, but the access time (a-time) of the file is updated to the latest time of access in Ubuntu 18.04 system.

4.2.5. Observation 5. Modify

When modifying a file, different operating environments each have an effect on the timestamps.

- Rule 7: When a file is modified using GUI, the create time (*cr*-time) and access time (a-time) are not changed, but the modify time (*m*-time) and change time (*c*-time) are updated to the time of modification on Raspbian 9.14. When a file is modified using GUI, all timestamps are updated to the time of file modification across UbuntuMATE and Ubuntu 18.04. This is because the filesystem assigns a new inode value after the file is saved.
- Rule 8: When a file is modified using the ‘*nano*’ command, the create time (*cr*-time) and access time (*a*-time) are not changed, but the modify time (*m*-time) and change time (*c*-time) are updated to the time of modification in Raspberry Pi systems. However, the create time (*c*-time) is not changed, but the other three timestamps are updated to the time of modification in Ubuntu 18.04 system.

4.2.6. Observation 6. Move

When moving a file, the timestamp remains the same in both Raspberry Pi systems as well as the Ubuntu 18.04 system.

- Rule 9: When a file is moved using the ‘*mv*’ command, the change time (*c*-time) is updated to the time of file move, and the rest of the timestamps remain the same.

4.2.7. Observation 7. Rename

On renaming a file, the timestamp remains the same in both Raspberry Pi systems as well as Ubuntu 18.04 system.

- Rule 10: When a file is renamed, the change time (*c*-time) is updated to the time of rename, and the rest of the timestamps remain unchanged.

4.2.8. Observation 8. Copy

On copying a file, a new file is created in the filesystem, which generates a new inode value with new timestamps, all reflecting the time of creation. But this also makes some changes in the timestamps of the original file.

- Rule 11: When a file is copied in the same directory, or in a different directory using the same name, or into a different directory using a different name, a new file is created with new timestamps reflecting the time of creation, while the timestamp of the original file remains unchanged in Raspberry Pi systems.
- Rule 12: When a file is copied in the same directory, or in a different directory using the same name, or into a different directory using a different name, a new file is created with new timestamps reflecting the time of creation, but the access time (*a*-time) of the original file is updated to the time of creation of the new file in Ubuntu 18.04 system.

4.2.9. Observation 9. Delete

On deleting a file, some of the timestamp characteristics are updated with the time of file deletion depending on the operating system.

- Rule 13: When a file is deleted, the create time (*cr*-time) and access time (*a*-time) are not changed, whereas the modify time (*m*-time) and change time (*c*-time) are updated to the time of deletion in Raspbian 9.14 and Ubutntu 18.04 systems.

- Rule 14: When a file is deleted, the change time (*c*-time) is updated to the time of deletion, but the rest of the timestamps remain the same in UbuntuMATE.

4.2.10. Observation 10. File transfer to private cloud

We observed file operations when a file is transferred to a private cloud (Ubuntu 18.04). The '*SCP*' (secure copy) command—'*scp pi@ipaddress:myfile.txt*'—is used.

- Rule 15: There is no change in the timestamp when uploading a file from the Raspberry Pi system to the private cloud.
- Rule 16: The change time (*c*-time) and modification time (*m*-time) are updated to the time of download when the same file is downloaded back to the same location on both Raspberry Pi's.

4.2.11. Observation 11. File upload/ download to public cloud

The public cloud enables file synchronization. When a file is uploaded to the cloud, it is automatically synchronized with the local drive. Thus, there is no download operation in our observations. We observed the file synchronization operation with the public cloud. ownCloud is used as our public cloud through the self-synchronization system as well as through web browser.

- Rule 17: There was no change in the timestamps of the file when uploading a file to the public cloud using self-synchronization system in both Raspberry Pi's.
- Rule 18: The create time (*cr*-time) and change time (*c*-time) are updated to the time of upload while the rest of the timestamps remain unchanged using a browser in both Raspberry Pi's.

4.3. Cross-Sectional Observations

We compared different file operations across ext4 filesystems including Raspbian 9.14, UbuntuMATE, Ubuntu 18.04, Ubuntu 18.04 (VM), Ubuntu 14.04 (VM), and conducted cross- sectional analysis on timestamps collected during the same time period to derive the following observation rules.

4.3.1. Observation 12. Updates of create time

We compared the file's create time (*cr*-time) and derived that the create time is updated when a file is compressed, but not updated when the file is decompressed, accessed, moved, renamed or copied.

- Rule 19: Create time is updated when the file is compressed across all five operating systems and updated when the file is modified using a GUI across Ubuntu operating systems, but not Raspbian 9.14.

4.3.2. Observation 13. Updates of modify time

We compared the file's modify time (*m*-time) and observed that the timestamp is updated when a file is created, compressed, decompressed, and modified across all five file operations.

- Rule 20: Modify time (*m*-time) is updated when a file is created using '*cat*' command, but not '*nano*' and '*echo*' commands across all operating systems.
- Rule 21: Modify time (*m*-time) is updated when a file is compressed using '*tar*' command, but not '*gzip*' command across all operating systems.
- Rule 22: Modify time (*m*-time) is updated when a file is modified using both 'GUI' and '*nano*' commands across all the five platforms.
- Rule 23: Modify time (*m*-time) is updated when a file is deleted in Raspbian 9.14, Ubuntu 18.04 and Ubuntu 18.04 VM, but not in UbuntuMATE and Ubuntu 14.04 VM.

4.3.3. Observation 14. Updates of access time

We compared the file access time (*a*-time) and observed that it is updated when a file is compressed across all five operating systems. However, the access time is also updated when a file is compressed, decompressed, accessed, modified, moved, or copied in Ubuntu 18.04.

- Rule 24: Access time (*a*-time) is updated when a file is compressed using '*tar*' across all five operating systems. But the timestamp is updated only when a file is decompressed using '*tar*' in virtual environment (Ubuntu 18.04 and Ubuntu 14.04). Also, the timestamp is updated when a file is compressed using '*gzip*' in both Ubuntu 18.04, but the timestamp is updated only when a file is decompressed using '*gzip*' in both Ubuntu 18.04 and 14.04 VM.
- Rule 25: Access time (*a*-time) is updated when a file is accessed using both GUI and '*cat*' command in Ubuntu 18.04.
- Rule 26: Access time (*a*-time) is updated when a file is modified using GUI in all Ubuntu systems whereas when using a '*nano*' command, it is updated in both Ubuntu 18.04 and Ubuntu 14.04.
- Rule 27: Access time (a-time) is updated when a file is renamed or moved only in Ubuntu 14.04.

- Rule 28: Access time (*a*-time) is updated when a file is copied in both Ubuntu 18.04.

4.3.4. Observation 15. Updates of change time

We compared file change time and observed that it is updated when a file is created, compressed, decompressed, modified, moved, renamed or deleted, across all five operating systems.

- Rule 29: Change time (*c*-time) is updated when a file is created using the '*cat*' command across all five operating systems, but it remains unchanged when '*nano*' and '*echo*' commands are used.
- Rule 30: Change time (*c*-time) is updated when a file is compressed using 'tar' and '*gzip*' commands whereas the *c*-time is updated only when a file is decompressed using the '*gzip*' command.
- Rule 31: Change time (*c*-time) is updated when a file is modified, moved, renamed or deleted across five operating systems using both GUI and the '*nano*' command.

4.3.5. Observation 16. Deletion of disingenuous file

We compared the cross-sectional data of the file's create time, access time, modify time and change time, and observed the following rule:

- Rule 32: When a file's create time (*cr*-time) is recorded to be later than the modify time (*m*-time) or access time (*a*-time), it indicates that the file was uploaded or downloaded to the cloud from Raspberry Pi System.

5. Results and Discussion

When conducting forensic experiments on IoT devices (such as the Raspberry Pi), it is important to be aware of changes to the inode timestamps incurred by various file operations. In this section, we not only discussed the significant changes of the file operation on the current IoT forensics study of the ext4 filesystems, but also compared the current findings with the observations of Ubuntu ext4 filesystems in the standalone as well as the cloud settings conducted by Ho et al. (2018).

5.1. Differences between Pi Raspbian 9.14 and Pi with UbuntuMATE O/S

Although the same set of file operations were performed, there is only one noticeable difference—i.e., when modifying a file using GUI—as observed between two Raspberry Pi's. When a file is modified using GUI, the modify time (*m*-time) and change time (*c*-time) are updated to the time of modification in Raspbian 9.14. However, UbuntuMATE assigns a new inode number to the modified file as soon it is saved. All four timestamps are updated to the time of modification (observation rule 7).

5.2. Differences between Pi Ext4 and Ubuntu Ext4

When comparing Pi's operations with Ubuntu, we observed that the ext4 filesystems in Pi's are less sensitive than the ext4 filesystems in Ubuntu. That is, we noticed six (6) significant differences between the Raspbian and the Ubuntu 18.04 ext4 filesystems. First, when a compressed file is decompressed using the '*tar*' command, there is no change logged in Raspberry Pi, whereas in Ubuntu, the access time (*a*-time) is updated to the time of decompression (observation rule 4). Second, when a compressed file is decompressed using the '*gzip*' command, the modify time (*m*-time) and change time (*c*-time) are updated to the time of decompression in Raspberry Pi. However, the modify time (*m*-time), access time (*a*-time) and change time (*c*-time) are updated to the decompression time is in Ubuntu (observation rule 5). Third, when a file is accessed using both GUI and the '*cat*' command, there is no change in the Raspberry Pi, but in Ubuntu the access time (*a*-time) of the file is updated to the latest time of access (observation rule 6). Fourth, when a file is modified using GUI, the modify time (*m*-time) and change time (*c*-time) are updated to the time of modification on Raspberry Pi. However, all timestamps are updated to the time of file modification in Ubuntu (observation rule 7). Fifth, when a file is modified using the '*nano*' command, the modify time (*m*-time) and change time (*c*-time) are updated to the time of modification in Raspberry Pi whereas in Ubuntu 18.04 the modify time (*m*-time), access time (*a*-time) and change time (*c*-time) are updated to the time of modification (observation Rule 8). Finally, when a file is copied, all timestamps of the original file remain unchanged in Raspberry Pi whereas the access time (*a*-time) of the original file is updated to the time of creation of the new file in Ubuntu (observation rule 12).

5.3. Differences between Ubuntu Ext4

To compare the differences of file operations among different Ubuntu ext4 filesystems, we further set up two virtual machines (i.e., Ubuntu 18.04 and 14.04 VMs) in the Hyper-V management system (***Table 10*** in Appendix B)—running on Windows Server 2016 Datacenter—so we could observe how ext4 filesystems of the same O/S behave on a different environments (i.e., physical vs. virtual) in order to compare our results with the findings from Ho et al. (2018). We adopted the same experimental approaches—performing the same set of file operations and instructions as illustrated in Ho et al. (2018). As a result, we observed five (5) differences when comparing both Ubuntu 18.04 and the Ubuntu 14.04 Virtual Machine. The behaviors of the three Ubuntu systems are not completely the same. More specifically, both the Ubuntu 18.04 standalone system and Ubuntu 18.04 Virtual Machine have no difference in the three phases of the observations (***Table 11***). However, we attribute the ext4 filesystems in Ubuntu 18.04 as being more sensitive than ext4 filesystems in Ubuntu 14.04, which includes the following differences.

First, when a file is accessed without making any modification using GUI or the '*cat*' command, the access time (*a*-time) is updated to the time of access in both Ubuntu 18.04, whereas there is no change in Ubuntu 14.04. Second, when a file is moved, the change time (*c*-time) is updated to the time of move in both Ubuntu 18.04, whereas the access time (*a*-time) and change time (*c*-time) are both updated to the time of move in Ubuntu 14.04. Third, when a file is renamed, the change time (*c*-time) is updated to the time of rename in both Ubuntu 18.04 whereas the access time (*a*-time) and change time (*c*-time) are both updated to the time of rename in Ubuntu 14.04. Fourth, when a file is copied, the access time of the original file is updated to time of copy in both Ubuntu 18.04, whereas there is no change in Ubuntu 14.04. Fifth, when a file is deleted, the modify time (*m*-time) and change time (*c*-time) are updated to the time of deletion in both Ubuntu 18.04, whereas only the change time (*c*-time) is updated to the time of deletion in Ubuntu 14.04.

6. Conclusion

As cloud computing and the Internet of Things (IoT) become increasingly prevalent in everyday use, it is important for law enforcement investigators to understand how these files operate, as well as how they are recorded and transferred in different filesystems (e.g., the ext4 filesystem) over the Cloud environments. Timestamp metadata can change across the ext4 filesystems—even when executing the same file operations. This IoT forensics study was designed and conducted to examine and compare how timestamps vary within IoT devices, particularly between Raspberry Pi (Raspbian 9.14 vs. UbuntuMATE) and Ubuntu (e.g., 18.04). We further experimented on file transfer operations; from the Raspberry Pi (Raspbian 9.14 vs. UbuntuMATE) to the private cloud (i.e., Ubuntu 18.04), and moved further into file synchronization operations from the Raspberry Pi (Raspbian 9.14 vs. UbuntuMATE) to the public cloud (i.e., ownCloud and Dropbox).

We anticipate that the reasons for differences in some observations among the Pi's are caused by the differences in their O/S (i.e., Raspbian 9.14 vs. UbuntuMATE). We further anticipate the reasons for the differences between the Pi and Ubuntu 18.04 or 14.04 may be caused by the different architectures of the hardware processors (i.e., arm7 vs. X.86). Different versions of hardware processors such as Raspberry Pi can support only arm architecture. In general, the Ubuntu 18.04 is more sensitive than the Pi's or the Ubuntu 14.04.

6.1. Limitations and Future Directions

The authors were able to analyze the *ext4* filesystem in the Raspbian environments with arm architecture and Ubuntu with X.86 architecture. Analyzing other proprietary IoT O/S—with Linux expansions—could provide for a more complete dataset while also painting a more accurate picture of the IoT forensic investigative process. Although Raspberry Pi is an incredibly versatile tool with boundless possibilities in the IoT field, it is still limited by its processing power, and must use extremely lightweight distributions. Further exploration of the NTFS and FAT32 filesystems could provide a more comprehensive view should these filesystems be adopted in the IoT world. In the current study, differences in timestamp metadata within ext4 filesystems were identified. The same examination of file operations could be performed on different Linux distributions, which may yield a greater varied set of results and observations.

6.2. Implications to Practitioners

It is a common practice for organizations—both in public and private sectors—to store and access corporate information in the cloud environment. The threats that organizations face and experience are twofold. First, IoT devices make the modern lifestyle more connected, such as connected cars, homes, hospitals, healthcare facilities, retail stores, manufacturing, and smart cities. However, the connectivity of IoT devices make our society more vulnerable as cyberinfrastruture can be compromised. IoT devices can be used by hackers to form botnets and launch distributed denial-of-service (DDOS) attacks against corporate backend servers that can exploit users' privacy, which further cause safety concerns. Second, organizations face significant white collar crime and insider threats in unauthorized modification of sensitive and/or classified information. It is critical for cyber forensics practitioners to gain intrinsic knowledge and techniques of how file systems operate and behave for advanced threat investigation. This study provides detailed steps and techniques for IoT forensics, and is significantly relevant to cyber investigation. The study demonstrates mechanisms of digital evidence analysis and observation with regards to access and modification of file systems within normal versus abnormal conditions.

6.3. Contributions

The purpose of this study is to build a thorough understanding of how the *ext4* filesystem timestamps reflect file operation activities in cloud environments. This cyber forensics study—specifically, IoT forensics—was designed and conducted to examine, compare and provide baseline observations for how timestamps change in the same *ext4* filesystems between IoT devices and Linux-based systems. The observations, techniques and methodologies discussed in this article contribute to a thorough understanding of how the same *ext4* filesystem timestamps work differently, or perhaps similarly, for Pi and Linux-based systems (e.g., Ubuntu) in cloud environments. Furthermore, this study advances cyber forensics and cyber-criminal investigation for use by the law enforcement and intelligence community.

Declaration of Conflicting Interest

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The authors acknowledge Florida Center for Cybersecurity (FC2), also known as Cyber Florida, for the grant 3910-1007-00-B 07/01/18—06/30/20.

Acknowledgments

The authors thank research assistant Raghav Rathi, and research participants including Alex Coalla, Ryan, Ratkovich, Michael Costello, En-Cih Chang, and En-Chun Kuo. The authors also wish to thank Conrad F. Metcalfe for assistance in editing.

Appendix A. Direct Observations of Timestamp during File Operations

Table 4. *File standalone operations on Raspberry Pi with Raspbian 9.14, Raspberry Pi with UbuntuMATE, and Ubuntu Linux 18.04*

		Raspberry Pi Raspbian 9.14				Raspberry Pi UbuntuMATE				Ubuntu Linux 18.04			
Operation	Command	Create	Modify	Access	Change	Create	Modify	Access	Change	Create	Modify	Access	Change
		cr-time	m-time	a-time	c- time	cr-time	m-time	a-time	c- time	cr-time	m-time	a-time	c- time
1. Create	1.1 nano												
	nano 11.txt	9:46:45	9:46:45	9:46:45	9:46:45	11:16:45	11:16:45	11:16:45	11:16:45	11:46:40	11:46:40	11:46:40	11:46:40
	1.2 echo												
	echo "Hello!">12.txt	9:49:24	9:49:24	9:49:24	9:49:24	11:24:03	11:24:03	11:24:03	11:24:03	11:48:02	11:48:02	11:48:02	11:48:02
	1.3 cat												
	cat>13.txt	9:50:07	9:50:17	9:50:07	9:50:17	11:24:58	11:25:03	11:24:58	11:25:03	11:48:44	11:48:56	11:48:44	11:48:56
2. Compress/Decompress	2.1 tar (compress)												
	Baseline	9:55:53	9:55:53	9:55:53	9:55:53	11:28:13	11:28:13	11:28:13	11:28:13	10:20:44	10:20:44	10:20:44	10:20:44
	New (.tar)	9:56:47	9:56:47	9:56:47	9:56:47	11:30:20	11:30:20	11:30:20	11:30:20	10:22:52	10:22:52	10:22:52	10:22:52
	original(.txt)	9:55:53	9:55:53	9:55:53	9:55:53	11:28:13	11:28:13	11:28:13	11:28:13	10:20:44	10:20:44	10:22:52	10:20:44
	tar cvf 111.tar 11.txt												
	2.2 tar(decompress)												
	Baseline	9:56:47	9:56:47	9:56:47	9:56:47	11:30:20	11:30:20	11:30:20	11:30:20	10:22:52	10:22:52	10:22:52	10:22:52
	New (.txt)	9:59:23	9:55:53	9:59:23	9:59:23	11:32:59	11:28:13	11:32:59	11:32:59	10:25:19	10:20:44	10:25:19	10:25:26
	original(.tar)	9:56:47	9:56:47	9:56:47	9:56:47	11:30:20	11:30:20	11:30:20	11:30:20	10:22:52	10:22:52	10:25:00	10:22:52
	tar xvf 111.tar												
	2.3 gzip (compress)												
	Baseline	10:02:51	10:02:51	10:02:51	10:02:51	11:40:44	11:40:44	11:40:44	11:40:44	10:37:45	10:37:45	10:37:45	10:37:45
	New (.gz)	10:03:38	10:02:51	10:02:51	10:03:38	11:41:39	11:40:44	11:40:44	11:41:39	10:38:26	10:37:45	10:37:45	10:38:26
	original(.txt)	10:02:51	10:03:38	10:02:51	10:03:38	11:40:44	11:41:39	11:40:44	11:41:39	10:37:45	10:38:26	10:38:26	10:38:26
	gzip 12.txt												
	2.4 gzip (decompress)												
	Baseline	10:03:38	10:02:51	10:02:51	10:03:38	11:41:39	11:40:44	11:40:44	11:41:39	10:38:26	10:37:45	10:37:45	10:38:26
	New (.txt)	10:08:36	10:02:51	10:02:51	10:08:36	11:43:49	11:40:44	11:40:44	11:43:49	10:40:24	10:37:45	10:37:45	10:40:24
	original (.txt.gz)	10:03:38	10:08:36	10:02:51	10:08:36	11:41:39	11:43:49	11:40:44	11:43:49	10:38:26	10:40:24	10:40:24	10:40:24
	gzip -d 12.txt.gz												
3. Access without modification, or move	Baseline	10:14:57	10:14:57	10:14:57	10:14:57	11:59:11	11:59:11	11:59:11	11:59:11	11:51:34	11:51:34	11:51:34	11:51:34
	3.1 GUI(click)	10:14:57	10:14:57	10:14:57	10:14:57	11:59:11	11:59:11	11:59:11	11:59:11	11:51:34	11:51:34	11:52:44	11:51:34
	Baseline	10:14:57	10:14:57	10:14:57	10:14:57	11:59:11	11:59:11	11:59:11	11:59:11	11:53:59	11:53:59	11:53:59	11:53:59
	3.2 cat	10:14:57	10:14:57	10:14:57	10:14:57	11:59:11	11:59:11	11:59:11	11:59:11	11:53:59	11:53:59	11:54:36	11:53:59
	cat 13.txt												
4. Modify	4.1 GUI												
	Baseline	10:14:57	10:14:57	10:14:57	10:14:57	12:05:08	12:05:08	12:05:08	12:05:08	13:02:59	13:02:59	13:02:59	13:02:59
	4.1.txt	10:14:57	10:18:44	10:14:57	10:18:44	12:05:51	12:05:51	12:05:51	12:05:51	13:03:37	13:03:37	13:03:37	13:03:37
	4.2 nano												
	Baseline	10:20:10	10:20:10	10:20:10	10:20:10	12:07:20	12:07:20	12:07:20	12:07:20	11:16:13	11:16:13	11:16:13	11:16:13
	nano 42.txt	10:20:10	10:21:03	10:20:10	10:21:03	12:07:20	12:07:48	12:07:20	12:07:48	11:16:13	11:16:50	11:16:38	11:16:50

5. Move	Baseline	10:25:27	10:25:27	10:25:27	10:25:27	13:16:56	13:16:56	13:16:56	13:16:56	11:18:56	11:18:56	11:18:56	11:18:56
	mv 5.txt/home/Downloads	10:25:27	10:25:27	10:25:27	10:27:17	13:16:56	13:16:56	13:16:56	13:17:40	11:18:56	11:18:56	11:18:56	11:20:02
6. Rename	Baseline	10:34:11	10:34:11	10:34:11	10:34:11	13:21:33	13:21:33	13:21:33	13:21:33	12:36:30	12:36:30	12:36:30	12:36:30
	Rename	10:34:11	10:34:11	10:34:11	10:35:07	13:21:33	13:21:33	13:21:33	13:23:04	12:36:30	12:36:30	12:36:30	13:37:32
	Mv 6.txt 6n.txt												
7. Copy	7.1 Copy a file into the same directory												
	Baseline	10:36:47	10:36:47	10:36:47	10:36:47	13:25:01	13:25:01	13:25:01	13:25:01	12:42:05	12:42:05	12:42:05	12:42:05
	cp Copied	10:37:37	10:37:37	10:37:37	10:37:37	13:25:47	13:25:47	13:25:47	13:25:47	12:42:46	12:42:46	12:42:46	12:42:46
	Original	10:36:47	10:36:47	10:36:47	10:36:47	13:25:01	13:25:01	13:25:01	13:25:01	12:42:05	12:42:05	12:42:46	12:42:05
	cp 7.txt 7n.txt												
	7.2 Copy a file into another directory												
	Baseline	10:37:37	10:37:37	10:37:37	10:37:37	13:25:47	13:25:47	13:25:47	13:25:47	12:44:24	12:44:24	12:44:24	12:44:24
	cp Copied	10:39:53	10:39:53	10:39:53	10:39:53	13:26:14	13:26:14	13:26:14	13:26:14	12:45:04	12:45:04	12:45:04	12:45:04
	Original	10:37:37	10:37:37	10:37:37	10:37:37	13:25:47	13:25:47	13:25:47	13:25:47	12:44:24	12:44:24	12:45:04	12:44:24
	cp 72.txt /home												
	7.3 Copy a file into another directory and give it a new name												
	Baseline	10:39:53	10:39:53	10:39:53	10:39:53	13:26:14	13:26:14	13:26:14	13:26:14	12:47:05	12:47:05	12:47:05	12:47:05
	cp Copied	10:42:35	10:42:35	10:42:35	10:42:35	13:13:27	13:13:27	13:13:27	13:13:27	12:48:08	12:48:08	12:48:08	12:48:08
	Original	10:39:53	10:39:53	10:39:53	10:39:53	13:26:14	13:26:14	13:26:14	13:26:14	12:47:05	12:47:05	12:48:08	12:47:05
	cp 73.txt /home/73new.txt												
8. Delete	Baseline	10:44:07	10:44:07	10:44:07	10:44:07	13:29:11	13:29:11	13:29:11	13:29:11	12:52:07	12:52:07	12:52:07	12:52:07
	rm	10:44:07	10:44:52	10:44:07	10:44:52	13:29:11	13:29:11	13:29:11	13:23:36	12:52:07	12:52:39	12:52:07	12:52:39
	rm 8.txt d-time'				10:44:52				13:23:36				12:52:39

* Red color indicates a timestamp change (i.e., difference) recorded and observed—as compared with previous timestamp observation.

Table 5. *File standalone operations on Virtual Ubuntu Linux 18.04 and Virtual Ubuntu Linux 14.04 in Hyper-V Management*

Operation	Command	Virtual Ubuntu Linux 18.04				Virtual Ubuntu Linux 14.04			
		Create cr-time	Modify m-time	Access a-time	Change c- time	Create cr-time	Modify m-time	Access a-time	Change c- time
1. Create	1.1 nano								
	nano 11.txt	9:49:08	9:49:08	9:49:08	9:49:08	13:09:52	13:09:52	13:09:52	13:09:52
	1.2 echo								
	echo "Hello!">12.txt	9:58:15	9:58:15	9:58:15	9:58:15	13:12:25	13:12:25	13:12:25	13:12:25
	1.3 cat								
	cat>13.txt	9:59:20	9:59:28	9:59:20	9:59:28	13:17:42	13:17:59	13:17:42	13:17:59
2.Compress/Decompress	2.1 tar (compress)								
	Baseline	10:01:56	10:01:56	10:01:56	10:01:56	13:21:54	13:21:54	13:21:54	13:21:54
	New (.tar)	10:04:36	10:04:36	10:04:36	10:04:36	13:22:28	13:22:28	13:22:28	13:22:28
	original(.txt)	10:01:56	10:01:56	10:04:36	10:01:56	13:21:54	13:21:54	13:22:28	13:21:54
	tar cvf 111.tar 11.txt								
	2.2 tar(decompress)								
	Baseline	10:04:36	10:04:36	10:04:36	10:04:36	13:22:28	13:22:28	13:22:28	13:22:28
	New (.txt)	10:06:08	10:01:56	10:06:08	10:06:08	13:24:26	13:24:26	13:21:54	13:24:26
	original(.tar)	10:04:36	10:04:36	10:06:03	10:04:36	13:22:28	13:22:28	13:24:26	13:22:28
	tar xvf 111.tar								
	2.3 gzip (compress)								
	Baseline	10:24:51	10:24:51	10:24:51	10:24:51	13:29:34	13:29:34	13:29:34	13:29:34
	New (.gz)	10:25:20	10:24:51	10:24:51	10:25:20	13:30:23	13:29:34	13:29:34	13:30:23
	original(.txt)	10:24:51	10:25:20	10:25:20	10:25:20	13:29:34	13:30:23	13:30:23	13:30:23
	gzip 12.txt								
	2.4 gzip (decompress)								
	Baseline	10:25:20	10:24:51	10:24:51	10:25:20	13:30:23	13:29:34	13:29:34	13:30:23
	New (.txt)	10:29:02	10:24:51	10:24:51	10:29:02	13:33:18	13:29:34	13:29:34	13:33:18
	original(.tar)	10:25:20	10:29:02	10:29:02	10:29:02	13:30:23	13:33:18	13:33:18	13:33:18
	gzip -d 12.txt.gz								
3. Access without	Baseline	10:35:36	10:35:36	10:35:36	10:35:36	13:53:02	13:53:02	13:53:03	13:53:02
modification, or move	3.1 GUI(click)	10:35:36	10:35:36	10:35:58	10:35:36	13:53:02	13:53:02	13:53:03	13:53:02
	3.2 cat								
	Baseline	10:37:13	10:37:13	10:37:13	10:37:13	13:53:02	13:53:02	13:53:03	13:53:02
	cat 13.txt	10:37:13	10:37:13	10:37:36	10:37:13	13:53:02	13:53:02	13:53:03	13:53:02
4. Modify	4.1 GUI								
	Baseline	10:38:46	10:38:46	10:38:46	10:38:46	13:53:02	13:53:02	13:53:03	13:53:02
	4.1.txt	10:39:32	10:39:32	10:39:32	10:39:32	13:55:24	13:55:24	13:55:25	13:55:24
	4.2 nano								
	Baseline	10:41:13	10:41:13	10:41:13	10:41:13	13:58:50	13:58:50	13:58:52	13:58:50
	nano 42.txt	10:41:13	10:41:34	10:41:41	10:41:41	13:58:50	14:01:09	14:01:10	14:01:09
5. Move									
	Baseline	10:46:38	10:46:38	10:46:38	10:46:38	14:04:13	14:04:13	14:04:14	14:04:13
	mv 5.txt/home/admin/Downloads	10:46:38	10:46:38	10:46:38	10:48:09	14:04:13	14:04:13	14:05:17	14:05:16
6. Rename	Baseline	10:49:04	10:49:04	10:49:04	10:49:04	14:08:04	14:08:04	14:08:05	14:08:04
	rename	10:49:04	10:49:04	10:49:04	10:49:29	14:08:04	14:08:04	14:08:50	14:08:49
	Mv 6.txt 6n.txt								

7. Copy	7.1 Copy a file into the same directory								
	Baseline	10:58:05	10:58:05	10:58:05	10:58:05	14:10:26	14:10:26	14:10:27	14:10:26
	cp Copied	10:58:43	10:58:43	10:58:43	10:58:43	14:11:24	14:11:24	14:11:25	14:11:24
	Original	10:58:05	10:58:05	10:58:43	10:58:05	14:10:26	14:10:26	14:10:27	14:10:26
	cp 7.txt 7n.txt								
	7.2 Copy a file into another directory								
	Baseline	11:00:29	11:00:29	11:00:29	11:00:29	14:13:55	14:13:55	14:13:57	14:13:55
	cp Copied	11:01:08	11:01:08	11:01:08	11:01:08	14:14:40	14:14:40	14:14:41	14:14:40
	Original	11:00:29	11:00:29	11:01:08	11:00:29	14:13:55	14:13:55	14:13:57	14:13:55
	cp 42.txt /home								
	7.3 Copy a file into another directory and give it a new name								
	Baseline	11:02:42	11:02:42	11:02:42	11:02:42	14:15:41	14:15:41	14:15:42	14:15:41
	cp Copied	11:03:19	11:03:19	11:03:19	11:03:19	14:17:10	14:17:10	14:17:11	14:17:10
	Original	11:02:42	11:02:42	11:03:19	11:02:42	14:15:41	14:15:41	14:15:42	14:15:41
	cp 43.txt /home/43new.txt								
8. Delete	Baseline	11:04:11	11:04:11	11:04:11	11:04:11	14:25:08	14:25:08	14:25:09	14:25:08
	rm	11:04:11	11:04:52	11:04:11	11:04:52	14:25:08	14:25:08	14:25:09	14:25:38
	rm 8.txt d-time'				11:04:52				14:25:38

* Red color indicates a timestamp change (i.e., difference) recorded and observed—as compared with previous timestamp observation.

Table 6. *File transfer operations and interaction of Raspberry Pi with Private Cloud*

		Raspberry Pi 9.14 with Ubuntu 18.04				Raspberry Pi UbuntuMATE with Ubuntu 18.04			
	Operation	Create cr-time	Modify m-time	Access a-time	Change c-time	Create cr-time	Modify m-time	Access a-time	Change c-time
Send file from Client(pi) to cloud (Linux)	Baseline	10:53:22	10:53:22	10:53:22	10:53:22	13:33:24	13:33:24	13:33:24	13:33:24
‘*scp pi@ipaddress:myfile.txt*’ command is used		10:53:22	10:53:22	10:53:22	10:53:22	13:33:24	13:33:24	13:33:24	13:33:24
Downloading file from cloud (Linux) on Client (Pi)	Baseline	10:53:22	10:53:22	10:53:22	10:53:22	13:33:24	13:33:24	13:33:24	13:33:24
‘*scp pi@ipaddress:myfile.txt*’ command is used		10:53:22	11:00:04	10:53:22	11:00:04	13:33:24	13:35:10	13:33:24	13:35:10

* Red color indicates a timestamp change (i.e., difference) recorded and observed—as compared with previous timestamp observation.

Table 7. *File transfer operations and interaction of Ubuntu 18.04 with Private Cloud*

		Physical Ubuntu 18.04 with Raspberry Pi				Virtual Ubuntu 18.04 with Ubuntu 14.04				Virtual Ubuntu 14.04 with Ubuntu 18.04			
	Operation	Create cr-time	Modify m-time	Access a-time	Change c-time	Create cr-time	Modify m-time	Access a-time	Change c-time	Create cr-time	Modify m-time	Access a-time	Change c-time
Uploading a file from Ubuntu system to private server	baseline	14:31:37	14:31:37	14:31:37	14:31:37	14:47:35	14:47:35	14:47:35	14:47:35	14:36:01	14:36:01	14:36:02	14:36:01
		14:31:37	14:31:37	14:33:29	14:31:37	14:47:35	14:47:35	14:48:51	14:47:35	14:36:01	14:36:01	14:36:02	14:36:01
Downloading a file back to Ubuntu system from a private server	baseline	14:31:37	14:31:37	14:33:29	14:31:37	14:47:35	14:47:35	14:48:51	14:47:35	14:40:36	14:36:01	14:36:02	14:36:01
		14:31:37	14:35:12	14:33:29	14:35:12	14:47:35	14:52:22	14:48:51	14:52:22	14:40:36	14:40:36	14:40:37	14:40:36

* Red color indicates a timestamp change (i.e., difference) recorded and observed—as compared with previous timestamp observation.

Table 8. *File transfer operations and interaction of Raspberry Pi with Public Cloud (ownCloud)*

		Raspberry Pi 9.14 with ownCloud				Raspberry Pi UbuntuMATE with ownCloud			
	Operation	Create cr-time	Modify m-time	Access a-time	Change c-time	Create cr-time	Modify m-time	Access a-time	Change c-time
Send file from Client(pi) to cloud (OwnCloud) using sync	baseline	10:28:51	10:28:51	10:28:51	10:28:51	14:59:50	14:59:50	14:59:50	14:59:50
	upload	10:28:51	10:28:51	10:28:51	10:28:51	14:59:50	14:59:50	14:59:50	14:59:50
using browser	baseline	10:33:08	10:33:08	10:33:08	10:33:08	14:03:43	14:03:43	14:03:43	14:03:43
	upload	10:33:21	10:33:08	10:33:08	10:33:21	14:04:14	14:03:43	14:03:43	14:04:14

* Red color indicates a timestamp change (i.e., difference) recorded and observed—as compared with previous timestamp observation.

Table 9. *File transfer operations and interaction of Ubuntu with Public Cloud (Dropbox)*

		Physical Ubuntu 18.04 with Dropbox				Virtual Ubuntu 18.04 with Dropbox				Virtual Ubuntu 14.04 with Dropbox			
	Operation	Create cr-time	Modify m-time	Access a-time	Change c-time	Create cr-time	Modify m-time	Access a-time	Change c-time	Create cr-time	Modify m-time	Access a-time	Change c-time
Send file from Client(Ubuntu) to cloud (Dropbox) through terminal	baseline	11:02:14	11:02:14	11:02:14	11:02:14	10:58:39	10:58:39	10:58:39	10:58:39	10:47:49	10:47:49	10:47:49	10:47:49
Upload file from Client (Ubuntu) to cloud (Dropbox) through browser	baseline	11:04:10	11:04:10	11:04:10	11:04:10	10:59:37	10:59:37	10:59:37	10:59:37	10:51:30	10:51:30	10:51:31	10:51:30
		11:04:10	11:04:10	10:04:57	10:04:57	10:59:37	10:59:37	10:59:54	10:59:54	10:51:30	10:51:30	10:53:13	10:53:12

* Red color indicates a timestamp change (i.e., difference) recorded and observed—as compared with previous timestamp observation.

Appendix B. Additional Ubuntu Virtual Machine Setup for Comparison Purposes

Table 10. *Computing environments for Ubuntu virtual machines*

Computing Environment	Ubuntu 18.04 O/S Virtual Machine	Ubuntu 14.04 O/S Virtual Machine
Specifications	○ O/S: Ubuntu 18.04 LTS on 64-bit ○ Instruction Set Architecture: x86 ○ Filesystem: ext4 ○ Browser: Firefox 67.0.1 ○ Forensic Tool: Command line *debugfs* ○ File type: Text file (.txt)	○ O/S: Ubuntu 14.04 LTS on 64-bit ○ Instruction Set Architecture: x86 ○ Filesystem: ext4 ○ Browser: Firefox 66.0.3 ○ Forensic Tool: Command line *debugfs* ○ File type: Text file (.txt)
Private Cloud	○ Hyper-V Management Windows Server 2016 Datacenter	○ Hyper-V Management Windows Server 2016 Datacenter
Public Cloud	○ Dropbox	○ Dropbox

Table 11. *File synchronization operations and interaction with the public Cloud (Dropbox)*

File Operation	Ubuntu Linux 18.04 with Dropbox				Ubuntu Linux 18.04 VM with Dropbox				Ubuntu Linux 14.04 VM with Dropbox			
	Create time	Modify time	Access time	Change time	Create time	Modify time	Access time	Change time	Create time	Modify time	Access time	Change time
	cr-time	m-time	a-time	c-time	cr-time	m-time	a-time	c-time	cr-time	m-time	a-time	c-time
Send file from Pi to cloud (ownCloud) using sync	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change	No Change
Send file from Pi to cloud (ownCloud) using browser	No Change	No Change	File upload time	File upload time	No Change	No Change	File upload time	File upload time	No Change	No Change	File upload time	File upload time

References

Casey, E. 2015. "Smart Home Forensics," *Digital Investigation* (13), pp. a1-a2. doi: 10.1016/j.diin.2015.05.017

Chung, H., Park, J., and Lee, S. 2017. "Digital Forensic Approaches for Amazon Alexa Ecosystem," *Digital Investigation* (22:Supplement), pp. S15-S25. doi: 10.1016/j.diin.2017.06.010

Chung, H., Park, J., Lee, S., and Kang, C. 2012. "Digital Forensic Investigation of Cloud Storage Services," *Digital Investigation* (9:2), pp. 81-95. doi: 10.1016/j.diin.2012.05.015

Feng, X., Babatunde, O., and Liu, E. 2017. "Cyber Security Investigation for Raspberry Pi Devices," in: *International Refereed Journal of Engineering and Sciences*. Bedfordshire, UK: University of Bedfordshire Repository, pp. 1-14.

Ho, S. M., Kao, D., and Wu, W.-Y. 2018. "Following the Breadcrumbs: Timestamp Pattern Identification for Cloud Forensics," *Digital Investigation* (24), pp. 79-94. doi: 10.1016/j.diin.2017.12.001

Martini, B., and Choo, K.-K. R. 2012. "An Integrated Conceptual Digital Forensic Framework for Cloud Computing," *Digital Investigation* (9:2), pp. 71-80. doi: 10.1016/j.diin.2012.07.001

Martini, B., and Choo, K.-K. R. 2013. "Cloud Storage Forensics: Owncloud as a Case Study," *Digital Investigation* (10:4), pp. 287-299. doi: 10.1016/j.diin.2013.08.005

Murray, R. 2017. "A Raspberry Pi Attacking Guide," pp. 1-8.

Quick, D., and Choo, K.-K. R. 2013a. "Digital Droplets: Microsoft Skydrive Forensic Data Remnants," *Future Generation Computer Systems* (29:6), pp. 1378-1394. doi: 10.1016/j.future.2013.02.001

Quick, D., and Choo, K.-K. R. 2013b. "Dropbox Analysis: Data Remnants on User Machines," *Digital Investigation* (10:1), pp. 3-18. doi: 10.1016/j.diin.2013.02.003

Quick, D., and Choo, K.-K. R. 2014. "Google Drive: Forensic Analysis of Data Remnants," *Journal of Network and Computer Applications* (40), pp. 179-193. doi: 10.1016/j.jnca.2013.09.016

Quick, D., and Choo, K.-K. R. 2017. "Pervasive Social Networking Forensics: Intelligence and Evidence from Mobile Device Extracts," *Journal of Network and Computer Applications* (86), pp. 24-33. doi: 10.1016/j.jnca.2016.11.018

Quick, D., and Choo, K.-K. R. 2018. "Iot Device Forensics and Data Reduction," *IEEE Access* (6:Special section on Internet-of-Things (IoT) big data trust management), pp. 47566-47574. doi: 10.1109/ACCESS.2018.2867466

Roussev, V., Barreto, A., and Ahmed, I. 2016. "Api-Based Forensic Acquisition of Cloud Drives," *Proceedings of the IFIP International Conference on Digital Forensics: Advances in Digital Forensics XII (DigitalForensics 2016)*, New Delhi, India: Springer, pp. 213-235. doi: 10.1007/978-3-319-46279-0_11

Roussev, V., and McCulley, S. 2016. "Forensic Analysis of Cloud-Native Artifacts," *Digital Investigation* (16:Supplement), pp. S104-S113. doi: 10.1016/j.diin.2016.01.013

Zawoad, S., and Hasan, R. 2015. "Faiot: Towards Building a Forensics Aware Eco System for the Internet of Things," *Proceedings of the 2015 IEEE International Conference on Services Computing (SCC'15)*, New York, NY: IEEE, pp. 279-284. doi: 10.1109/SCC.2015.46