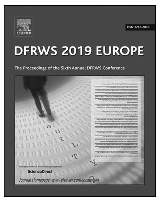
Digital Investigation 28 (2019) S134–S140

Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin



SyncTriage: Using synchronisation artefacts to optimise acquisition order

Christopher Hargreaves [a, *], Angus Marshall [b]

[a] Department of Computer Science, University of Oxford, UK
[b] Department of Computer Science, University of York, UK

ARTICLE INFO

Article history:

Keywords:
Digital forensics
Triage
Synchronisation

ABSTRACT

While the number and variety of devices can be problematic in a digital investigation, it is also a problem for consumers. As a result, software developers have implemented synchronisation features to assist customers handle the multitude of devices that they now use. This paper describes how these synchronisation features can be exploited as part of a digital investigation to use the results from the examination of one device to infer content of other devices. This extracted information is potentially useful in determining the devices that should have the most resources expended during an investigation to obtain the most actionable evidence in the quickest and most efficient manner.

Introduction

Garfinkel (2010) discusses changes in the computer industry that can create challenges for digital forensics. While we are coming to the end of the ten-year period discussed in the paper, challenges such as the growing size of storage devices, the increasing need to analyse and correlate data from multiple devices, and pervasive encryption are still highly relevant. For example, Luck (2016) reported that in 2016 the Metropolitan Police conducted an estimated 49,036 digital forensic examinations. Also, another source (UK Parliament, 2016) describes a 2006 investigation that involved 274 computers and 1785 external storage devices, and Hall (2018) describes the ongoing problem with encrypted evidence.

In order to effectively handle large numbers of cases containing large numbers of exhibits using limited resources it is necessary to prioritise which devices to examine first. Casey et al. (2009) presents three levels of digital forensic examination: *survey/triage forensic inspection*, *preliminary forensic examination*, *in-depth forensic examination*. For the second two approaches, various models of digital forensics can be applied that contain differing levels of detail that extend Carrier (2003): *acquisition*, *analysis*, *presentation*. For the *survey/triage forensic inspection* level, which involves a "targeted review of all available media to determine which items contain the most useful evidence and require additional processing", this process is expanded in Overill et al. (2013) to encompass the following stages: i) pre-seizure: generating a list of anticipated digital devices, ii) search and seizure of devices, iii) post-seizure – screening of the seized devices for the likely existence of relevant evidence in a prioritised manner.

This last stage is described in the literature, for example, Rogers et al. (2006) introduced the Computer Forensics Field Triage Process Model (CFFTPM), which includes stages that include a review of user profile content, a timeline, and an internet artefact review, followed by case specific examination. Variations and extensions of this process are also implemented in a variety of commercial tools that can be used for triage, e.g. *ADF*, *Axiom*, *SPEKTOR* etc.

However, despite recognition of the benefit of a triage stage of digital investigations and known methods for extracting data that provides the best overview of a device to inform triage decisions, there is still a bottleneck in the process. Fig. 1 shows that in order to make a triage decision, data is needed, and therefore first access must be gained to devices, and then some basic extraction of data performed. As the number of devices in a case expands, and given additional protection mechanisms on devices, combined with the volume of data, this approach does not scale.

On initial consideration it appears that this bottleneck is impossible to overcome, but this paper provides an example of where in certain situations there is a work-around to allow a partial "examination" of devices to take place without necessarily needing to gain access to them or to extract any data from the devices themselves. The approach involves exploiting synchronisation

* Corresponding author.
E-mail addresses: chris@hargs.co.uk, christopher.hargreaves@cs.ox.ac.uk (C. Hargreaves), angus.marshall@york.ac.uk (A. Marshall).

https://doi.org/10.1016/j.diin.2019.01.022
1742-2876/

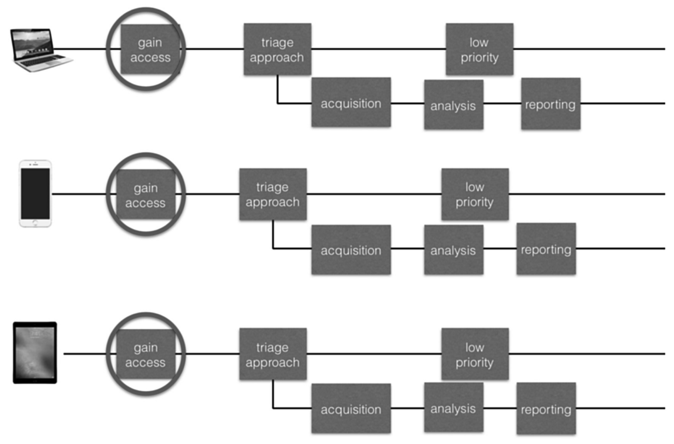


Fig. 1. This shows that existing triage approaches have an inherent bottleneck; at a minimum requires access to the device and inspection of some data, albeit not necessarily a full acquisition.

artefacts to infer the content of devices that have not been accessed, or indeed may not even have been identified and seized.

It should be noted that there is a challenge in reporting this work as it was conducted in 2016. Since then, there have been several papers about synchronisation artefacts that cover some of the components of the work. These are discussed in the related work section, but they are either focused on specific synchronisation artefacts, or problems in digital forensics other than triage. Therefore, despite the work published in this area since 2016, this paper is still able to make the following contributions:

- It provides a summary of some artefacts that can be used to extract synchronisation information,
- It presents and evaluates a new overall approach for inferring the existence of, and partial content of other devices.

The remainder of the paper is structured as follows: section 2 provides a summary of the previous work in this area, section 3 discusses the methodology for the research and sections 4 and 5 presents the results. The work is evaluated in section 6 and section 7 provides the conclusions and further work.

Related work

The need for triage in modern digital investigations and details of triage approaches have already been discussed in the introduction. Therefore, this section focuses on the existing work on synchronisation artefacts.

Several papers discuss cloud-based storage. For example Chung et al. (2012) provides an overall method for investigating such services. Several services are considered including *Amazon S3*, *Dropbox*, *Evernote*, and *Google Docs*. The paper understandably focuses on content, access records, and times of activity, but there is mention of *Evernote* storing references to the type of smartphone OS that created a note. Farina and Kechadi (2014) discusses artefacts left by *BTSync* (now *Resilio Sync*) but there were no artefacts reported that could be used to identify other devices that have synchronised content.

Operating system level synchronisation is also discussed in some previous work, for example, Friedman et al. (2012) discusses *iCloud* data and its synchronisation to *Apple* devices. The work provides information to determine if *iCloud* was enabled on a device, and also identified the same content present on multiple synchronised devices, including the same calendar web addresses, but "there was little evidence showing the two devices were connected to each other through *iCloud*" i.e. it was not possible to identify one device from the other. Further work in this area was reported in a later work Bubbins (2015), which specifically examined the *FindMyiPhone* feature of *iCloud* and was able to retrieve a list of devices connected to the account and their properties e.g. model, battery level etc. This was achieved using *MacOS* cached browser data and data from *iOS* devices prior to iOS 8.3.

Browser synchronisation work includes Wright (2015) which examined the synchronisation artefacts in Google Chrome and provided a means to determine that Chrome Synchronisation was enabled. It also showed that visits to web pages were synchronised across multiple devices, and provided a method to determine which URLs in the history were conducted on another device. It also discusses that the SyncData.sqlite3 database includes references to the other devices included in the synchronisation process. Boucher and Le-Khac (2018) provides a framework to address the problem of determining whether artefacts found on a device really originated on that device or if they were synchronised from somewhere else. Again, the research in the paper supports the existence of synchronisation artefacts, but the focus is very different to the research aim described in this paper since Boucher and Le-Khac (2018) treats synchronisation as a problem for an examination rather than exploiting it to assist investigations.

There are two obvious examples of this approach that are already performed. First, the examination of *iPhone* backups that are stored on a PC. In this case the examination of one device (the PC) results in the indirect examination of another device (the iPhone) that may not necessarily be in the possession of the examiner. The second example is the examination of *Windows* shortcut files and similar artefacts, which if they reference removable storage or network storage, provide information about the content of secondary devices that again, may not have even been identified.

Both of these examples are extremely useful techniques, but the overall concept, which is that data from one device can be used to infer the content of other devices has not been explored in terms of its generalisability as an approach.

Methodology

Overview

The overall aim of the research is to determine if it is possible to mitigate the "gain access" bottleneck of the triage process in order to determine the priority of examination. Fig. 2 illustrates the overall approach, which can be compared with Fig. 1. The method is to gain access to a subset of the total number of devices and focus on extracting artefacts that can be used both to determine the existence of other devices, and also to determine content and events on those other devices. This information can then be used to inform the overall triage process.

To address this aim, the research in this paper was carried out in several stages. Firstly, a review of apps was performed to determine the likely categories of apps that have some sort of synchronisation capability. To achieve this, the 'app categories' on both iOS and Android app stores were examined. A simple feasibility study was conducted, installing several apps from each category and reviewing the features from a user perspective.

Secondly, candidate apps were selected based on the features identified during the feasibility study, and previous work documented in previous work. For each of these apps, digital forensic artefact research was carried out, with a focus on obtaining information about other devices that were part of the synchronisation set. The methodology for this stage is discussed in more detail in

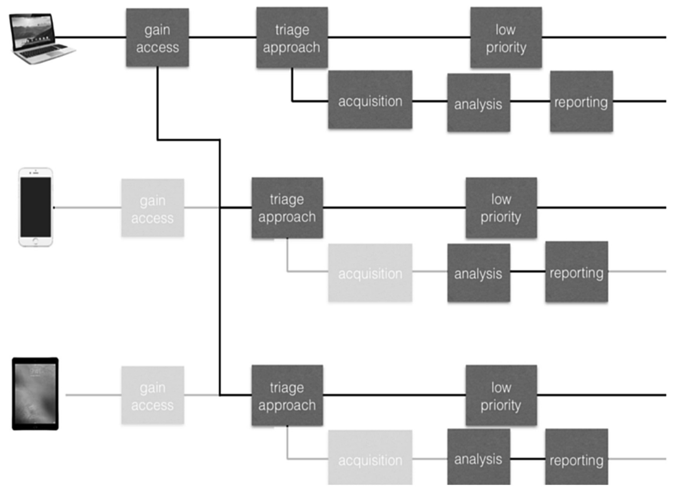


Fig. 2. This illustrates the new approach where access to, and analysis of one device can produce data that can be used to inform the triage process with data about other devices.

Section 3.2
Finally, a software prototype was designed and implemented that practically applied the artefact knowledge identified during the research phase.

Synchronisation artefact research

Experiments were conducted with a range of different apps (discussed in section 4.2). The experiments for each of these followed a typical pattern for digital forensic artefact research: experimental setup, data generation, data analysis.

Experimental setup: Several devices were obtained and set up. These included: iPhone 4S (iOS 9.3), iPhone 4S (iOS9 8.4 jailbroken), Nexus 5 (Android 6.0.2), Vodafone Prime 6 (Android 5.0.2 rooted), PC (Windows 10), Mac (OS X El Capitan). The devices were selected based on availability, and to provide coverage across a variety of operating systems, and for the mobile devices, variations of rooted/jailbroken or standard. The PC/Mac based devices were virtualized using *VMware*.

Data generation: Each app was installed on a subset of the devices, and accounts created. Data was then incrementally added on each device, with device acquisition/imaging taking place after each addition. Imaging of the PC/Mac was achieved by duplicating the VMDK files, and data from the mobile devices was acquired using *Magnet Acquire*. The mobile data was also processed using *pymobilesupport* (Hargreaves, 2016) to export the data into a format that was easier to analyse, e.g. mapping hash-based filenames to their original path on the device. The precise data generated depended on the nature of the application under test, but ranged from web visits for browsers, messages being sent for messaging apps, photographs being taken for photo-based apps, etc. For text-based data generation, unique and easily searchable data was used and test URLs related to the *SyncTriage* project were set up to make keyword searching during the analysis stage more effective.

Data analysis: The data analysis consisted in some cases of manual inspection of all the files within app file system containers, plus known keyword searching. This was mostly performed using *X-Ways Forensics* and various bespoke searching tools, for example to expand *plist* data stored within SQLite database fields, or to interpret *NSKeyedArchiver* formats.

In some cases, several iterations of this data generation/data analysis cycle were performed. For some apps with negative results, only a single iteration was performed, but for apps with complex data formats, additional iterations of this process were performed to confirm the formats, and interpret the data stored.

Results: experimental

App review

The app stores for Android and iOS were examined and lists of app categories extracted. For brevity the app categories are not listed in full here, but the points below discuss some of the categories that were determined to have the most potential in terms of synchronisation artefacts, either from common knowledge of app features, or artefacts discussed in the previous work section.

Browsers: The major browsers now all implement some form of synchronisation, from bookmarks to history. Fig. 3 shows one of the examples from a live device of what *SyncTriage* should aim to recover, where the examination of one device shows the tabs open on other devices.

Communication Apps: Many of the chat applications (*Telegram*, *Hangouts* etc.) are designed such that conversations can be carried out across multiple devices. There are many types of investigation where communication is critical and if possible, knowing the origin device of specific messages may provide insight into which device(s) should be prioritised.

Social Networking: Similar to the communication apps, social media apps present the same content on multiple devices. Either knowing that a specific device was used to share content at a particular time, or even that a device was in general use at a particular time could assist in prioritising the examination of a specific device.

Media & Video: Several apps e.g. *VLC* or *Plex* allow media to be viewed on remote storage on the local network. Certain investigation categories for example, indecent images of children, may benefit from the ability to determine the main source device of such media. This may be particularly true in cases where a network storage device has been physically concealed by the suspect and potentially not recovered during a seizure.

Note taking apps: Notes are used for a variety of purposes, from storing URLs, contacts, to-do lists etc. This app category has been included since if it were possible to identify the source device of a

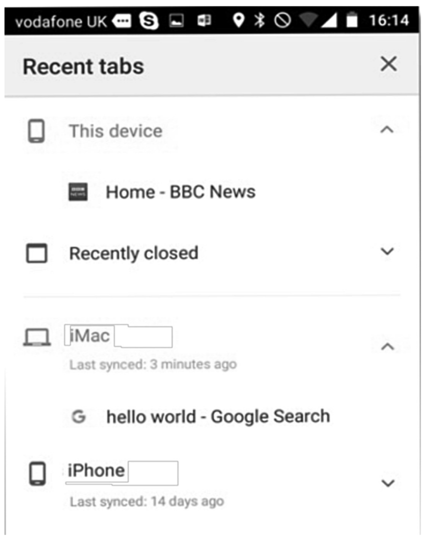

vodafone UK 16:14
Recent tabs
This device
Home - BBC News
Recently closed
iMac
Last synced: 3 minutes ago
hello world - Google Search
iPhone
Last synced: 14 days ago

Fig. 3. An example of early indications of the type of synchronisation data that can be used to infer existence/content of other devices.

note of interest, it may provide insight into what a device was being used for at a specific time. That information could be used to prioritise further examinations of specific devices.

Photos: Photographs are a common media type to be synchronised over multiple devices. In addition, they are known to contain metadata including the device type, dates and times, and potentially geo-location information, on the originating device at least. Despite being a relatively simple and well understood artefact they have not yet been explored in the context of the *SyncTriage* process, i.e. can synchronised copies be used to identify the presence and nature of the devices from which they have originated.

Cloud Storage: Files stored in the cloud that are synchronised to other devices may provide the opportunity to determine the existence of other devices, or if metadata and the origin device is recoverable, may indicate that a specific device was in use at a particular time.

App selection for testing and results

After noting the categories above, considering known features of several common apps, and taking into account previous literature on synchronisation artefacts, the following ‘candidate apps’ were selected for more detailed examination: *Chrome*, *Firefox*, *Facebook Messenger*, *WhatsApp*, *Google Hangouts*, *Telegram*, *Viber*, *Skype*, *VLC*, *YouTube*, *Evernote*, *Google Photos*, *Instagram*, *Facebook*, *Twitter*, and *Dropbox*. This is far from an exhaustive list of applications that showed potential for synchronisation artefact recovery. However, the focus of this paper is to provide a proof of concept of the use of synchronisation artefacts for digital forensic triage, rather than an exhaustive artefact research piece.

There is insufficient space in this paper to provide full details on all the artefact results. Nevertheless, a summary of some of the key results for several of the applications studied are shown in Table 1.

Results: software prototype

Overall design

The overall design of the *sync_triage* tool is a plug-in based framework written in *Python* 3. It currently supports mounted disk images (tested using FTK Imager for mounting), iOS backup folders, and Android ADB backup files. The tool is currently command line only. The overall design of the software is shown in Fig. 4, and each of the stages are discussed in the subsequent sections.

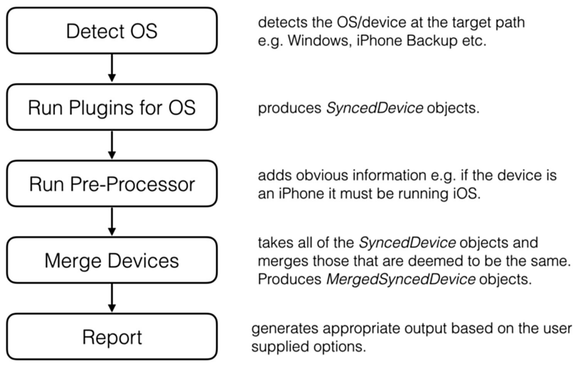

Detect OS
detects the OS/device at the target path e.g. Windows, iPhone Backup etc.
Run Plugins for OS
produces SyncedDevice objects.
Run Pre-Processor
adds obvious information e.g. if the device is an iPhone it must be running iOS.
Merge Devices
takes all of the SyncedDevice objects and merges those that are deemed to be the same. Produces MergedSyncedDevice objects.
Report
generates appropriate output based on the user supplied options.

Fig. 4. This shows the overall flow of the *sync_triage* software.

Operating system specific plugins

At time of writing, plugins have been written for *Windows*, *iOS*, and *Android* in order to demonstrate the feasibility of the approach on multiple platforms. The plugins implemented are listed in Table 2.

All the plugins produce lists of *SyncedDevices*, which include details about the inferred device, but also the provenance, i.e. the source device and forensic artefact from which it was inferred.

Preprocessor

This stage processes the list of devices and applies rules to add in any information that can be obviously inferred. For example, if the device is an *iPhone*, then the operating system is known to be iOS, even if the specific version of the operating system is not known. An example of before and after processing is shown in Fig. 5. Currently, these are hard coded rules and therefore need to be manually updated as new devices and operating systems are released. If information is inferred from external general knowledge, rather than taken directly from information found in the analysed data, then the value is enclosed in square brackets in order to distinguish it.

Merging devices

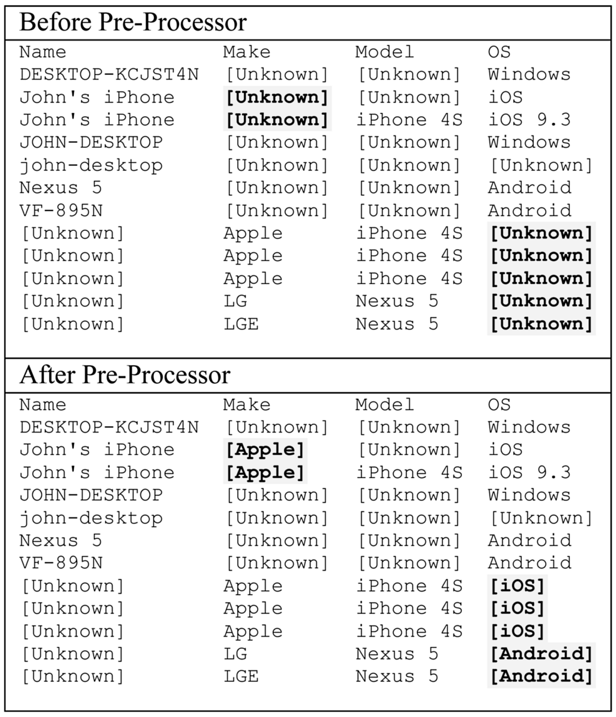
Once the pre-processing has been completed it is necessary to de-duplicate the results. The reason duplicates exist is that plugins simply extract device information from the various sources in the data being processed, and report them to the main program. As a result, a device which runs several apps that independently

Table 1
A summary of some of the artefacts recovered for some of the applications examined.

Application	Key Results
Chrome (Windows)	• Device names of linked synchronised devices • A subset of URLs visited on those devices
Windows 10 Mail (Windows)	• Notifications of service use on other devices can be extracted from emails stored on disk.
Evernote (Windows)	• The type of device used to create each note can be recovered, along with note creation time, and possibly GPS location.
Dropbox (Windows)	• EXIF data from synced photos reveal the make and model of devices, along with times of activity and possible GPS data,
Firefox (Windows)	• Open tabs on other devices
Firefox (iOS)	• List of synced devices, their names, types and operating system • Open tabs on those devices
Google Photos (Android)	• Information about photos stored on remote devices and the Google's servers. Provides make and model of other devices as well as timestamps of usage and possibly GPS data.
iCloud Sync (Windows)	• Using EXIF scanning: make, model, timestamps and possibly GPS data. • Using client.db: times that an iOS device was used to take pictures.
Skype (Windows)	• Content from other devices is recoverable, but it is not possible to determine the origin device.
Messages (iOS)	• Contains authorisation codes from service providers so the use of an app can be determined, although the device in use cannot be determined.

Table 2
A list of plugins currently implemented in the *sync_triage* prototype.

Target Device	Plugins Implemented
Windows	chrome – *extracts devices from SyncData.sqlite3.* dropbox_photos – *scans exif data in photos for devices.* email_scanner – *scans emails for "account in use on device" emails.* evernote – *scans note_attr table for references to devices.* icloud_photos – *scans client.db for photos in server_items table.* viber – *extracts phone number of device used for service.*
Mac OS	–
iOS	chrome – *extracts devices from SyncData.sqlite3.* firefox – *scans browser.db for clients.* sms – *searches for references to service authentication messages.*
Android	google_photos – *extracts exif data from remote_media table.*



```
Before Pre-Processor
Name              Make       Model       OS
DESKTOP-KCJST4N   [Unknown]  [Unknown]   Windows
John's iPhone     [Unknown]  [Unknown]   iOS
John's iPhone     [Unknown]  iPhone 4S   iOS 9.3
JOHN-DESKTOP      [Unknown]  [Unknown]   Windows
john-desktop      [Unknown]  [Unknown]   [Unknown]
Nexus 5           [Unknown]  [Unknown]   Android
VF-895N           [Unknown]  [Unknown]   Android
[Unknown]         Apple      iPhone 4S   [Unknown]
[Unknown]         Apple      iPhone 4S   [Unknown]
[Unknown]         Apple      iPhone 4S   [Unknown]
[Unknown]         LG         Nexus 5     [Unknown]
[Unknown]         LGE        Nexus 5     [Unknown]

After Pre-Processor
Name              Make       Model       OS
DESKTOP-KCJST4N   [Unknown]  [Unknown]   Windows
John's iPhone     [Apple]    [Unknown]   iOS
John's iPhone     [Apple]    iPhone 4S   iOS 9.3
JOHN-DESKTOP      [Unknown]  [Unknown]   Windows
john-desktop      [Unknown]  [Unknown]   [Unknown]
Nexus 5           [Unknown]  [Unknown]   Android
VF-895N           [Unknown]  [Unknown]   Android
[Unknown]         Apple      iPhone 4S   [iOS]
[Unknown]         Apple      iPhone 4S   [iOS]
[Unknown]         Apple      iPhone 4S   [iOS]
[Unknown]         LG         Nexus 5     [Android]
[Unknown]         LGE        Nexus 5     [Android]
```

Fig. 5. An example of the pre-processing of results, showing before and after inferencing of missing data.

synchronise will be reported by multiple plugins. The deduplication process attempts to eliminate all obvious duplications of a device.

Devices are merged as a result of several relatively simplistic rules, for example:

- Merge if has the same name
- Merge if same make and model

When merging does occur, the rules used are preserved in the log file and the individual devices that were combined are recorded within the newly created merged device so that full provenance of results can be inspected. This can be seen in the 'Refs' column in Fig. 6. These are very simple rules at present, but in future, more complex logic could be substituted in for this phase.

Reporting

The results shown earlier in Fig. 6 demonstrated the default output from the tool. There are two other modes that provide more details. The –details option displays additional information about the discovered devices, shown in Fig. 8. You can see that in addition to the name, make, model and operating system that was shown in the summary view, much more information has been recovered. You can see software that is known to have been installed on the device, information about web visits conducted on the device, a basic timeline of activity (in this case just reporting pictures taken, time and location). The details view also reports the original *synced_device* objects that were merged to infer the existence and information about this device, which in turn provide the original file path from which that information was extracted.

The other display option that has been implemented is the 'Universal Timeline' view, invoked with the –timeline option and shown in Fig. 7. This extracts the events from each inferred device and presents them all in a timeline. The use case for this feature is to assist with decision making about which device to examine, particularly in cases where the time of the alleged offence is known. It may be possible to identify the device that was in use closest to the time of the incident.

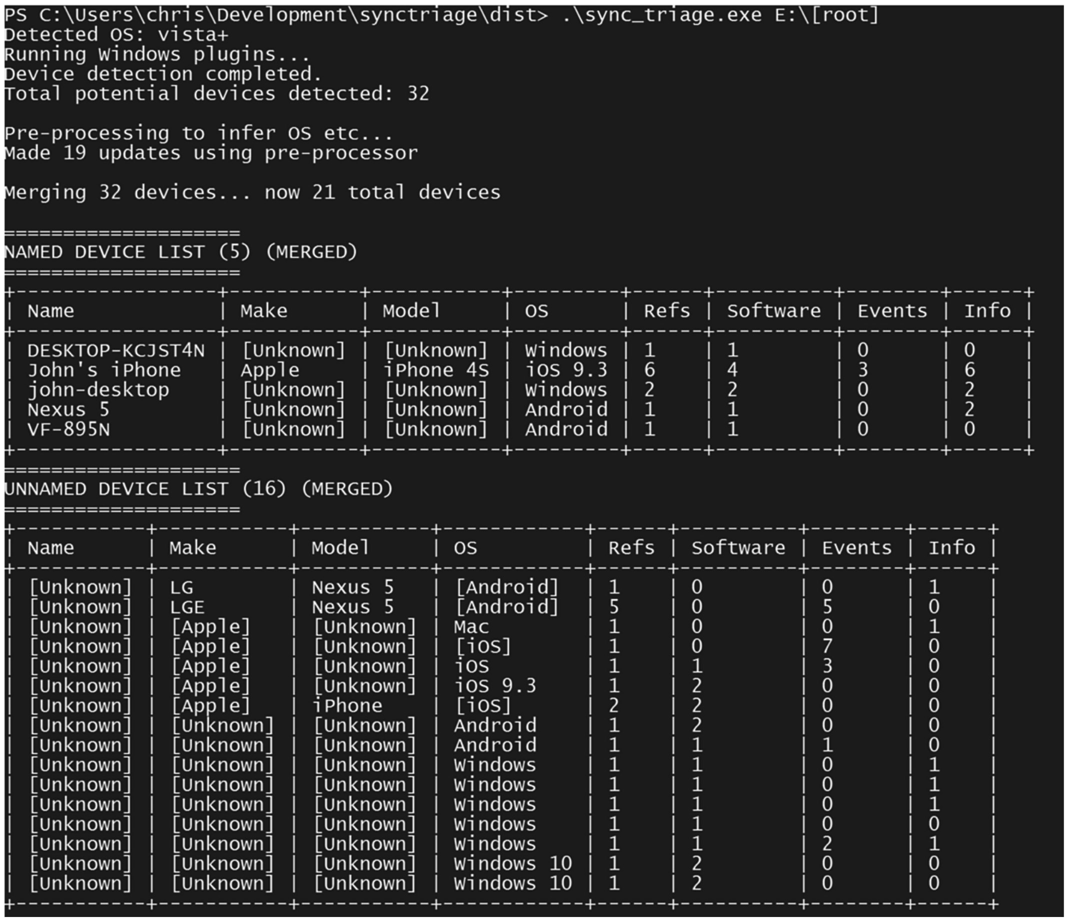
Evaluation

Overall the *SyncTriage* has been a successful proof of concept. In Fig. 8, many details can be seen about the use of an iPhone that has not been accessed at all. Also, in Fig. 7, only 2 of the 21 timeline events shown occurred on the device that is being examined.

In terms of use cases for this approach, *SyncTriage* should help with: detecting devices that have not been seized, determining which device was in use at the time of an offence, inferring content on devices that have not yet been forensically processed. All of these use cases ultimately will help in prioritising the devices to examine first and retrieve actionable evidence as quickly and efficiently as possible.

There are however limitations to this research. For example, the review of apps was far from systematic, although as a proof of concept piece of work, this is not a major concern as the apps selected have allowed the approach to be demonstrated. However, what it does not show is the scale of the effectiveness of the approach. For example, a number of plugins have been produced for a *Windows* examination, but far fewer for *Android* and *iOS*. Even in the case of the existing plugins they are likely to be highly sensitive to the version of the application or operating system. This was the rationale of the plugin-based architecture, but that does not reduce the overhead of conducting the research and software development to keep this artefact extraction and processing up-to-date.

In terms of performance, the program runs on the sample *Windows 10* image and an example 'real world' system in less than a second, since it precisely targets specific artefacts. This could be further cut down as the program is currently single threaded, but



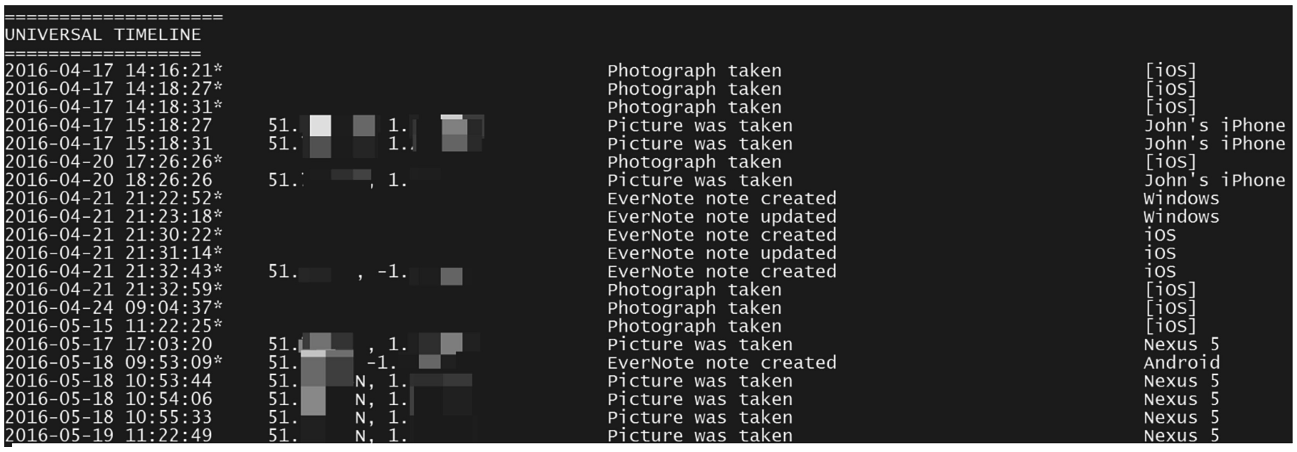
```
PS C:\Users\chris\Development\synctriage\dist> .\sync_triage.exe E:\[root]
Detected OS: vista+
Running Windows plugins...
Device detection completed.
Total potential devices detected: 32

Pre-processing to infer OS etc...
Made 19 updates using pre-processor

Merging 32 devices... now 21 total devices

====================
NAMED DEVICE LIST (5) (MERGED)
====================
+-----------------+-----------+-----------+---------+------+----------+--------+------+
| Name            | Make      | Model     | OS      | Refs | Software | Events | Info |
+-----------------+-----------+-----------+---------+------+----------+--------+------+
| DESKTOP-KCJST4N | [Unknown] | [Unknown] | Windows | 1    | 1        | 0      | 0    |
| John's iPhone   | Apple     | iPhone 4S | iOS 9.3 | 6    | 4        | 3      | 6    |
| john-desktop    | [Unknown] | [Unknown] | Windows | 2    | 2        | 0      | 2    |
| Nexus 5         | [Unknown] | [Unknown] | Android | 1    | 1        | 0      | 2    |
| VF-895N         | [Unknown] | [Unknown] | Android | 1    | 1        | 0      | 0    |
+-----------------+-----------+-----------+---------+------+----------+--------+------+
====================
UNNAMED DEVICE LIST (16) (MERGED)
====================
+-----------+-----------+-----------+------------+------+----------+--------+------+
| Name      | Make      | Model     | OS         | Refs | Software | Events | Info |
+-----------+-----------+-----------+------------+------+----------+--------+------+
| [Unknown] | LG        | Nexus 5   | [Android]  | 1    | 0        | 0      | 1    |
| [Unknown] | LGE       | Nexus 5   | [Android]  | 5    | 0        | 5      | 0    |
| [Unknown] | [Apple]   | [Unknown] | Mac        | 1    | 0        | 0      | 1    |
| [Unknown] | [Apple]   | [Unknown] | [iOS]      | 1    | 0        | 7      | 0    |
| [Unknown] | [Apple]   | [Unknown] | iOS        | 1    | 1        | 3      | 0    |
| [Unknown] | [Apple]   | [Unknown] | iOS 9.3    | 1    | 2        | 0      | 0    |
| [Unknown] | [Apple]   | iPhone    | [iOS]      | 2    | 2        | 0      | 0    |
| [Unknown] | [Unknown] | [Unknown] | Android    | 1    | 2        | 0      | 0    |
| [Unknown] | [Unknown] | [Unknown] | Android    | 1    | 1        | 1      | 0    |
| [Unknown] | [Unknown] | [Unknown] | Windows    | 1    | 1        | 0      | 1    |
| [Unknown] | [Unknown] | [Unknown] | Windows    | 1    | 1        | 0      | 1    |
| [Unknown] | [Unknown] | [Unknown] | Windows    | 1    | 1        | 0      | 1    |
| [Unknown] | [Unknown] | [Unknown] | Windows    | 1    | 1        | 0      | 0    |
| [Unknown] | [Unknown] | [Unknown] | Windows    | 1    | 1        | 2      | 1    |
| [Unknown] | [Unknown] | [Unknown] | Windows 10 | 1    | 2        | 0      | 0    |
| [Unknown] | [Unknown] | [Unknown] | Windows 10 | 1    | 2        | 0      | 0    |
+-----------+-----------+-----------+------------+------+----------+--------+------+
```

Fig. 6. Example output from *sync_triage* after examining a disk image, showing four other devices detected and 16 other entries that could not be automatically merged.



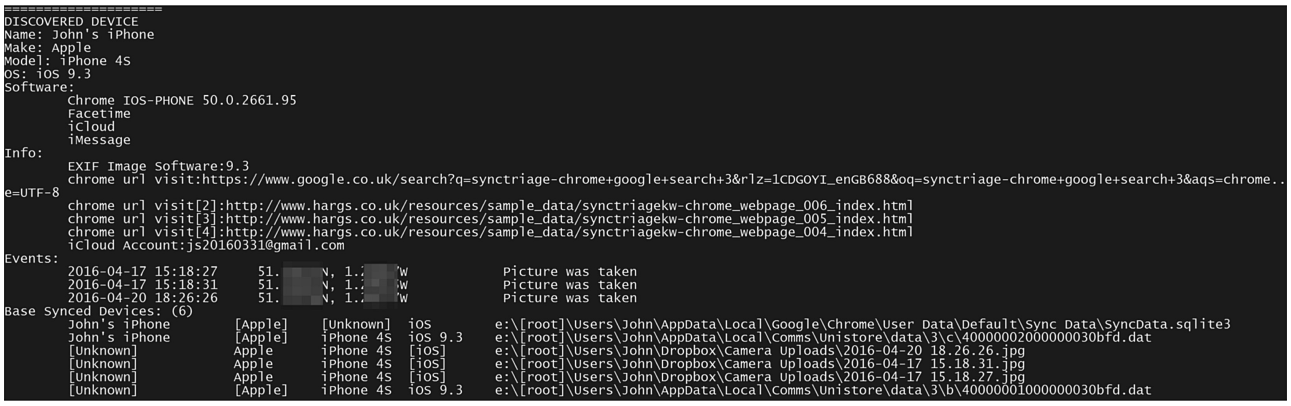
```
====================
UNIVERSAL TIMELINE
==================
2016-04-17 14:16:21*                                       Photograph taken            [iOS]
2016-04-17 14:18:27*                                       Photograph taken            [iOS]
2016-04-17 14:18:31*                                       Photograph taken            [iOS]
2016-04-17 15:18:27    51.[redacted] 1.[redacted]          Picture was taken           John's iPhone
2016-04-17 15:18:31    51.[redacted] 1.[redacted]          Picture was taken           John's iPhone
2016-04-20 17:26:26*                                       Photograph taken            [iOS]
2016-04-20 18:26:26    51.[redacted], 1.[redacted]         Picture was taken           John's iPhone
2016-04-21 21:22:52*                                       EverNote note created       Windows
2016-04-21 21:23:18*                                       EverNote note updated       Windows
2016-04-21 21:30:22*                                       EverNote note created       iOS
2016-04-21 21:31:14*                                       EverNote note updated       iOS
2016-04-21 21:32:43*   51.[redacted], -1.[redacted]        EverNote note created       iOS
2016-04-21 21:32:59*                                       Photograph taken            [iOS]
2016-04-24 09:04:37*                                       Photograph taken            [iOS]
2016-05-15 11:22:25*                                       Photograph taken            [iOS]
2016-05-17 17:03:20    51.[redacted], 1.[redacted]         Picture was taken           Nexus 5
2016-05-18 09:53:09*   51.[redacted] -1.[redacted]         EverNote note created       Android
2016-05-18 10:53:44    51.[redacted] N, 1.[redacted]       Picture was taken           Nexus 5
2016-05-18 10:54:06    51.[redacted] N, 1.[redacted]       Picture was taken           Nexus 5
2016-05-18 10:55:33    51.[redacted] N, 1.[redacted]       Picture was taken           Nexus 5
2016-05-19 11:22:49    51.[redacted] N, 1.[redacted]       Picture was taken           Nexus 5
```

Fig. 7. This shows the 'universal timeline' view of *sync_triage* which combines all of the identified events from all the extrapolated devices into a single timeline. This has significant potential if the time of an offense is known and it can be correlated with a particular device being in use at that time.

with the current runtimes this is not a priority.

It has been shown that a significant amount of automation is possible, certainly for low level device detail extraction, for some inference of missing information, and the merging of some devices. However, even with enhancements to the merging process there will be limits to what is possible for an automated process to do. Therefore, part of the development of this approach must include a more interactive user interface that allows the devices to be explored, manually merged, and the automatic merging reapplied in light of the user suppled information.

Nevertheless, more automation in the merging process may be possible, including information from the events, e.g. if an event is recorded for an unnamed *iOS* device, and an event was recorded for a named *iOS* device at a very similar time, then this could be



```
====================
DISCOVERED DEVICE
Name: John's iPhone
Make: Apple
Model: iPhone 4S
OS: iOS 9.3
Software:
        Chrome IOS-PHONE 50.0.2661.95
        Facetime
        iCloud
        iMessage
Info:
        EXIF Image Software:9.3
        chrome url visit:https://www.google.co.uk/search?q=synctriage-chrome+google+search+3&rlz=1CDGOYI_enGB688&oq=synctriage-chrome+google+search+3&aqs=chrome..
e=UTF-8
        chrome url visit[2]:http://www.hargs.co.uk/resources/sample_data/synctriagekw-chrome_webpage_006_index.html
        chrome url visit[3]:http://www.hargs.co.uk/resources/sample_data/synctriagekw-chrome_webpage_005_index.html
        chrome url visit[4]:http://www.hargs.co.uk/resources/sample_data/synctriagekw-chrome_webpage_004_index.html
        iCloud Account:js20160331@gmail.com
Events:
        2016-04-17 15:18:27     51.[redacted]N, 1.[redacted]W          Picture was taken
        2016-04-17 15:18:31     51.[redacted]N, 1.[redacted]W          Picture was taken
        2016-04-20 18:26:26     51.[redacted]N, 1.[redacted]W          Picture was taken
Base Synced Devices: (6)
        John's iPhone       [Apple]    [Unknown]  iOS       e:\[root]\Users\John\AppData\Local\Google\Chrome\User Data\Default\Sync Data\SyncData.sqlite3
        John's iPhone       [Apple]    iPhone 4S  iOS 9.3   e:\[root]\Users\John\AppData\Local\Comms\Unistore\data\3\c\40000002000000030bfd.dat
        [Unknown]           Apple      iPhone 4S  [iOS]     e:\[root]\Users\John\Dropbox\Camera Uploads\2016-04-20 18.26.26.jpg
        [Unknown]           Apple      iPhone 4S  [iOS]     e:\[root]\Users\John\Dropbox\Camera Uploads\2016-04-17 15.18.31.jpg
        [Unknown]           Apple      iPhone 4S  [iOS]     e:\[root]\Users\John\Dropbox\Camera Uploads\2016-04-17 15.18.27.jpg
        [Unknown]           [Apple]    iPhone 4S  iOS 9.3   e:\[root]\Users\John\AppData\Local\Comms\Unistore\data\3\b\40000001000000030bfd.dat
```

Fig. 8. This shows the 'details' view of an inferred device. It can be seen that an iPhone 4S exists running iOS 9.3, that it is known to contain several apps, and was in use at specific times in 2016, and visited several websites using *Google Chrome*.

considered to be a 'session' and the devices could be merged.

Furthermore, the events/timeline feature would benefit from expansion, so that the times that particular devices were in used can be more easily and reliably determined.

Finally, at present, this tool analyses only one device at a time. It would be beneficial for additional devices to be added to the set from different sources as this is needed to explore the idea of 'acquisition order optimisation'.

Conclusions and future work

This research has tested the concept of exploiting synchronisation artefacts on one device to extrapolate the existence and content of other devices for the purposes of digital forensic triage. The approach shows promise and further work involves expanding the range of plugins to test the extent to which artefacts exist that can be used for this device inference. There is also additional work to do on the concept of merging the inferred devices in a more sophisticated manner, or providing a user interface that allows the investigator to easily manually merge devices together. Finally, process-based research also needs to be conducted on how this approach can be integrated into digital forensic workflows and used to improve the acquisition order of devices.

Acknowledgements

This work was funded under a Home Office Innovation call through SBRI/Innovate UK.

References

Boucher, Le-Khac, 2018. Forensic framework to identify local vs synced artefacts. Digit. Invest. 24 (Suppl.), 68–75.

Bubbins, 2015. Identification of Devices Connected to a Suspect's iCloud Account when Using the Application Find My iPhone. MSc Thesis. Cranfield University.

Carrier, 2003. Defining digital forensic examination and analysis tools using abstraction layers. International Journal of Digital Evidence 1 (4).

Casey, et al., 2009. Investigation Delayed Is Justice Denied: Proposals for Expediting Forensic Examinations of Digital Evidence.

Chung, H., Park, J., Lee, S., Kang, C., November 2012. Digital forensic investigation of cloud storage services. Dig. Invest. 9 (2), 81–95.

Farina, Scanlon, Kechadi, 2014. BitTorrent sync: first impressions and digital forensic implications. Digit. Invest. 11 (Suppl. 1), 77–86.

Friedman, Brunty, Fenger, 2012. A Digital Forensic Analysis on the iCloud® and its Synchronization to Apple® Devices. http://www.marshall.edu/forensics/files/FRIEDMANRACHEL-Research-Paper-08242012.pdf. (Accessed 23 October 2018).

Garfinkel, 2010. Digital Forensics Research: The Next 10 Years. DFRWS 2010.

Hall, 2018. The Reg Visits London Met Police's Digital and Electronics Forensics Labs. The Resister. https://www.theregister.co.uk/2018/01/22/digital_forensics/. (Accessed 23 October 2018).

Hargreaves, 2016. Pymobilesupport. https://bitbucket.org/chrishargreaves/pymobilesupport.

Luck, 2016. Challenges and Opportunities for Statistics in Digital Forensics. https://www.turing-gateway.cam.ac.uk/sites/default/files/asset/doc/1612/Luck.pdf. (Accessed 28 October 2018).

Overill, Silomon, Roscoe, 2013. Triage template pipelines in digital forensic investigations. Digit. Invest. 10 (2), 168–174.

Rogers, 2006. Computer Forensics Field Triage Process Model. J. Digital Forensics Secur. Law 1 (2).

UK Government, 2016. Annex: Visits to the Metropolitan Police Forensic Services and LGC Forensics. https://publications.parliament.uk/pa/cm201617/cmselect/cmsctech/501/50110.htm. (Accessed 23 October 2018).

Wright, 2015. Forensic Artefacts Related to Google's Chrome Synchronisation Feature. MSc Thesis. Cranfield University.