Investigation and Automating Extraction of Thumbnails Produced by Image viewers

Wybren van der Meer
School of Computer Science
University College Dublin,
Belfield, Dublin 4, Ireland
wybren.van-der-meer@ucdconnect.ie

Kim-Kwang Raymond Choo
Department of Information Systems,
University of Texas at San Antonio,
San Antonio, TX 78258, USA
raymond.choo@fulbrightmail.org

Nhien-An Le-Khac, M-Tahar Kechadi
School of Computer Science
University College Dublin,
Belfield, Dublin 4, Ireland
{an.lekhac,tahar.kechadi}@ucd.ie

***Abstract*— Data carving is generally used to recover deleted images in digital investigations, but carving time can be significant and the deleted images may have been overwritten. Thus, thumbnails of (deleted) images are an alternative evidence, and can often be found within databases created by either operating systems or image viewers. Existing literature generally focus on the extraction of thumbnails from databases created by the operating system. Understanding thumbnails created by image reviewers is relatively understudied. Therefore, in this paper, we propose a new approach of automating extraction of thumbnails produced by image viewers. We then evaluate the utility of our approach using popular image viewers.**

Keywords—thumbnails; image acquisition; thumbnail forensics; automate process;

I. INTRODUCTION

Images are a common source of evidence in a typical digital investigation, such as those involving child abusive materials (CAM) [1]. Images once downloaded to a device are usually viewed using image viewers. Standard image viewers in operating system do not provide a way to effectively organize and categorize image collections. To effectively organize image collections, the users will need to use specialized image viewers such as Xnview [2] and ACDsee [3].

To effectively organize and categorize the images, image viewers normally save information about the images (also known as image meta-data). The image meta-data may contain thumbnails, dates, EXIF information and other information of forensic interest [4]. The image meta-data is saved in databases, which reside on the hard disk of the machine with the installed image viewer. These databases may also contain meta-data information of images no longer available (e.g. images can be deleted, or reside on other sources that are not available to access at that time. Eventually, when the source image is not available, the databases of image meta-data from image viewers are the only important sources of information available to investigators.

The image meta-data we look at in this paper is the thumbnails. In the literature, most research and practical focus on the extraction of thumbnails from databases created by the operating system. There is little research focusing on thumbnails created by image reviewers due to many challenges. The first challenge is the storage locations for the databases containing the thumbnails differ between image viewer [5], and there are many image viewers in the market [6]. When a database containing thumbnails is located, another challenge is to extract available relevant information. These databases often have their own way of storing thumbnails. Consequently, traditional ways of extracting information could not be applied. One traditional way of extracting information is to use data carving. Carving tools take into account sector size of a hard disk [7]. When a carving tool finds an image, the tool often skips a certain amount of disk space. The skipping of a certain amount of disk space is performed because partitions on storage devices have a dedicated minimal storage space for each file. Hence, the skipping of this minimal storage space of each file is done to make the carving process faster. Thumbnails within a database can however, ignore these minimal storage spaces, as the database itself is one file. A carving program does not know when there are multiple thumbnails within one file and automatically skips the minimal storage space, a space where other thumbnails may reside.

A scan of the literature suggests that investigating image databases is a relatively unexplored practice. Most information found was related to current carving methods, databases and thumbnails in general. Besides, the large number of thumbnails stored in databases is also a challenge in terms of the amount of time required in the investigation. This is referred to as the big forensic data challenge by Quick and Choo [8][9][10][11].

Therefore, in this paper, we propose a new approach to locate and extract thumbnails from image viewers. First, this approach is to locate the image databases by examining the image viewers and then to search for changes made on the hard drive. The changes provided clues to the location of the image databases. When the image database is found, we then extract the thumbnails using different techniques described in this paper, based on the three different storage structures observed. Another contribution of this paper is to automate the entire proposed process of investigation, using freely available tools and code to reduce the time needed for the forensic investigation.

The rest of this paper is organized as follow: Section II reviews related literature. We present the problem statement of investigating the thumbnails from image viewers in Section III. We describe our approach in Section IV. Findings are

presented in Section V. Finally, we conclude and discuss future work in Section VI.

II. Releated work

With the increasing size of images due to advances in image sensors and capturing capabilities, thumbnails are often used to expedite the viewing of images and minimize data usage [12]. Thumbnails are often created before a picture has been manipulated. The original picture can then be altered, and the thumbnail associated to an image would then be different compared to the new altered image [13]. Thumbnails also offer a useful source of information about an image if the original is no longer available. One good way of investigating an image is to reverse search the image. In this manner, the source of the original image can be tracked down [14]. The reverse searching can also be performed with just the thumbnail.

For example, Quick, Tassone and Choo conducted a forensic analysis of Windows *thumbcache* files [15]. The research provides a blueprint on how to design an automated tool for investigated thumbnails. It does, however, focus only on thumbnails created by the different Windows operating systems. Thumbnails made by individual programs were not investigated.

In [16], the authors focus on when Windows thumbnails are created. The authors conclude that sometimes thumbnails are even produced when the user does not browse related pictures. The actually viewing of a picture is a parameter and has been taken into account during the choice of the method to investigate thumbnails generated by image viewers.

Thumbnails produced by Android were also previously investigated [17]. In this research, the authors focus on extracting thumbnails and the file information of the original files from the main *imgcache* file found within an Android installation. The authors were successful in extracting this information and finding out how the information was stored. Also, the EXIF information is stored within the found thumbnails. Their findings revealed that thumbnails remained in the *imgcache* file, even after the original files were deleted.

Databases are an important source of information to conduct a forensic investigation [18] [19] [20]. Indeed, SQLite database is used a lot in mobile operating systems present on smartphones [21]. In [22], the main database "*index.sqlite*" used for thumbnails in OS X Mountain Lion was investigated. Authors conducted tests to see which actions led to changes in this "*index.sqlite*". Findings were checked by browsing pictures on a new account on a newly installed machine. This check also led to the conclusion that the database was not removed when a user was removed from the system. The fact a database is able to store different kinds of data and retrieve this information makes it an ideal choice to store thumbnails, and information about the original pictures.

The authors in [23] describe an effort is made to bring the different fragmented parts of a carved image together. With this method it would be possible to carve complete images, even if the image is stored fragmented. Authors used a technique which checks for different patterns in JPG compression. If the pattern in the end of one fragment matched the pattern in the beginning of one other fragment, statistical tests were conducted in order to validate if the fragments should be stitched together.

Recovering data from databases may not be possible with carving, as the structure may be different. It is possible to retrieve data from databases by other means. In the study of [24], some work has been done on recovering data from DBMS structures. Another subject to consider is compressed data within databases. There are methods for extracting the compressed streams, however, errors may occur [25].

III. Problem Statement

When the original images are no longer available, and cannot be retrieved by carving files, one has to look for different sources of information. One source may be forensic artefacts created when an image is viewed. An image can be browsed and viewed by the file explorer of an operating system. This may lead to the creation of a thumbnail. Also metadata of the file may be stored. The retrieving of thumbnails and metadata from artefacts within the file explorer of an operating system has been researched and will remain an ongoing research area, due to constant updates in operating systems. Information is however not stored within the database of the operating system if the image was viewed by other means. When an image is viewed by a specialized image viewer, forensics artefacts may be created by the viewer itself. The method used for storing the information by a viewer may differ between different brands, and can change with each new version. The viewer, like the operating system, can store information within a database. The storing of the information in a database has different purposes. One main consideration for storing thumbnails is the speed increase for showing many pictures within a gallery mode.

Traditional forensic processes may be limited in recovering all thumbnails from the database. To be able to recover information from the databases containing images, the structure of these databases must be investigated. When the structure is found, the information needed can be extracted. Without a specialized tool for this, the extraction of information and thumbnails remains a manual task for the digital forensic investigator. This is a laborious task which needs many different tools, and many different steps. Considering the steps needed, investigating the databases produced by image viewers is time consuming and resource intensive. In other words, databases may not always be investigated, due to the time and effort required. This can lead to useful information not being found. Streamlining the process of extracting the thumbnails and other information would mean the investigation would take less time. Taking less time, the investigation may be done more often. This could lead to useful information found more often.

Automating the different steps needed within the investigation would be a step towards streamlining the process. For this to be possible, the different steps within the investigation must first be investigated. Hence, we seek to answering the following research questions: (i) How are the images stored within the databases for popular image viewing software? (ii) What methods can be applied to identify database structures containing images? (iii) If there are images

present, what is the best way to extract all of these images, without skipping any? (iv) How does a traditional forensic tool like Encase deal with these databases? With the answers of these questions clear, effort can be made to write an automated script, which will save time on investigating information stored by image viewers.

IV. Proposed Approach

There are two main stages in our proposed approach. First, it is the identification of storage location. This means we identify how different image viewers store the images and their thumbnails. The second stage is dealing with database structure containing images. Indeed, we have developed an automate method to identify the structure of the databases containing images, and to extract the identified images from the databases. Eventually, to investigate the storage methods of the image viewers, we apply the following steps: (i) Install different image viewers; (ii) Using image viewer to open and browse images stored in external device and monitoring these activities.

By having the images on an external device, and by only browsing the images with the image viewers, effort was made to reduce the traces by other means than the image viewers. By accessing images from an external drive, effort was made to come as close to real life usage of such an image viewer as possible. Besides, the FolderChangesView tool [26] is used to log the changes to the hard drive. The monitoring of changes during the installing of the viewers, and during the viewing of the images gave information about which files were written to.

By first investigating the files that were changed most often, effort was made to find the databases. Also the location of the files that had changes was a factor that was looked at. Combining these factors lead to files that had the most change of being a database. Monitoring the file changes on the live system had the advantage of being able to look at the results in a real time fashion. In this way the file changes occurring exactly during the image browsing could be more easily determined. After the first stage, suspect files are extracted. Next, in the second stage, we examine the suspect files to locate images stored. By looking at the header and other plain text contents of the file, sometimes the structure of the database could be determined. If the type of file could not be determined this way, the file was carved to see if any of the viewed images returned. If the suspected file is a database and contains the pictures as raw data, this process would have carved the pictures out, making the thumbnails available for further investigation. If the header of the file is a SQLite database, the file will be opened in a SQLite database viewer. A SQLite database viewer explores the fields and tables contained within the SQLite database file. Such a field within the database can contain binary data. This is then called a data "blob". If such a field contains the binary data of an image, the data can be copied out and written to a new file, which then can be opened like a normal image file. When the structure of the databases containing images, and the methods to extract the images, were already known, a method is determined on how to extract information from the databases containing images in an automated way. Our approach is to write a script, which was able to automate the extraction of found thumbnails. In other words, the script has to be able to find known databases containing images, and be able to extract said images. There are four phases in our automate process: (i) opening the forensic image; (ii) locating and extraction the databases; (iii) parsing SQLite databases and (iv) carving databases for images.

V. Experiments and Discussion

In our experiments, we tested our approach implemented as a Python script with seven popular image viewers listed as Irfanview, Faststone, Adobe Lightroom, Zoner photo studio, and Acdsee 19. We describe first of all how these image viewers store images and how we can retrieve images from them by using our automate approach mentioned above. We used the following datasets in our evaluation: (i) Images from www.pexels.com (open stock photo site, 4862 images, no EXIF); (ii) NEC animal toy dataset (5000 images from 60 toy animals, no EXIF); (iii) Describable Textures Dataset (5500 images of textures, no EXIF); (iv) ImageCLEF2011 collection (18000 images, EXIF, not embedded) [27]; (v) Imagenet Large Scale Visual Recognition Challenge 2015 test dataset (11142 images, no EXIF); (vi) California-ND Q0mex 2013 dataset (704 images, EXIF embedded) [28]; (v) Self-shot collection (392 images, EXIF embedded). We also used Encase to provide a baseline. During this testing, an estimation was made on how the entire process of extracting thumbnails from image databases could be automated within Encase.

A. Tools and storage methods

1) Irfanview:

The tool Irfanview is an image viewer, which at the first glance seemed to be an image viewer to only show single files. However, under the file option, a thumbnail option is visible. After selecting the thumbnail option, a folder tree of the machine becomes visible. When selecting a map under this thumbnail viewer, the tool will start to collect thumbnails of each picture available in the folder. During the scanning of the folder and the building of thumbnails, no files were written which could be related to Irfanview. It was possible the thumbnails were only loaded into memory. To validate this a folder with large amount of pictures was opened and the memory usage was monitored. The process "i_view32.exe" was using an increasing amount of memory as the thumbnails were created. A screenshot was made at the end of the scanning of the images and is shown in the image below (Figure 1).

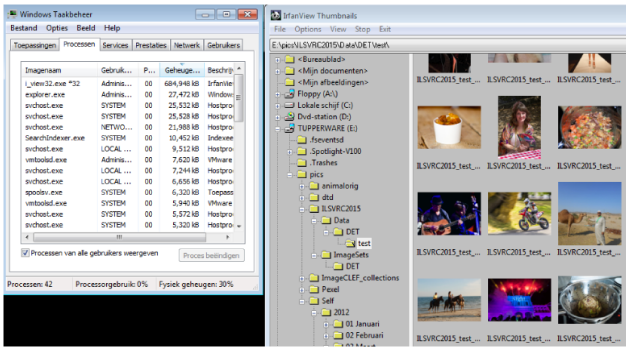

Fig. 1. Irfanview thumbnails

2) Faststone:

Faststone is a freeware image viewer tool. This tool also showed a browsable folder tree of the machine. The software was started and images were browsed. The file "FSViewer.db" located at the folder path *C:\Users\Administrator\AppData\Roaming\FastStone\FSIV* showed a high number of changes during the monitoring of file and folder changes. This file turned out to be a *TinyDB* database. This was done by looking at the data with the hex viewer "xxd". The thumbnails within were however not obfuscated and carving techniques were able to carve images out of the database. The tool scalpel was used for the carving process. The outcome of the scalpel tool is shown in the picture below (Figure 2).

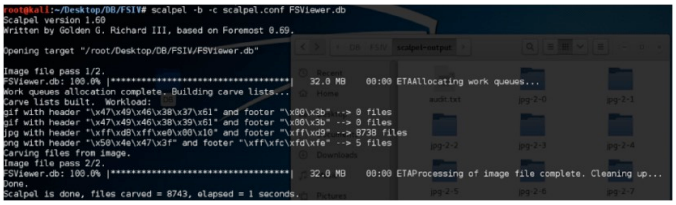

Fig. 2. Carving Fastone thumbnails

3) Adobe Lightroom:

Adobe Lightroom is a professional tool for managing and editing photos. It features many functions like adjusting colour, cutting photos and many more functions. To test this software a trial version of the software was downloaded and installed. The prepared images were browsed and the file changes were monitored. It became clear that the program created many files with the .db extension. These DB files turned out to be sqlite databases. Many of these database files seemed to log data about what has been done to images in the program. The database "root pixel.db" under the folder *"C:\Users\Administrator\Pictures\Lightroom\Lightroom CatalogPreviews.lrdata"* contained some thumbnails. The thumbnails were found under the Table "RootPixels" in the row "jpegData" of this SQlite database. The data of this field was not encoded and could be extracted to recreate a viewable picture, as shown below (Figure 3). Carving the data of the SQlite database with a standard tool did not yield any images.

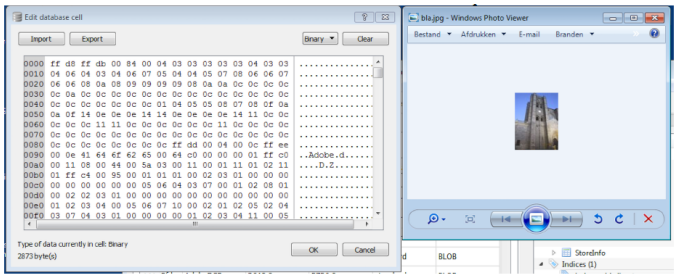

Fig. 3. Carving Adobe Lightroom thumbnails

4) Zoner photo studio:

Zoner photo studio is a tool for managing and editing photos. It features functions like adjusting colour, cropping and many more functions needed for a professional photographer. In this tool the folder structure was visible in a window on the left side. When browsing through this folder tree the images of the selected folder became visible on the screen. During the monitoring of folder and file changes one file stood out due to the large number of changes. The file that stood out was "data.zoner-index-cache" residing in the folder *"C:\Users\Administrator\AppData\Local\Zoner\ZPS 18\ZPS Cache.dat"*. This file was identified as an SQlite database and contained the metadata related to the files viewed within the photo studio. In this case metadata such as created date, filepath of the picture, author of the picture file and many more was saved within the database. The database did not contain the thumbnails for the viewed images. Thumbnails were saved as image files, although missing their extension in the filename, under the path *"C:\Users\Administrator\AppData\Local\Zoner\ZPS18\ZPSCache.dat\000\"*. Using Kali Linux, a preview of this folder has been made and is shown in the picture below (Figure 4).

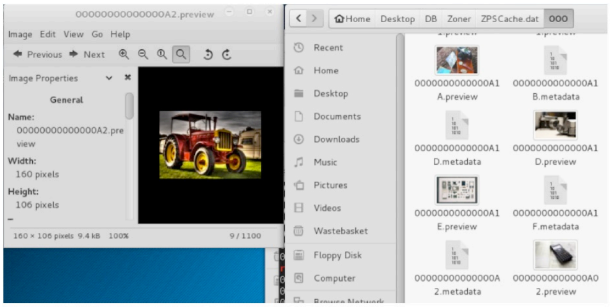

Fig. 4. Zoner thumbnails

5) ACDsee 19:

ACDsee is a tool for organising images. To test this tool a trial version was used. The tool notices the user of the existence of the database as soon an external media was browsed, due to the fact the tool asks the user if data of the external media should be stored in the database. After browsing the prepared images on the USBdrive, due to the logging of folder changes, it became clear that ACDsee stored its data under the path *"C:\Users\Administrator\AppData\Local\ACDSystems\Catalogs\190\Default\"*. Within this folder, tree files stood out due to the number of writes to the files, size and the filenames; Thumb1.fpt, Thumb2.ftp and Thumb3.ftp. These file where examined using kali Linux and the number of hits we obtained are 1406, 1342, 1269for Thumb1.ftp, Thumb2.ftp and Thumb3.ftp respectively. The Thumb.ftp files were also carved using the command "recoverjpeg –b 1 <filename>". The option –b1 stands for the blocksize of 1. This means the tool will ignore the standard block size of 512 and will recover images if they are written to a file continuously. In the image below the results of this process are shown (Figure 5). Upon visual inspection the recover files seemed to be the same except they were differing in size, with the Thumb1.ftp database containing the files with the largest size and best quality.

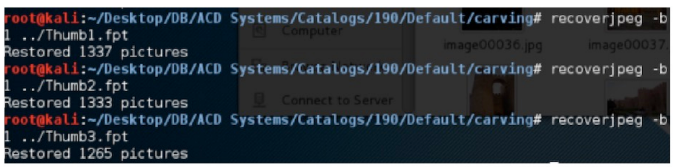

Fig. 5. ACDsee thumbnails

6) General discussion

Having investigated the way popular image viewers storing their data shows that different storage methods were used: (i) SQLite database; (ii) Custom database; (iii) Thumbnails as normal files in file system and (iv) in memory. The storage methods seem to differ so much from each other that one method of retrieving the information is not sufficient to retrieve the information of all the tools. For this reason multiple methods were used for retrieving the information from popular image viewers. In some cases the image viewers did not store any data about the viewed images on the hard disk. These tools stored the data in the memory. This leads to missing information when only the forensic image is investigated. Using live forensics, it may be possible to retrieve information from these tools that do not store information on the hard disk. Live forensics was however not in the scope of this paper. The reason is that live forensics cannot be conducted on a forensic image, but have to be conducted on the live running machine.

Only Irfanview stores data in memory. The database used by XnView was a SQLite database. The tables within the SQLite database, which likely contained the image data, seemed obfuscated. Another tool using a SQLite database for storing information about viewed images was Adobe Lightroom. The database was investigated with a SQLite database viewer. Such a viewer provides an easy way to examine the different fields stored within the database. Adobe Lightroom is a tool with many options and methods for viewing images. The tool Zoner Photo Studio also uses a SQLite database. This database stored the metadata of the viewed pictures. The thumbnails were stored as files within the file system. The way the thumbnails were stored by Zoner Photo Studio is a good example of a different way of storing them, in comparison to the other tools. The different method of storage also meant a different method of extracting the images should be used in comparison to when the images were stored within a database. Faststone and ACDsee 19 are tools that proved to have a custom database for storing the thumbnails of the viewed images. These databases did however not use compression or encryption to store the thumbnails. Not using encryption or compression means the data within is available in raw format, which means carving for files will retrieve the files contained in the database. With these tools, carving the databases for images retrieved the thumbnails of the images viewed within the virtual machine. It is possible more information was contained within these databases, and could have been extracted using tools for accessing exactly the used databases. These tools were however unavailable during the research, which is why only carving methods were used.

7) Discussion on the automate approach:

To find the databases the decision was made to search for the filenames of the databases within the file systems present in the forensic image. The downside of searching for filenames is when the creators of the image viewing tool change the name of the database, this has to be changed in the tool also. However this method saves time opposed to extracting all possible databases found on the image, and then having to determine which the right one is. Furthermore the searching within the file system is a less time consuming method then to carve the entire forensic image for database files. If a file matching the name of the database was found, it was extracted for further processing. To extract the images from the SQLite database, the exact table containing the images was extracted. By writing the data of these fields to a new jpg file, the images become available to the users of the script. The extracting of the fields containing the image data was done using the sqlite3 module within Python. There was also the idea to extract all SQLite databases available on the image, and then to parse each entry of all SQLite databases. This could however greatly increase the time the script needs to run as the data that needs to be processed increases. Also this would not only extract the images found in databases of image viewers, but also images which were found elsewhere. Such a module would be more suited for a script, which has as goal to extract as many images from a forensic image as possible. This may be a goal for a future research. For the carving of images from the databases containing the images in a raw format, an own function was created. The reason an own function was created for the carving is that a module that already did this with the options needed for the databases, was not found. The found modules within Python that did carving were focused on carving from a file system. To carve the files form the database file, the database file was first converted to hex values. Regular expressions were used to look for headers and footers of jpg images in the hex values of the database files. To keep this regular expression simple the choice was made to only extract jpg images. Making a regular expression search for many different headers and footers would also increase the time the script would need to carve trough the databases. Using many headers and footers may also return many false positive files.

B. Encase

As mentioned previously, we also did experiments with Encase to check if it is also able to extract the thumbnails from the databases. Using Encase, the databases containing images were investigated. All the actions in Encase used to retrieve the images were manual. Some steps may be automated using Enscripts. The parsing of SQLite databases needed to be done outside of Encase, as Encase cannot process these files alone. Encase sending the SQLite databases to an external tool makes it more complicated to automate the entire process of extracting images from the databases. The carving process in Encase seems to carve all the files within the image. This is a time consuming action to do because the tools has to scan for all files in allocated space of the hard drive investigated. The carving of the entire image will return many images available on the hard disk, images such as icons. Analyzing all images may cost a lot of time, compared to when only the viewed images from image viewers are extracted.

VI. Conclusion and Future Work

The time it takes to extract thumbnails from custom databases used by programs is an operational challenge faced by forensic investigators, and one of the problems stated in this paper. This could lead to the investigation of thumbnails being less common. An attempt was made to tackle this problem in this paper, by making a script which can extract the thumbnails in an automated way. First, the storage methods of the thumbnails were investigated. By using different tools, it

became apparent that the following storage methods were used: the images were stored in SQLite database, in other database in raw form, in raw form as one file per thumbnail, or only in the memory. The automated script dealt with this by looking for the thumbnails in the databases that were stored on the hard disk, not in the memory. During the investigation of the storage methods, the file names and locations of the database became apparent. This information was used to identify and extract the databases in the script. To extract the thumbnails from the found databases different techniques were used. With the goal of reducing the time needed to extract thumbnails from custom databases used by image viewing programs in mind, the script was designed to work as efficient as possible. The script took around one minute to complete on the forensic image used in this paper, returned thousands of thumbnails. There are many research can benefit from our approach such as the investigation of IMVU apps [29] or GIS forensic tools [30].

Locating and extracting the thumbnails was also evaluated using the widely used commercial forensic software, Encase. Findings suggested that not all steps could be automated. This was due to the fact a new case had to be made and the carving options to be set. The carving options were so that these could return the thumbnails. The downside however was that then all files were scanned, taking large amounts of time. Also, SQLite databases were not dealt with in the program itself. Automating the process using Encase may be done in a future work. In order to analyze a huge amount of thumbnails, we are also looking at data reducing techniques mentioned in [31][32]. In this case, we also need integrity rules [33] to assure the quality of representative data.

REFERENCES

[1] Aiken, M., Moran, M., and Berry, M. J. (2011) Child abuse material and the Internet: Cyberpsychology of online child related sex offending, Lyons, France: Meeting of the INTERPOL Specialist Group on Crimes against Children Lyons.

[2] http://www.xnview.com/en/xnview/. Accessed: February 2017

[3] http://www.acdsee.com/ . Accessed: February 2017

[4] Prat L., Baker C., Le-Khac NA. (2015) MapExif: An Image Scanning and Mapping Tool for Investigators, International Journal of Digital Crime and Forensics (IJDCF) Vol.7(2), pp.53-78

[5] http://stackoverflow.com/questions/3748/storing-images-in-db

[6] http://listoffreeware.com/list-of-best-free-image-viewer-software/

[7] Poisel R., Tjoa S., and Tavolato P. (2011). Advanced File Carving Approaches for Multimedia Files, Austria: St. Poelten University of Applied Sciences

[8] Quick D., Choo K-K R. (2014). Impacts of increasing volume of digital forensic data: A survey and future research challenges. Digital Investigation 11(4): 273-294

[9] Quick D., Choo K-K R. (2017). Big Forensic Data Management in Heterogeneous Distributed Systems: Quick Analysis of Multimedia Forensic Data. Software: Practice and Experience, In press

[10] Quick D., Choo K-K R. (2017). Digital Forensic Intelligence: Data Subsets and Open Source Intelligence (DFINT+OSINT): a Timely and Cohesive Mix. Future Generation Computer Systems, In press.

[11] Quick D., Choo K-K R. (2016). Big forensic data reduction: digital forensic images and electronic evidence. Cluster Computing 19(2): 723-740

[12] Swain M. J., Frankel C., Athitsos V. (1996). An Image Search Engine for the World Wide Web, Illinois: The University of Chicago

[13] http://www.sevenforums.com/music-pictures-video/244886-photo-thumbnails-explorerwont-update-after-photo-altered.html

[14] Curran K. (2009) TinEye Reverses the Search Process for Image Creators, Multimedia Information & Technology

[15] Quick D., Tassone C., and Choo K-K.R. (2014), Forensic Analysis of Windows Thumbcache files, SavannahUSA, Twentieth Americas Conference on Information Systems

[16] Pasonage H. (2012), Under My Thumbs:Revisiting Windows thumbnail databases and some new revelations about the forensic implications, Harry Personage

[17] Leom M.D., D'Orazio C.J., Deegan G., and Choo K.K.R. (2015), Forensic Collection and Analysis of Thumbnails in Android, IEEE Trustcom/BigDataSE/ISPA

[18] Pereira M.T. (2009), Forensic analysis of the Firefox 3 Internet history and recovery of deleted SQLite records, Fortaleza, Brazil: Digital Investigation.

[19] Sears R., Ingen C. V., and Gray J. (2007), To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem? California USA: University of Cailfornia

[20] Faheem M., Kechadi, M-T., Le-Khac N-A. (2015). The State of the Art Forensic Techniques in Mobile Cloud Environment: A Survey, Challenges and Current Trends, International Journal of Digital Crime and Forensics (IJDCF), Vol 7(2), pp.1-19

[21] Sgaras C., Kechadi M-T., Le-Khac N-A. (2015). Forensics Acquisition and Analysis of Instant Messaging and VoIP Applications. In: Garain U., Shafait F. (eds) Computational Forensics. Lecture Notes in Computer Science, Vol. 8915. Springer, Cham

[22] Newcomer S. and Martin L. (2014), Determining User Actions in OS X, based on Quicklook Thumbnail Cache Database Entries, Issues in Information Systems ,Volume 15

[23] Mohamad K.M., Patel A., and Deris M.M. (2011), Carving JPEG Images and Thumbnails Using Image Pattern Matching, IEEE Symposium on Computers & Informatics

[24] Wagner J., Rasin A. and Grier J. (2015), Database forensic analysis through internal structure carving, DFRWS USA: Elsevier.

[25] Brown Ralf D. (2014), Improved recovery and reconstruction of deflated files, Pittsburgh USA: Carnegie Mellon University Language Technologies Institute

[26] http://www.nirsoft.net/utils/folder_changes_view.html.

[27] Nowak s., Nagel K., Liebetrau J. (2011), The CLEF 2011 Photo Annotation and Conceptbased Retrieval Tasks, Amsterdam The Netherlands: CLEF 2011 working notes

[28] Jinda-Apiraksa A., Vonikakis V., Winkler S. (2013): California-ND: An annotated dataset fornear-duplicate detection in personal photo collections. In Proc. 5th International Workshop on Quality of Multimedia Experience, Klagenfurt, Austria: QoMEX

[29] Voorst RV., Kechadi M-T., Le-Khac N-A. (2015). Forensic Acquisition of IMVU: A Case Study, Journal of Digital Forensics, Security and Law, Vol. 10(4) pp.69-78

[30] Tillekens A., Le-Khac N-A., Pham Thi TT. (2016). A Bespoke Forensics GIS Tool. In Proceedings of 2016 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 15-17 December

[31] Le-Khac NA., Bue M., Whelan M., Kechadi MT. (2010) A Clustering-Based Data Reduction for Very Large Spatio-Temporal Datasets. In: Cao L., Zhong J., Feng Y. (eds) Advanced Data Mining and Applications. ADMA 2010. Lecture Notes in Computer Science, vol 6441. Springer, Berlin, Heidelberg.

[32] Aouad L., Le-Khac NA.,Kechadi MT. (2009) Grid-based approaches for distributed data mining applications, Journal of Algorithms & Computational Technology, Vol.3 (4) pp.517-534

[33] Pham-Thi TT., Helfert M., (2010) Discovering dynamic integrity rules with a rules-based tool for data quality analyzing, In the Proceedings of the 11th International Conference on Computer Systems and Technologies, June 17-18, Sofia, Bulgaria