Redacting Private and Sensitive Information in Born-Digital Collections

Kam Woods and Christopher A. Lee; University of North Carolina; Chapel Hill, NC

Abstract

Redaction of personal, private, and sensitive information from born-digital materials is increasingly important for repositories. Collection sizes are often too large to process without automation – the assistance of software designed specifically to identify and classify such information and present it in a format that facilitates redaction decisions. Distinguishing between items that may be redacted automatically and those that require manual intervention is similarly important.

This paper examines the identification, organization, and redaction of private and sensitive information identified within born-digital materials, particularly those contained on disk images extracted from fixed and removable media carriers. We identify specific items of interest in file systems and individual file formats that may be targets for redaction, and present two approaches to managing and providing access to redacted materials using open source tools developed for the BitCurator Access project (bca-webtools) along with supporting digital forensics software.

Motivation

Planning and implementing a redaction strategy for born-digital materials is an essential task for libraries, archives and museums (LAMs). Digital media acquisitions often contain data that may be classified as private, sensitive, or individually identifying, and the complexity and volume of information being collected demands automation to ensure that risks of inadvertent disclosure are minimized.

Personally identifying information (PII) is defined as any information that can be used to uniquely identify a specific individual. Common examples include full name, address, email address, date of birth, gender, age, financial account information, biometric data, and vehicle data. Sensitive information may be defined more broadly, and include privileged communications, classified data and/or communications, or information that may compromise legal, ethical, or contractual agreements.

The increasing complexity of mechanisms used to manipulate digital data – modern operating systems, file systems, and networked communications – mean that traditional (predominately manual) methods of redaction are in many cases no longer practical or effective.

These issues affect information produced in a wide range of domains. Publicly available government documents have long been sources of inadvertently leaked sensitive information. Unredacted information on legal proceedings, organizational procedures, and financial records are routinely described in popular media outlets. Research data sets (especially those from biomedical and social science studies) may include unredacted data that can be used to identify individuals. Finally, archives and manuscript materials are often collected from producers (e.g. individuals, organizations, families) who may have been unaware of the potential for extraction of sensitive content from their digital assets.

Storage devices containing bootable operating systems in particular present many challenges. Log files, web browser caches, system hibernation and recovery files can be significant sources of PII and sensitive data. Partially overwritten data and "deleted" files (those no longer visible in the mounted file system but still partially or fully recoverable) can be identified and extracted using specialized software.

In this paper, we examine applications of open source digital forensics and data recovery tools to digital materials acquired by LAMs. We do this specifically in the context of software designed to automate redaction, and address several specific issues: why redaction is needed; entities and events that can be identified in the data and associated metadata; types of redaction that can be performed on born-digital data with associated risks to data integrity; and software tools and workflows required for redaction.

Related Work

Information on current redaction practices for born-digital materials in LAMs is relatively sparse. A range of standards describe *types* of information to be redacted, but the actual practice of redacting this information is often bespoke – tuned to the collection, file types, and file systems being processed.

Privacy and confidentiality of information within collections of primary sources have been discussed thoughtfully within the archival literature [1][9], though discussions of technically implementing measures for born-digital materials have been quite limited. A guideline for archives planning to redact information from specific commonly used document formats can be found in [16]. More general guidance may be found in [4], [15] and [18], although the latter does not explicitly discuss electronic records or born-digital formats. The authors have previously published a general discussion of approaches to automating redaction of information from disk images [8], including potential applications of digital forensics tools [20].

In previous work [21], we developed a graphical interface and reporting mechanism that could be used by LAMs to build visualizations and text reports describing possible PII contained within disk images. This software depends on Simson Garfinkel's *bulk_extractor* to identify PII at byte offsets within a disk image, and uses the *identify_filenames.py* script distributed with that software to link those identified items, or features, to extant files. A set of Python scripts then transform this output into Excel and PDF documents that can be used to prioritize handling of individual files.

Approach

We focus on redaction of information from disk images extracted from digital media carriers, and on techniques that apply to a broad range of modern file systems and files understood by software libraries designed to identify them within a range of disk image formats. Our approach uses open source digital forensics tools and software developed for the BitCurator Access project to identify, report on, and redact PII and other sensitive data found in disk images.

In the following sections, we explain our criteria for selecting redaction candidates that may be automatically redacted and automatically verified, describe two approaches to redacting these items in raw and forensically packaged disk images, and demonstrate how these approaches are implemented in BitCurator

Access tools, including *bca-webtools*. We focus on the *process* of redaction, rather than the identification of individual items to be redacted, which vary considerably based on the tools being used and the criteria for redaction for specific collections.

Identifying Candidate Redaction Items

Knowing what to redact, selecting strategies to identify items that meet the relevant criteria, and executing the search for those items varies according to collection contents and institutional policy. For "data at rest" – in this case, data contained on storage devices transferred to a collecting institution – there are general redaction guidelines relevant to the needs of different types of institutions that wish to de-identify these media of PII and sensitive data [10]. Common redaction cases for LAMs may include:

- Names, addresses, phone numbers, Social Security numbers, PII associated with minors, birth dates, court records, and closed legal records
- Medical data that can be linked to individuals
- Corporate and personal financial information
- Data from research involving human subjects
- Classified information

Some of these categories are broad enough to require manual review, while others may be identified with greater consistency using automated tools. It is important to recognize that born-digital data can contain various patterns that, prima facie, do not appear to be sensitive, but can be sensitive when combined with other patterns (e.g. IP addresses that can be linked to specific individuals or behaviors) [11]. We focus on a subset of the scanners used by *bulk_extractor* to identify items likely to constitute a concern for LAMs: Social Security numbers, credit card numbers, birth dates, geolocation (GPS) metadata, exif metadata, email addresses, and email attachments.

In past work, we modified existing digital forensics tools and created new tools designed to build reports about many of these redaction candidates [21]. Here we describe a new tool chain to parse these lists and apply redaction actions in bulk. The following section discusses these approaches.

Redacting Items of Interest

Most file format-specific redaction tools manipulate file items directly using existing application programming interfaces (APIs) or knowledge of the file format structure to ensure the object remains format-compliant and will continue to render correctly using standard viewing software (e.g., Acrobat for PDF files).

Redaction of information from disk images introduces additional complexity. Redaction candidates can exist within specific files, in slack or unallocated space on the device, or in areas reserved for operation of the file system itself (for example, those reserved for file system metadata, boot operations, and swap space).

Commercial and open source forensic disk image formats are structured to discourage deliberate alteration. A forensic disk image, once created, is effectively read-only. It may be altered, but not without evidence of that alteration manifesting in mismatches in cyclic redundancy checks embedded in the file, or in MD5 and SHA checksum failures corresponding to the raw image contained within the forensic package. Redacting contents from a forensic image requires one of several alternate approaches: extracting individual file items from the image to be redacted and provided for access separate from the original file system; extracting the raw disk image from the forensic package and redacting it at the block level (either to be provided for access in raw form or repackaged in a new forensic container); and constructing a software library to blacklist contents of the original disk image at the directory, file, or block level.

In this paper we present implementations of the latter two approaches. Both of these approaches retain the structure and organization of the original file system and file-level metadata. Each has specific advantages depending on the access scenarios envisioned for the materials.

The first is the creation of a redacted "surrogate" disk image – a copy of the original disk image prepared for access by redaction at the block level. In the following section, we describe how to provide access using existing digital forensics software libraries to parse the structure of the file system in this surrogate and expose directories and files directly without mounting the file system. Use of existing digital forensics software libraries can prevent access to data in unallocated areas, as the access mechanism provides access only to those blocks associated with allocated files.

The second approach is the creation of an "access overlay" used to provide access only to specific directories and file items within the file system. This whitelist is constructed as an annotated Digital Forensics XML [5] document in which specific files and directories are effectively marked as "do not display." The second approach may be advantageous when institutional policy prohibits redaction of source materials, but there is some desire to provide access not only to files (which can always be copied out of a disk image), but also to the original, context-rich file system environment. In this case, the "access copy" is not a copy in the strict sense, but an access environment that prevents direct access to the original disk image, enabling the user to view the *structure* of the file system(s) while selectively masking out specific data and metadata.

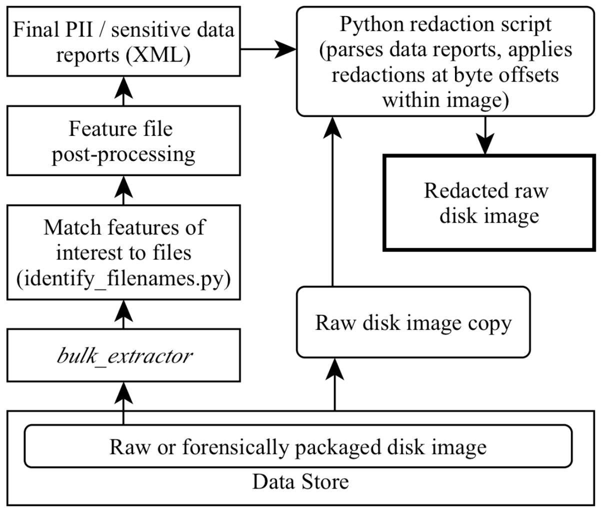
Creating Redacted Copies of Disk Images

Our first approach begins with the creation of a redacted disk image from an existing raw or forensically packaged disk image. A raw disk image may be redacted (albeit with potential loss of file system integrity) by writing changes back to the original file. A forensically packaged image (for example, one encoded using the Expert Witness format or Advanced Forensic Format) cannot be altered without compromising the validity of the checksums embedded within the file.

The original image may be captured raw from the source medium, or exported from a forensically packaged image – a facility provided by most commercial and open-source forensic format processing tools (including *libewf* and *AFFLIB*).

The first three steps of the process are automated by a master Python script, essentially identical to code developed for the BitCurator reporting tool [21]. The *bulk_extractor* tool is run with a user-selected set of scanners; the master script then runs *fiwalk* (distributed with The Sleuth Kit) and *identify_filenames.py* (distributed with *bulk_extractor*) to match items located by *bulk_extractor* to files in the file system. The output (tab-delimited line items in a text file) is then reprocessed into XML.

The redactions will be performed at the byte level within the raw disk image, so these intermediate reports are not strictly necessary. Their primary purpose is to provide the user a simple mechanism for editing the final list of items to be redacted (e.g., in Excel).



Final PII / sensitive data reports (XML)
Python redaction script (parses data reports, applies redactions at byte offsets within image)
Feature file post-processing
Redacted raw disk image
Match features of interest to files (identify_filenames.py)
Raw disk image copy
bulk_extractor
Raw or forensically packaged disk image
Data Store

Figure 1: Redacting byte sequences within a raw disk image.

The master script creates a raw copy of the disk image and executes a modified version of the *iredact.py* script (distributed with Simson Garfinkel's *DFXML* tools) to read each of these report files in turn and apply a user-specified redaction pattern – either a string (one or more ASCII characters), or pseudo-random characters at the relevant byte offsets. A final XML report is then produced to record each committed redaction alongside the original pattern and offset.

Use Cases and Concerns

This approach may be compelling to institutions that are preserving disk images, but do not – by design or mandate – wish to retain the source image as a preservation object (or wish to create access surrogates). Each step in the process described in the previous section produces a log of the analysis and any modifications in the redacted disk image. Retention of this log ensures there is a clear record of each alteration, and simplifies the process of identifying redaction actions should they cause damage to the file system or individual files.

There are several potential disadvantages to this approach. First, while tools such as *bulk_extractor* are capable of finding features within many common file formats (including those that have been compressed within certain container formats such as *zip* and *gzip*), many binary-encoded formats will elude such analysis without specialized plugins to process them (few of which currently exist for the open source tools discussed here).

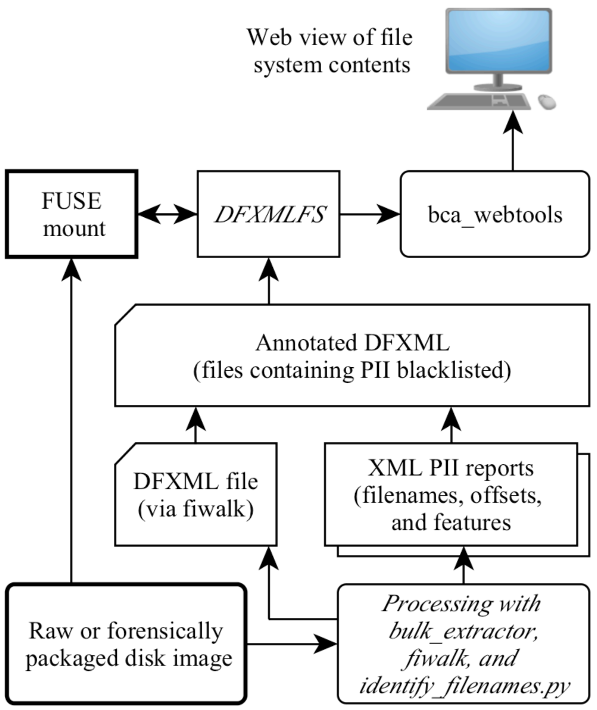
Second, edits to raw bytes within the disk image may cause files to fail to render properly using common tools, or the file system to fail to mount. As noted previously, most file format-specific redaction tools manipulate file items directly using existing APIs or knowledge of the file format structure to ensure the object will continue to render correctly.

Finally, the need to create a separate copy of the disk image for redaction may be undesirable when the intention is to retain the original as a preservation object.

Two additional concerns are the possibility of PII within the file system metadata, and the presence of private and sensitive material in unallocated areas in the disk image. Issues concerning the redaction of PII from file system metadata have been discussed elsewhere [3]. Altering these items may affect the ability of the relevant operating system to mount and navigate the disk image. The redaction process described in the previous section is capable of scrubbing select data from unallocated blocks, but does not currently include an option to "zero out" *all* unallocated areas in an existing (unmounted) disk image. There are dedicated utilities to perform this type of scrubbing for specific file systems [22].

Access Overlays for Disk Image Contents

The second approach leaves the unredacted disk image intact and relies on access controls to securely isolate the user from access to specific items (files or directories) within that disk image. In this implementation, access to the file system tree is provided via a web interface exposing only those items that have been marked as free of private and sensitive information. Access to the underlying disk image is moderated via a synthetic file system view using existing digital forensics software to extract files directly from the disk image. This approach provides redaction at the *directory* and *file* levels – file system objects are marked as "unavailable" if they contain PII or other sensitive data identified by the previously described digital forensics tools or the user.



Web view of file system contents
FUSE mount
DFXMLFS
bca_webtools
Annotated DFXML (files containing PII blacklisted)
DFXML file (via fiwalk)
XML PII reports (filenames, offsets, and features
Raw or forensically packaged disk image
Processing with bulk_extractor, fiwalk, and identify_filenames.py

Figure 2: Masking out files and directories containing sensitive information.

As in the previous approach, the disk image is processed using *bulk_extractor*, *fiwalk*, and *identify_filenames.py* to create an initial list of PII and sensitive items and link them to files within the file system. XML-formatted versions of these reports, along with the DFXML file, are used to produce an annotated DFXML file in which file items identified in the reports are marked as restricted.

The annotated DFXML file is passed to Alex Nelson's *DFXMLFS* [13], which presents the DFXML file as a mounted file system, rather than the originating disk image. The disk image (shown in Figure 2) is *only* accessed (via a file system in user space mount using *fuse-python*) when the contents of a file are requested by another application.

The *bca-webtools* web interface developed for the BitCurator Access project allows the user to navigate the view of the file system (or file systems, for images with multiple partitions) presented by *DFXMLFS*. This view is constructed using *only* the contents of the annotated DFXML file, effectively separating the user from restricted contents within the raw bitstream. This view may also be used to restrict access to file system items that are marked as unallocated (for example, those that have been deleted but are still identified as having data extant within the bitstream).

Use Cases and Concerns

The second approach could be appropriate for use cases in which unredacted disk images are retained as preservation objects, but access controls to specific contents of those images are desired. The disk image itself is isolated from the access interface (which sees only those file system items that are marked as unrestricted in the DFXML file), ensuring that both file items marked as restricted and all unallocated areas of the disk image remain protected.

In the implementation presented here, file items are marked simply as "available" or "restricted" depending on whether they contain specific features identified by a tool such as *bulk-_extractor*. No redactions are performed within the files themselves. While this ensures that no file or file system is altered in way that renders it inaccessible, it represents a reduced granularity of control with respect to the byte stream redaction method described previously.

Selective access to file-level items extracted from an unredacted disk image raises some additional security concerns. In our reference implementation, the *bca-webtools* application generates views into the file system solely based on information read from a DFXML report on that file system. Providing file download links necessitates (indirect, via *DFXMLFS*) server-side access to the unredacted disk image. Possible mitigations for the risk associated with this include a security audit of the code, or restriction of access to file items to a secured location (e.g., a monitored reading room).

Quantifying Successful Redaction

Identifying a completed redaction event is possible when the baseline set of items to be redacted in an object is known in advance; the modified bitstream can be tested directly to determine if the redaction pattern has, in fact, been applied. However, this basic test may not be the sole acceptable criterion for success. A file redacted at the block level may be damaged and unrenderable using standard tools. Likewise, a file system redacted at the block level may trigger errors in file system integrity checks imposed by the operating system.

In the block-level redaction approaches described here, we do not consider whether or not a redaction action affects the ability of a file to be rendered or a file system to be mounted without error. The tools produce simple logs to verify the raw number of PII and other sensitive items automatically detected, and the subsequent count of actions performed to redact from the raw disk image or annotate the DFXML structure representing the file system.

The criteria for success may also include whether or not the redaction procedure can be easily reversed. Yet even this simple case may pose problems. Even if a unique-length string is replaced with default-length substitute text, it may still be possible to infer – from sentence structure or document layout – pertinent information about the redacted contents from the original document.

For large acquisitions, manually verifying redaction performance may be impractical. This is problematic when different PII and sensitive data identification tools perform at different levels of accuracy and precision, although some effort has been made in the past to quantify the performance of open source tools against existing commercial solutions [6].

Comparing Digital Objects to Identify Redactions and Discrepancies

LAMs working with born-digital materials are tasked with maintaining records of provenance and ensuring a clear chain of custody for acquired materials. Yet there are few mechanisms to ensure that any given copy of a digital object (whether it is acquired on digital media or transferred over the network) is both authentic *and* unredacted. There is ongoing work in the digital forensics field to develop tools that perform statistical analysis of block-level data contained within disk images to try and distinguish between compressed, encrypted, and randomized data within the context of the file system [7]. The techniques we have described here depend on recordkeeping to ensure a clear history of any redaction actions undertaken.

When working with "data at rest" – here, data that is acquired from fixed or removable digital media and stored within a repository) – the simplest way to ensure a clear record of any alterations or redactions performed is to retain a copy of the forensically acquired (using a hardware write-blocker) bitstream from a given device. In cases when this is impossible or impractical – due to device size, large quantities of unneeded or unwanted data, or other considerations – retention of the file system metadata alone can still be useful, as it provides a record of original file sizes on disk, file names, and other metadata – including cryptographic checksums for individual files that will change following a redaction.

Future Directions

Creation of *redacted* disk images is not currently a common practice in collecting institutions, and there are a number of limitations associated with the approaches described here. In this section, we discuss possibilities for future development of redaction tools in the BitCurator Access project as well as implications for professional practice.

Redaction of a raw disk image has the potential to leave the file system – or file items contained within the file system – in an inconsistent state. Currently, the tools described here do not perform any validation on the structure of the disk image after redacting the raw bitstream. The addition of a file system consistency check tool such as *fsck* (both prior to and post-redaction) is planned for future versions.

Similarly, the tool chain does not currently attempt to verify the consistency of individual files (i.e., whether the format structure remains viable post-redaction). Tools developed for digital preservation, such as *FIDO* and *JHOVE2*, provide this functionality for certain formats, at the expense of considerable additional time overhead.

With respect to identification of PII and sensitive data, the tools described here rely heavily on pattern matching techniques, but they do not perform any linguistic or semantic analysis of the content. This may be a concern as named entities – people, places, and things – are likely to be common redaction candidates for preservation institutions. The additional of a natural language processing model as a purpose-built plugin for *bulk_extractor* or as a standalone processing stage could assist in identifying these items.

The approaches described here also could be enhanced to address more complex needs, including redaction of file system metadata (such as timestamps and filenames) while still retaining the ability to browse the file system directly, and encrypting – rather than scrubbing – arbitrary byte sequences within the disk image (similar functionality is proposed but not currently implemented within the *iredact.py* script).

Finally, it is important to recognize that the management of private and sensitive information within collections is a set of processes within an ever-changing landscape. Given the ability to infer data values through comparisons across data sources, privacy protection cannot be fully reduced to the redaction of fixed, discrete sets of data elements [12]. Protection of the interests of relevant stakeholders is a long-standing responsibility of LAM professionals, which has always involved professional judgement and response to unexpected contingencies. Responsible curation of born-digital data must be attentive not only to specific patterns in the data, but also to the role of those patterns in the materials' context of creation and use [2] and the potential "impact level" (risks from disclosure) of those patterns [10]. We are designing tools and guidance to automate as many of the discrete tasks as possible, so LAM professionals can focus their attention on higher-level issues related to preservation, description and provision of access to born-digital materials.

Conclusion

We have presented tools and techniques that support two core uses cases concerning redaction of and access to disk images. The first allows PII and sensitive data to be redacted from a raw copy of an existing disk image using modifications of existing open source digital forensics tools. The second provides select access to file and directory items within a disk image using a synthetic file system view in a web interface, masking out items containing PII and sensitive data at the file level but leaving the original disk image untouched.

These approaches address two ongoing needs in libraries, archives, and museums working with born-digital media. The ability to permanently redact data from complex digital objects such as disk images while minimizing impact to the structure and organization of the file system; and the ability to semi-automatically create online access environments that allow users to browse the natural structure of a disk image while securely preventing access to file-level items containing sensitive and restricted materials.

Acknowledgements

We would like to thank the other members of the BitCurator Access team (Alex Chassanoff and Sunitha Misra) and others who have provided input, including those serving on the BitCurator Access Advisory Board. BitCurator Access is funded by a grant from the Andrew W. Mellon Foundation.

References

[1] Behrnd-Klodt, Menzi L., and Peter J. Wosh, eds. 2005. Privacy & confidentiality perspectives: archivists & archival records. Chicago, IL: Society of American Archivists.

[2] Bingo, Steven. 2011. "Of Provenance and Privacy: Using Contextual Integrity to Define Third-Party Privacy." American Archivist 74:506-521.

[3] Emerson, Casey. *Automating Disk Image Redaction.* UNC SILS Masters Paper. Last accessed December 7, 2014 https://cdr.lib.unc.edu/record/uuid:5108ce54-27f4-42a0-b678-c3b89f69cd69

[4] *FBI redaction manual.* RT.com. Last accessed December 7, 2014 http://rt.com/usa/fbi-secret-manual-library-congress-580/

[5] Garfinkel, S. *Digital Forensics XML and the DFXML toolset, Digital Investigation*, 8 (2012), 161-174.

[6] Garfinkel, Simson, *Digital media triage with bulk data analysis and bulk_extractor*. Computers and Security 32: 56-72 (2013)

[7] Garfinkel S, Nelson A, White D, Roussev V. *Using purpose-built functions and block hashes to enable small block and sub-file forensics*. In: Proceedings of the tenth annual DFRWS conference. Portland, OR: Elsevier; 2010.

[8] Lee, Christopher A. and Kam Woods. 2012. *Automated Redaction of Private and Personal Data in Collections: Toward Responsible Stewardship of Digital Heritage*. The Memory of the World in the Digital age: Digitization and Preservation, 2012. Vancouver, BC.

[9] MacNeil, Heather. 1992. *Without consent: the ethics of disclosing personal information in public archives.* Chicago, IL: Society of American Archivists.

[10] McCallister, Erika, Tim Grance and Karen Scarfone. 2010. *Guide to Protecting the Confidentiality of Personally Identifiable Information (PII)*. National Institute of Standards and Technology.

[11] McIntyre, Joshua J. *Balancing Expectations of Online Privacy: Why Internet Protocol (IP) Addresses Should be Protected as Personally Identifiable Information*. DePaul Law Review 60 (2011): 895-936.

[12] Narayanan, Arvind, and Vitaly Shmatikov. 2010. "Privacy and Security: Myths and Fallacies of 'Personally Identifiable Information'." Communications of the ACM 53 (6):24-26.

[13] Nelson, Alex. 2015. Source code retrieved from GitHub: https://www.github.com/ajnelson/dfxmlfs

[14] *Personally identifiable information.* Wikipedia. Last accessed December 7, 2014 http://en.wikipedia.org/wiki/Personally_identifiable_information

[15] Ranker, Ben. 2009. *Redacting Software Recommendation for MARBL Digital Archiving*. Emory Libraries Tech Know-how. Last accessed March 16, 2015. https://techknowhow.library.emory.edu/blogs/branker/2009/06/03/redaction-software-recommendation-marbl-digital-archiving

[16] Redwine, Gabriella et al. 2013. *Born-Digital: Guidance for Donors, Dealers, and Archival Repositories*. Council on Library and Information Resources. Last accessed December 7, 2014. http://www.clir.org/pubs/reports/pub159/pub159.pdf

[17] Roussev, Vassil. *An Evaluation of Forensic Similarity Hashes.* Digital Investigation 8 (2011): S34 - S41.

[18] *UK National Archives Redaction Toolkit.* UK National Archives. Last accessed December 7, 2014 http://www.nationalarchives.gov.uk/documents/information-management/redaction_toolkit.pdf

[19] *When to redact AIPs vs DIPs in digital archives workflows?* Quanda forum. Last accessed December 7, 2014 http://qanda.digipres.org/121/when-to-redact-aips-vs-dips-in-digital-archives-workflows

[20] Woods, Kam, Christopher A. Lee and Simson Garfinkel *Extending digital repository architectures to support disk image preservation and access*. Proceedings of the 11th Annual International ACM/IEEE Joint Conference on Digital libraries, 2011.

[21] Woods, Kam, Christopher Lee, and Sunitha Misra. *Automated Analysis and Visualization of Disk Images and File Systems for Preservation*. In Proceedings of Archiving 2013 (Springfield, VA: Society for Imaging Science and Technology, 2013), 239-244.

[22] *Zerofree: Keeping File System Images Sparse* http://intgat.tigress.co.uk/rmy/uml/index.html Last accessed April 2, 2015

Author Biographies

Kam Woods is a Research Scientist at the School of Information and Library Science at the University of North Carolina at Chapel Hill. His research interests include long-term digital preservation, digital archiving, and the application of digital forensics tools and techniques to archival and preservation data analysis and management.

Christopher (Cal) Lee is an Associate Professor at the School of Information and Library Science at the University of North Carolina at Chapel Hill. His primary area of research is the long-term curation of digital collections. He is Principal Investigator for the BitCurator Access project and editor of I, Digital: Personal Collections in the Digital Era published by the Society of American Archivists.