

A Framework for Extended Acquisition and Uniform Representation of Forensic
Email Evidence

by

Justin W Paglierani

A Thesis Presented in Partial Fulfillment
of the Requirement for the Degree
Master of Science

Approved November 2013 by the
Graduate Supervisory Committee:

Gail-Joon Ahn, Chair
Stephen S. Yau
Raghu T. Santanam

ARIZONA STATE UNIVERSITY

December 2013

ABSTRACT

The digital forensics community has neglected email forensics as a process, despite the fact that email remains an important tool in the commission of crime. Current forensic practices focus mostly on that of disk forensics, while email forensics is left as an analysis task stemming from that practice. As there is no well-defined process to be used for email forensics the comprehensiveness, extensibility of tools, uniformity of evidence, usefulness in collaborative/distributed environments, and consistency of investigations are hindered. At present, there exists little support for discovering, acquiring, and representing web-based email, despite its widespread use. To remedy this, a systematic process which includes discovering, acquiring, and representing web-based email for email forensics which is integrated into the normal forensic analysis workflow, and which accommodates the distinct characteristics of email evidence will be presented. This process focuses on detecting the presence of non-obvious artifacts related to email accounts, retrieving the data from the service provider, and representing email in a well-structured format based on existing standards. As a result, developers and organizations can collaboratively create and use analysis tools that can analyze email evidence from any source in the same fashion and the examiner can access additional data relevant to their forensic cases. Following, an extensible framework implementing this novel process-driven approach has been implemented in an attempt to address the problems of comprehensiveness, extensibility, uniformity, collaboration/distribution, and consistency within forensic investigations involving email evidence.

To all that have pushed me to be better than I was the day before.

ACKNOWLEDGMENTS

First, I'd like to thank my advisor Dr. Gail-Joon Ahn for all of his help and support. Without you none of this would be possible — you've opened innumerable doors for me and pushed me to achieve things I could not have imagined I was capable of just three years ago.

Second, to my family who have loved and supported me through all of my trials and tribulations. I love you all dearly, hope I have made you proud, and will always try to live up to your expectations.

My friends and labmates — you have all acted as siblings, mentors, stressors, cheerleaders, and relaxants. I owe each and every one of you something and sincerely look forward to seeing what you achieve.

Finally, to my committee and the other faculty members who have taken an interest in me and my education. Your involvement has helped shape me as a student and a professional and is greatly appreciated.

All of your support has meant the world to me.

This work was completed under the generous support of Arizona State University's Scholarship for Service program, part of the National Science Foundation's CyberCorps initiative.

TABLE OF CONTENTS

	Page
LIST OF TABLES	vii
LIST OF FIGURES	viii
CHAPTER	
1 INTRODUCTION	1
1.1 Traditional Forensics	1
1.1.1 Acquisition	2
1.1.2 Authentication	4
1.1.3 Analysis	4
1.1.4 Presentation	5
1.2 Problems in Email Forensics	5
1.2.1 Comprehensiveness	6
1.2.2 Extensibility	6
1.2.3 Uniformity	6
1.2.4 Collaboration/Distribution	7
1.2.5 Consistency	7
2 RELATED WORK	9
2.1 Comprehensiveness	9
2.2 Extensibility	10
2.3 Uniformity	11
2.4 Collaboration/Distribution	11
2.5 Consistency	12
3 METHODOLOGY/PROCESS	13
3.1 Initial Acquisition	13
3.2 Credential Discovery	14

CHAPTER	Page
3.3	Evidence Mapping 15
3.4	Supplemental Acquisition 16
3.4.1	Acquisition Definitions 17
3.5	Evidence Processing and Authentication..... 18
3.6	Analysis 19
3.6.1	Reduced Effort of Analysis 20
3.6.2	Analysis Techniques..... 20
4	IMPLEMENTATION DETAILS 23
4.1	The PlugsE Framework 23
4.2	Initial Acquisition 26
4.3	Credential Discovery 27
4.4	Evidence Mapping 29
4.5	Supplemental Acquisition 31
4.6	Evidence Processing and Authentication..... 34
4.6.1	Evidence Representation 34
4.6.2	Evidence Processing..... 38
4.7	Analysis 40
5	EVALUATION 41
5.1	Comprehensiveness 41
5.2	Extensibility 42
5.3	Uniformity 44
5.4	Collaboration/Distribution 45
5.5	Consistency 48
5.6	Performance..... 49

CHAPTER	Page
5.7 Summary of Results	52
6 DISCUSSION	53
6.1 A Note on Legality	53
6.2 Rules of Evidence	53
6.3 Collaboration.....	55
6.4 General Use of the Approach	56
6.5 Limitations of the Approach	56
7 CONCLUSION	58
7.1 Summary	58
7.2 Contributions	58
7.2.1 Process-Driven Approach	59
7.2.2 Evidence Representation	60
7.2.3 Realization of Approach	60
7.3 Future Work	61
REFERENCES	63

LIST OF TABLES

Table	Page
2.1 Predicted daily email traffic in billions from 2012-2016 as published by the Radicati Group	10
5.1 Set of checksums after emails have been deleted and/or received	48

LIST OF FIGURES

Figure	Page
1.1 The possible complexity of evidence apparent within a system	2
1.2 The complexity of evidence directly apparent within traditional forensics	3
3.1 The traditional forensic workflow combined with the email approach . . .	14
3.2 The Possible Complexity of Information Flows within Email	21
4.1 The PlugsE framework	24
4.2 The Class Structure of the PlugsE Backbone	24
4.3 Sequence Diagram Showing the Information Flow Through the PlugsE Framework	25
4.4 A High-Level Representation of Henson’s Structure	26
4.5 PlugsE’s Command Line Interface Output	26
4.6 An Abbreviated Sample of Henson Output	29
4.7 All the cookies created by logging in to Gmail	31
4.8 An Abbreviated Sample of Crumbler Output	32
4.9 An Abbreviated Sample of Spatula’s Intermediate Representation	32
4.10 An Abbreviated Sample of Spatula’s Output	33
4.11 A Sample of a Thunderbird Mail Summary File	36
4.12 An Abbreviated Sample of the EFXML Format	36
4.13 An Abbreviated Sample of the EFRDF Format	37
5.1 The coverage of email evidence provided by traditional forensics	42
5.2 The coverage of email evidence provided by PlugsE’s comprehensive process	43
5.3 An Entry in a Manifest File	44
5.4 Gmail’s Presentation of Email — Internal Representation is Unknown .	44
5.5 Internal Mbox Representation of Email Used by Thunderbird	45

Figure	Page
5.6 uniform Representation from Either Data Source	46
5.7 An Overview of the Cuff Architecture [1]	47
5.8 Time to discover known cookies in Chrome cookie database	49
5.9 Conversion times of single mbox files in the Enron data set by number of messages	50
5.10 Average Size of mbox and EFXML Representations Relative to PST in the Enron Data (Normalized)	51

Chapter 1

INTRODUCTION

The recent investigation of a senior U.S. intelligence official reaffirmed the importance of email forensics [2]. The investigation relied on the simple inspection of the drafts folder of a shared email account, but if the suspects had taken the time to make their correspondence more clandestine, a more sophisticated approach would have been necessary to discover relevant email evidence. Current methodologies do not address the possible intricacies introduced when an investigation centers around the analysis of various email sources as shown in Fig. 1.1 and, furthermore, simple inspection does not aid an examiner in detecting the presence of email which is not locally stored [3, pg. 471] [4, ch. 11].

Consider a scenario in which a suspected computer criminal has communicated with many parties about the nature and means of their actions using various communication methods, including locally stored emails and webmail accounts. When an examiner seizes a suspect's hard drive, only the locally stored or cached email would be directly available and the webmail accounts would remain undiscovered without substantial manual effort, as seen in Fig. 1.2. These missing portions of data could lead to an incomplete investigation report with respect to the suspected act. Even if evidence resides locally in diverse formats, it is likely that an examiner would need separate tools and methods to analyze each of them.

1.1 Traditional Forensics

In order to fully understand the problems in the current forensic process which will be addressed, one must possess some background in the traditional forensic process.

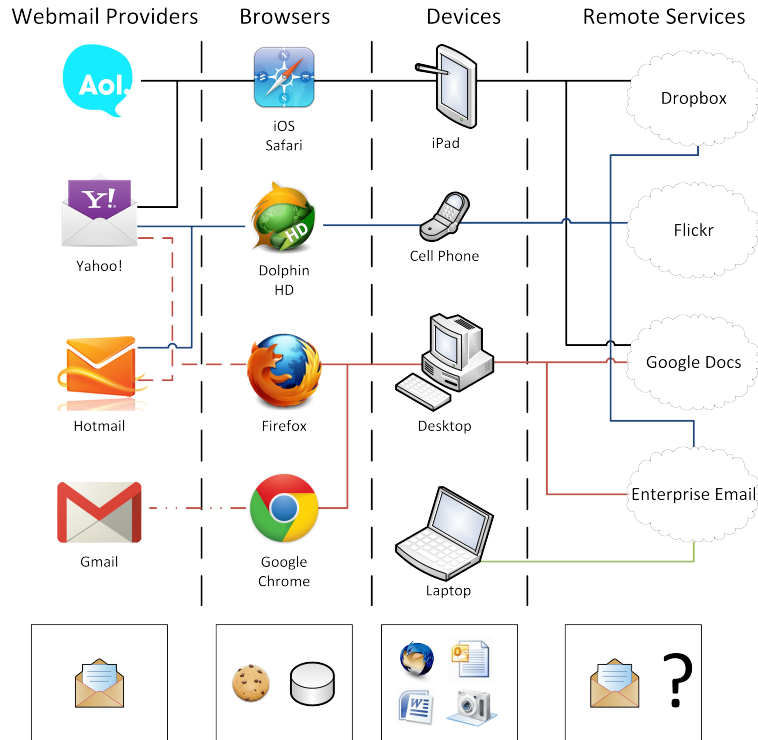


Figure 1.1: The possible complexity of evidence apparent within a system

In particular, current forensic processes typically focus on disk forensics and email forensics are left to be handled during the analysis phase of the investigation. Without treating email forensics as a process-driven task, the success of the investigation may hinge upon the examiner’s technical skill and intuition with regards to the specific evidence. In this section, a brief overview of the traditional forensic process will be given beginning with acquisition and continuing through authentication, analysis, and presentation [5, ch. 1]. While more complex process models have been emerged over time [6, ch. 6, sec. 1], the basic tenets of the process remain the same.

1.1.1 Acquisition

The goal of the acquisition phase is to obtain the relevant sources of information which may be relevant to building a case. During the acquisition phase, the system which contains evidence is first seized by the authorized parties, which may be law

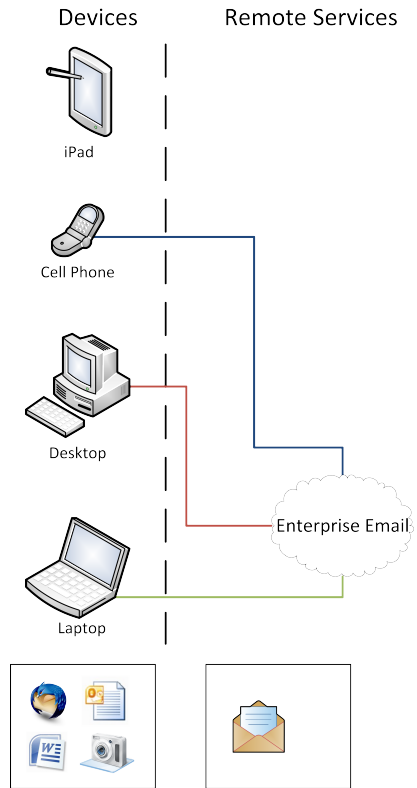


Figure 1.2: The complexity of evidence directly apparent within traditional forensics enforcement officials, technical staff, or a number other representative of the stakeholder which will be conducting the investigation. Next, the data it stores is copied in one of two fashions: live (or dynamic) acquisition, which is taken from a system as it runs and changes (often including highly volatile forms of storage such as RAM [5, pg. 5]), and static acquisition, which is taken from a system that is no longer changing (often it has been powered off). This phase is one of the most important in the forensic process, as it is the one which is most likely to introduce changes to the system which may misrepresent the evidence required, invalidating it in a court of law [7, pp. 135]. Such a copy is often referred to as a “forensic copy” [3, pp. 52] and is the representation of the evidence most often worked with in an attempt to minimize the amount of dangerous interactions with the original source of the data. Outside of the copies of

email evidence residing within the filesystem, there is currently no best practice with regards to acquiring email evidence.

1.1.2 Authentication

The authentication phase is concerned with the validation and verification of the forensic copy acquired. In the simplest sense, this means that the forensic copy is checked against the original and verified to have undergone no changes [5, pp. 13], however as the computing ecosystem evolves this task becomes nontrivial and some argue that is no longer the case [6, pp. 19], especially with regard to live acquisitions. This phase is most easily carried out using using cryptographic hash functions, such as MD5 or the SHA family, which are often referred to as a “digital fingerprint” [4, pp. 306] to explain their use as unique identifiers of pieces of data.

1.1.3 Analysis

The analysis phase is the one which most people envision when hearing the phrase “digital forensics”. It is within this phase that an examiner attempts to discover information about the suspect’s actions in order to recreate a time line of events which either incriminates or exonerates the suspect. This phase is highly intuitive and draws greatly upon the examiner’s technical knowledge as well as their past experiences; because of this it is sometimes referred to as ”the art of forensics” [7, pp. 143]. Examples of tasks which may be carried out during analysis include examining well-known file formats, looking in oft-ignored locations within storage media to discover intentionally hidden or obfuscated evidence, and recovering deleted files which the suspect may have tried to dispose of in an attempt to cover their tracks. While this phase is clearly the one which yields the most important results, it is incomplete if all of the possible evidence has been not acquired for its use.

1.1.4 Presentation

The final phase of traditional forensics is the presentation phase, in which a complete narrative of the sequence of events is documented with accordance to any laws or regulations in place to protect the accused [5, pp. 19]. This final representation of the case must include all relevant documents which may be used by a disciplinary committee or a lawyer in a court of law.

1.2 Problems in Email Forensics

From this discussion, it becomes clear that there is a need to provide a systematic and forensically sound methodology for discovering, extracting, and representing email evidence from media where it may not be immediately available. Using a process-driven approach [8], an extensible framework for conducting comprehensive and consistent acquisitions of email and representing them uniformly for the purpose of forensics can be created. This will aid examiners in carrying out forensics more efficiently while providing uniform results and more effectively through creating an environment that facilitates collaboration. This framework will utilize pluggable modules so that it may be provided as a service to multiple examiners without requiring them to alter their forensic environment or tools. Also, multiple examiners and tools have provided their own regulatory report and proprietary data formats, forming a critical barrier to the sharing of work and results between organizations. To mitigate this problem, the EFXML and EFRDF evidence container formats [8] will be utilized to create a system which represents email evidence from varying data sources uniformly in a structured and interoperable evidence container for collaborative email forensics.

1.2.1 Comprehensiveness

Since the goal of forensic investigations is to create a full narrative of events, as much evidence as possible should be collected through any process used during its duration. As current methodologies leave discovering and acquiring email evidence outside of those formats directly available in disk evidence to the examiner, a useful process-driven approach should provide comprehensive evidence acquisition so that as much email evidence as possible is detected through its use.

1.2.2 Extensibility

As the software ecosystem surrounding email has been changing rapidly since the advent of the internet, any tool which handles only a specific data source is likely to become outdated somewhat quickly. As such, a framework following a process-driven approach should exist and be extensible so that, as best practices emerge, evidence can be acquired from new services without large changes to an examiners forensic environment.

1.2.3 Uniformity

Once the various evidence sources have been approached by a process-driven framework to acquire data, there is a need to represent it uniformly. As differing email sources will represent email evidence differently both internally (e.g. at the service provider's data center) and to the user, the problem of treating email as a cohesive body of evidence rather than multiple disjoint parts arises. A uniform evidence container allows for varying email sources to be shown in the same manner so the examiner can focus on the task at hand (examining the characteristic email messages) rather than on the specifics of each provider's representation.

1.2.4 Collaboration/Distribution

Current forensic practices do not facilitate collaborative or distributed approaches to forensics. Often, investigations may be hindered by barriers such as the physical location of resources or conflicting policies of cooperating organizations. New approaches to forensics such as those presented in [1] attempt to address these issues generally, but processes must be put in place to apply them specifically to acquiring and representing forensic email evidence.

1.2.5 Consistency

As current practices in email forensics weigh heavily upon the technical skills and past experiences of the examiner, a consistent approach to email forensics is difficult to guarantee. These efforts can be further hindered during investigations focusing upon active systems, which may change state (e.g. emails are received, deleted, or both) in between separate acquisition events; such changes may bring into question whether the examiner initially missed or ignored evidence rather than the state of the system being changed. A well-defined methodology is needed to guarantee consistent results of acquisitions taking place at different times or by different examiners and, in the situation where differences are inevitable, their causes and results should be well-known and clearly stated.

This thesis is organized as follows. In Chapter 1 an overview of the current state of email forensics along with its problems will be discussed and in Chapter 2 a survey of related work will be shown. Chapter 3 will present a process-driven methodology for email forensics including its steps and their benefits. Chapter 4 will describe an accompanying proof-of-concept implementation to carry out this process-driven approach against a Gmail account. Following, Chapters 5 and 6 will show evaluations

in relation to the problems presented in Chapter 1 and discussion of the approach and implementation with respect to the approach's legality, bearing on the rules of evidence, general usage, and limitations. This thesis concludes in Chapter 7 with a summary of its contents, contributions, and future work.

Chapter 2

RELATED WORK

While this work represents a new contribution to the field of email forensics, there has been significant research into related fields. In this chapter a list of related works used as support for this work will be presented with regards to comprehensiveness, extensibility, uniformity, collaboration/distribution, and consistency as discussed in Chapter 1.

2.1 Comprehensiveness

As more interactions become digitized, ranging from communications to finances, a number of forensic hurdles present themselves [9]. Service-oriented computing presents an interesting challenge, as examiners no longer see unified bodies of evidence aggregated within traditional forensic mediums. Significant research has gone into approaching specific services [10, 11], but little work has gone into establishing a best practice approach to such evidence starting with the initial acquisition of disk-based evidence¹. Current tools only support email evidence which is directly evident in a disk image or web-based accounts to which the credentials are currently possessed [12]. A full solution should include a means to discover credentials, map them to a given service, and acquire the evidence from said service. In case there was ever any doubt as to how important email is to private and corporate communication, the Executive Summary [13] of the Radicati Group’s report titled “Email Market, 2012-2016” states that the total number of worldwide emails sent each day in 2012 was about 144.8 billion,

¹“Disk-based evidence” is meant to include all forms of digital evidence that have more traditionally been part of an investigation, not just hard drives.

Table 2.1: Predicted daily email traffic in billions from 2012-2016 as published by the Radicati Group

Year	2012	2013	2014	2015	2016
Total worldwide emails/day (B)	144.8	154.6	165.8	178.3	192.2
% Change	—	7%	7%	8%	8%

with steady growth predicted for years to come as shown in Table 2.1. Furthermore, the report states that “the installed base of Corporate Webmail Clients is expected to grow from 629 million in 2012 to over 1 billion by year-end 2016.” Clearly webmail is a significant communication medium, however it is not addressed by traditional forensics in a comprehensive manner.

Improvements in the forensic analysis of email have largely followed that of big data — recent contributions to the field include statistical and machine learning techniques used to facilitate stylometric analyses, author attribution, and more into a cohesive analysis technique [14, 15, 16]. While these methods improve the analytic process of email forensics, there still lacks a holistic approach which provides comprehensive results regardless of source format and the semantic details of the evidence.

2.2 Extensibility

As previously mentioned in Chapter 2.1, there is significant research focus on forensics related to specific services, but little in the way of a generic approach. Further, current email forensics tools only handle a handful of specific email file formats and service providers [12, 17, 18, 19] and offer no room for extensibility. With the ecosystem of web service providers and email clients rapidly growing, over time these tools may

begin to miss significant portions of relevant evidence, making extensibility of a solution critical to the comprehensiveness of the evidence it discovers [20].

2.3 Uniformity

Best practices have emerged in the forensic representation of evidence. In particular, XML has become known as a medium for the creation of well-structured forms of evidence representation [21]. A well-known example of this is the Digital Forensics XML (DFXML) format, used for representing disk media as a combination of disk partitions, file systems, and file metadata in XML [22]. These formats facilitate the storage, authentication and analysis of evidence in various ways, but do not meet the needs of email forensics. Similar to disk forensics, email also contains indexable metadata, in the form of headers, which can be useful to direct the focus of an analysis. From this metadata alone, an examiner can detect communication flows and evidence tampering, among other things. Just as filesystem metadata has been shown to be useful in forensic investigations [23], the email headers are a valuable source of information in a forensic investigation involving email [24]. These indexes of email metadata are currently used in many email clients, though their representation is not standardized and some are neither human readable nor easily utilized by developers [25].

2.4 Collaboration/Distribution

As cybercrime grows, digital "...forensics is increasingly a team effort..." and requires collaborative tools [20]. Also, since many recent developments (e.g. high disk capacity) in computing requires complex analysis to perform forensics on evidence, distributed approaches have been proposed as a means to address performance issues faced by examiners during investigations [26, 27]. A proposed cloud-based, collabora-

tive forensic framework known as CUFF addresses these issues, but its application to specific use cases is an ongoing effort [1].

2.5 Consistency

For forensic investigations to be successful, it is critical that their results be consistent — that is that the same conclusions should be drawn by multiple examiners acting independently. To achieve this, there has been a push to bring the scientific method into the field, removing many of the intuitive processes which examiners currently rely on [28].

Chapter 3

METHODOLOGY/PROCESS

In brief, the process to be followed consists of discovering online credentials from acquired evidence, mapping those credentials to their corresponding services, extracting evidence from each service, authenticating and processing that evidence into a standardized representation format, and then performing the actual analysis [8]. Fig. 3.1 depicts this flow. Each step in this process is detailed below.

3.1 Initial Acquisition

As in any investigation, once the examiner has secured the evidence, the first step is to acquire a “forensic copy,” which for all purposes is an exact duplicate of the original [3, pp. 52]. Forensic copies serve as a protection for the original evidence since the examiner works with these instead of the originals, allowing them to perform analyses without the risk of compromising the integrity of the evidence.

During the initial acquisition of a hard drive using tools such as `dd` or EnCase, the data that is available is mostly limited to the structure of the drive as found in the Master Boot Record (MBR) or in one of the Volume Boot Records (VBRs). From these structures, the examiner can also extract additional information about the file system for a particular volume, but again this only provides structural information, such as which sectors on the disk store parts of a file. At this stage, there is no indication of where any information related to the suspect’s online activity and accounts may reside on the disk. For this reason, initial acquisition requires the additional steps of credential discovery, evidence mapping, and supplemental acquisition, as described in Chapters 3.2, 3.3, and 3.4, respectively.

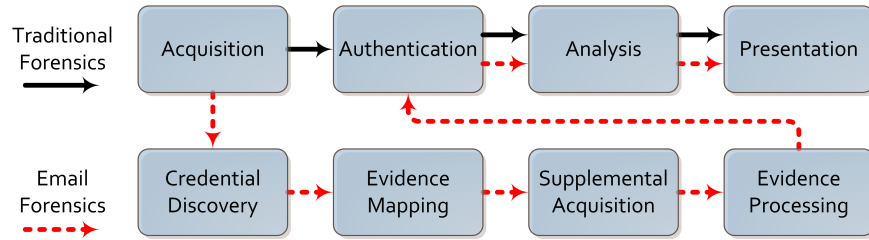


Figure 3.1: The traditional forensic workflow combined with the email approach

3.2 Credential Discovery

This process uses the term “credential” to denote any data which can identify the owner of the credential (e.g. the suspect) in some useful way. The breadth of this definition allows for its application without respect to the format in which the data is stored or the type of service to which it is mapped. Also, while other terms indicate a similar idea, such as “footprint” [29], “fingerprint”, and “profile”, none of these convey their purpose within the process, which is to reestablish a connection with online services to extract evidence. To this end the process defines a phase, “credential discovery”, in which practitioners attempt to detect credentials stored in a piece of evidence which an examiner can use to further recover additional evidence for the investigation.

These credentials must first be discovered within the files stored in forensic copy of created during the initial acquisition. The formats of credentials range from simplistic (text files containing user names and passwords) to complex (session cookies), and discovering all types of credentials will require an equally diverse set of approaches. Some possible credential discovery approaches include:

- **Brute force:** Given a set of criteria (such as a regular expression) for what may possibly be credentials, linearly search the evidence, including any file slack or bad sectors. This has the disadvantage of being neither intelligent nor efficient.

- **Search known locations:** Search for files known to regularly store credentials, such as key ring databases, cookie files or databases, registry entries, etc. While more efficient, this has a much narrower scope and may overlook legitimate credentials.
- **Heuristic-based:** Using machine learning or similar techniques, learn through past experiences and feedback from the examiner what constitutes a usable credential when investigating the raw data.

Practitioners may develop other approaches, and each approach may have varying levels of success on different data sets. As such, it may be necessary to use all available approaches on each data set, depending on the computational resources available. In the best case, a large-scale, distributed system would be utilized with as many approaches as possible including proprietary internal tools, remotely hosted tools, and open source tools to discover the largest possible set of credentials.

3.3 Evidence Mapping

Following the discovery of credentials, it is necessary to “map” them to a source of evidence before performing any further acquisition. To achieve this mapping, the discovered credentials must be reused to gain access to a remote service in either a “targeted” or “blind” approach, which entail testing these credentials against a service to which they are known to apply to or a service which they are not (to leverage the human tendency to reuse credentials across many services), respectively. In other words, this mapping determines what online service the suspect accesses using the credential. Depending on the approach taken to discover a set of credentials, the form in which they were stored, and any accompanying data stored with the credentials, the difficulty of the mapping process may vary. Examples include email addresses or

cookies which specify the domain to which they belong, spreadsheets organized in a manner which makes this information evident to an examiner, or a text file may store a user name and password with no indication of the service for which they are valid. Manual examination may be necessary to complete the mapping process if this is the case.

3.4 Supplemental Acquisition

After mapping a credential to a service, the next step is to acquire a forensically sound copy of the service's data. During the acquisition process, examiners must follow established best practices for any data source from which they extract evidence — this can be best achieved by defining an engine that utilizes modules which meet the requirements of *the rules of evidence* to acquire this data. To ensure the process is repeatable, the process treats this portion as a black-box engine which follows a set of steps to present the recovered data in a source-agnostic form so that the next module in the engine can process the evidence without regard to the source from which it originates. This engine should *reuse* the credentials previously discovered, *acquire* the most complete representation of the email (including headers and body), and then *store* them as a separate copy in an intermediate format for the purpose of evidence processing into a format which examiners will later use. Once these steps have become well defined, automation becomes trivial and should be implemented as a means to prove that the process is repeatable and forensically sound.

The nature of online storage requires careful consideration of data integrity and authentication issues; it is infeasible to represent the data exactly as stored at the remote location using this process; however, the focus of this process hinges on the text-based content of email evidence (including the headers and the body of the message) and not on the structure of the data stored on disk. As each email is a

discrete, individually identifiable piece of data, a checksum of the plain text content of the original form of an email message is the most useful check against data integrity. As the acquisition process is automated, repeatable, and the data yielded is verifiable using a hash, the evidence acquired in this phase is a forensic copy of the evidence in an intermediate format [8]. Such an intermediate format is required as the representation of emails in different systems will vary and influence the methods used to acquire them.

3.4.1 Acquisition Definitions

To help justify the use of certain acquisition methods during this phase, it is useful to draw parallels between different types of traditional forensic acquisition and the circumstances characterizing supplemental acquisition from online sources.

The two types of acquisition are “static” and “live”, which correspond to acquiring data from unchanging or volatile systems, respectively. Static acquisitions are carried out against a data source which is unchanging, for example, “. . . a computer seized during a police raid . . .” [3, pg. 106]. Conversely, live acquisitions are performed on systems in the process of running, often because “. . . the computer has an encrypted drive . . .” [3, pg. 106] or a live capture of memory may prove useful. When applied to performing forensics on an email account, a static acquisition is equivalent to acquiring data from stored Personal Storage Table (PST) files, frozen accounts, or logs from journaling or Simple Mail Transfer Protocol (SMTP) servers, whereas a live acquisition is equivalent to acquisition performed on active email accounts via Internet Message Access Protocol (IMAP) or using other methods, running servers (SMTP, journaling, etc.), or webmail services.

Further, two well-known approaches to acquisition are “logical” and “sparse”, each of which is useful in different situations. Logical acquisitions target “. . . only

specific files of interest . . .” [3, pg. 107], similar to if an examiner were to target only specific types of email archives (e.g. PST or mbox) or web services (possibly listed in the browser cache) which will typically be directly visible in the disk image. The traditional forensic approach is a clear example of a logical acquisition under this definition, as portions of the body of evidence may be overlooked. A sparse acquisition is more complete and includes the same data as a logical acquisition with the addition of “. . . fragments of unallocated (deleted) data,” [3, pg. 107] and for email forensics can be defined as extracting data from all sources identified in the credential discovery phase, which may not be stored on the disk in any way.

3.5 Evidence Processing and Authentication

To increase the value of the intermediate representation mentioned in the previous step, it is necessary to define a common evidence representation for the acquired data to be processed into, shifting the focus of analyzing this data from handling specific services to the task of email analysis itself. To simplify working with this data, the representation should retain metadata about the data source as well as the data itself, so as to clearly identify any specific characteristics of the data that would be important during the reporting process. Using a common representation also adds the benefit of being able to develop tools which treat the evidence in a source-agnostic manner, since the representation abstracts away the differences between webmail and locally stored emails, simplifying the development and validation/verification process of forensic tools.

A well-structured format lends itself to the above goals, allowing for easy searching, classification, and general use of the evidence while providing an extra layer of abstraction from the raw evidence to help maintain the forensic integrity of the information. When using a structured representation, an examiner can employ verifi-

cation techniques (such as schema verification) to prove the accurate representation of the evidence. Such a format and verification techniques also lends to the use of this process in a collaborative environment; for example, an organization may provide its implementation of an analysis tool remotely through a Service Oriented Architecture (SOA) implementation or multiple organizations may create separate, yet interoperable, tools using a common evidence representation.

Since the results of a forensic investigation must be accurate and verifiable, any data used during its execution must be able to be authenticated at any time. In order to properly authenticate the data after acquisition and processing, the evidence representation format used should store the checksum generated during the acquisition phase — such a checksum is created in practice by using cryptographic hash functions including MD5, SHA1, SHA256, etc. By storing this checksum, examiners will be able to confirm that the integrity of the data has not been compromised since it was first acquired (perhaps during transit or by intentional tampering), or if necessary and possible, they can perform a subsequent acquisition against the online service to check for changes to the available data or verify the accuracy of the first acquisition attempt.

3.6 Analysis

The next step after acquiring, processing, and authenticating the evidence is to perform forensic analyses that will be informative for the purposes of the investigation. Since the methodology only provides a process for the acquisition and storage of supplemental evidence, the implementation of new analysis tools is beyond the scope of this work. However, this methodology reduces the amount of effort required for analysis of online evidence in two ways and presents two techniques to be used in a successful analysis.

3.6.1 *Reduced Effort of Analysis*

First, the approach removes the need to manually acquire supplemental evidence as part of the examination workflow. The steps of discovering credentials, mapping them to online services, acquiring data from the service, and processing the data into a standard format are all performed automatically, saving time while providing increased breadth to the incident report.

Second, such a methodology specifies that the standardized data storage format should have a way by which to validate its structure. Three benefits arise from this requirement: 1) acquired data in a validated format gives tool developers confidence in the structure and type of the data; 2) developers do not need to write analysis tools to handle multiple formats, since it is possible to convert evidence to the format used in the process, making tools more reusable; 3) a comparison of the output from multiple tools allows for checking accuracy¹ or for evaluating performance.

With these benefits, the selected approach provides significant advantages in collaboratively discovering, collecting, and analyzing evidence stored by online services.

3.6.2 *Analysis Techniques*

When analyzing email, the most common approach is to use syntactic reasoning based on the headers of email evidence [24]. Analyses using this technique is based upon examining structural features of emails such as the fields defining who the sender and recipient(s) are. An examiner may, for example, notice that a suspect has recently been communicating with a known criminal and use this knowledge to focus their investigation on emails related to those activities. Recently, this has been achieved

¹As discussed in [30], validating forensic tools by comparing their output is important, but requires executing the tools against the same evidence.

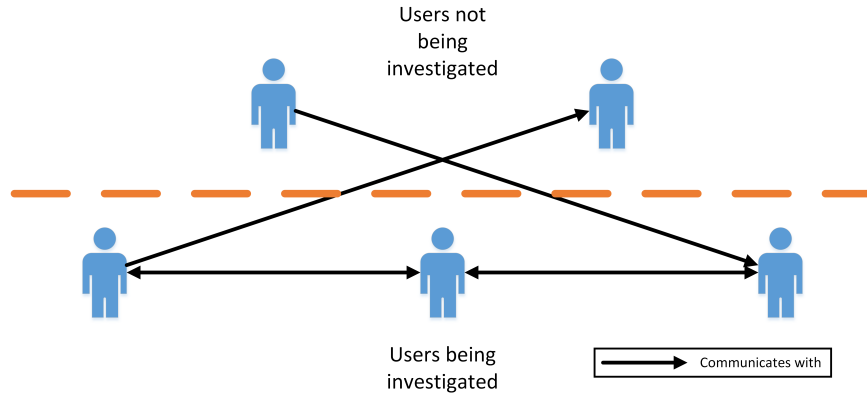


Figure 3.2: The Possible Complexity of Information Flows within Email

through the use of XML evidence containers [21] which can easily be queried using query languages such as XPath.

Another approach is the use of semantic reasoning, which uses computers to draw conclusions from evidence based upon the contents of fields in well-structured data. In email forensics, this may be used to query all emails whose body contains mention to specific topics or are addressed to known criminals. More importantly, when investigating multiple suspects this approach may be used to reveal information about the structure of or information flows within a group. For example, consider a situation in which three suspects are complicit in a crime though two of the suspects never communicate directly, as seen in Fig. 3.2 — if the examiner does not infer the connection between the two suspects who aren't making direct contact, blame may be placed solely on the intermediary. Using semantic reasoning tools, a query could be created which examines the connections between the senders and recipients of messages across the three distinct bodies of email evidence, creating a clearer picture of the course of events being investigated. In other cases, this same query may reveal that information could be flowing through another person outside of the group being investigated and point examiners towards new suspects who have evaded law enforcement thus far.

These two analysis techniques clearly provide a benefit with regards to the ease of the beginning phases of analysis and to increasing the thoroughness of the investigation.

Chapter 4

IMPLEMENTATION DETAILS

To demonstrate the framework, the details of a plugin-based forensics framework for online evidence, called PlugsE¹ will now be presented.

4.1 The PlugsE Framework

PlugsE is a framework implemented in Python meant to act as the black box engine mentioned in Chapter 3.4 consisting of separate modules to handle each step of the forensic process and a backbone which integrates them into a seamless tool. It has been developed with extensibility in mind, where adding a specific implementation of any step in this process is achieved by a system administrator manually adding entries to one of four JavaScript Object Notation (JSON) manifest files which specify to the PlugsE backbone the name of the module, the type of data (DFXML file, Google cookies, keyring file, etc.) it handles, as well as how to access the module from a programmatic standpoint. The access vector could be, for example, a command-line executable or a service available via a Remote Procedure Call (RPC) interface such as REST. PlugsE stores a manifest file for each step in the forensic process and parses them using the Python `json` library to create a vector table which the backbone uses to map the different types of data it is presented with to a specific implementation of a step. Through the use of these manifests, each step in the forensic process can be viewed as a collection of modules which implement differing approaches to the specific forensic task at hand.

¹Available at <https://bitbucket.org/jpaglier/plugse>

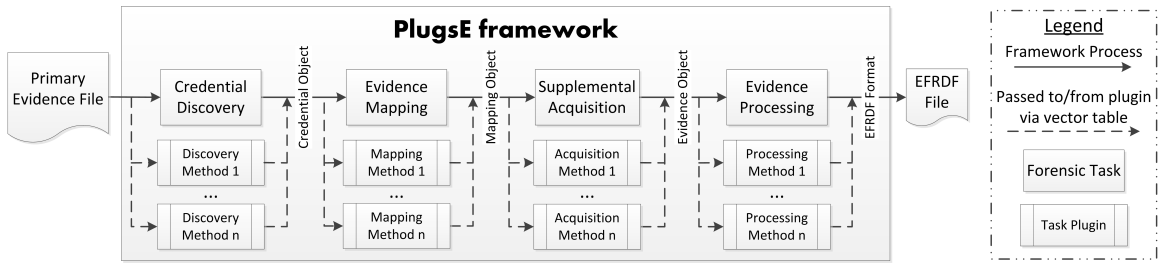


Figure 4.1: The PlugsE framework

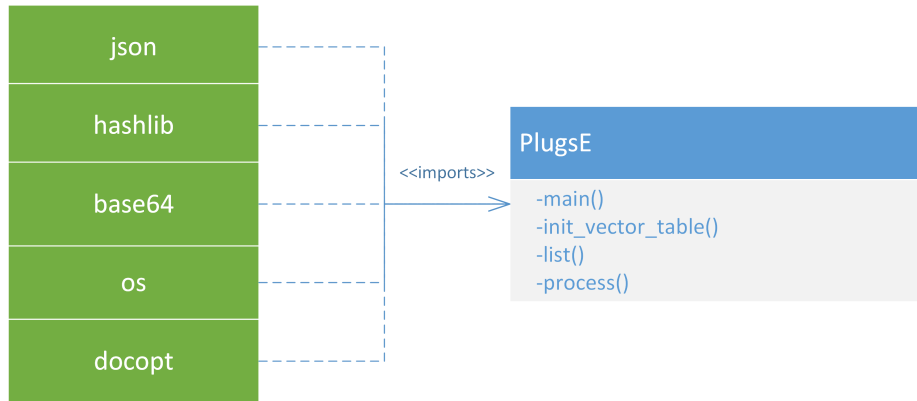


Figure 4.2: The Class Structure of the PlugsE Backbone

Each module must both accept as input and return as output JSON representations of the data being acted upon coupled with logging information (start/end times, check-sums, module name and version), which aids in providing a common representation of data within the system, facilitating interoperability of modules written by different developers, organizations, or even in distinct languages. Any binary data within the JSON representation is necessarily encoded into Base64 so that it may be represented as a string. In the case of the initially acquired data being passed into credential discovery modules, the PlugsE backbone utilizes Python’s `base64` library to do the encoding. A summary of PlugsE’s class structure can be seen in Fig. 4.2.

When executing, PlugsE is first started by an examiner running it while passing in the initially acquired data, in this example it is assumed to be a DFXML file along with a raw disk image acquired using the `dd` tool. Following, PlugsE will enumerate all existing modules from its manifest files based on which step the implement, creating

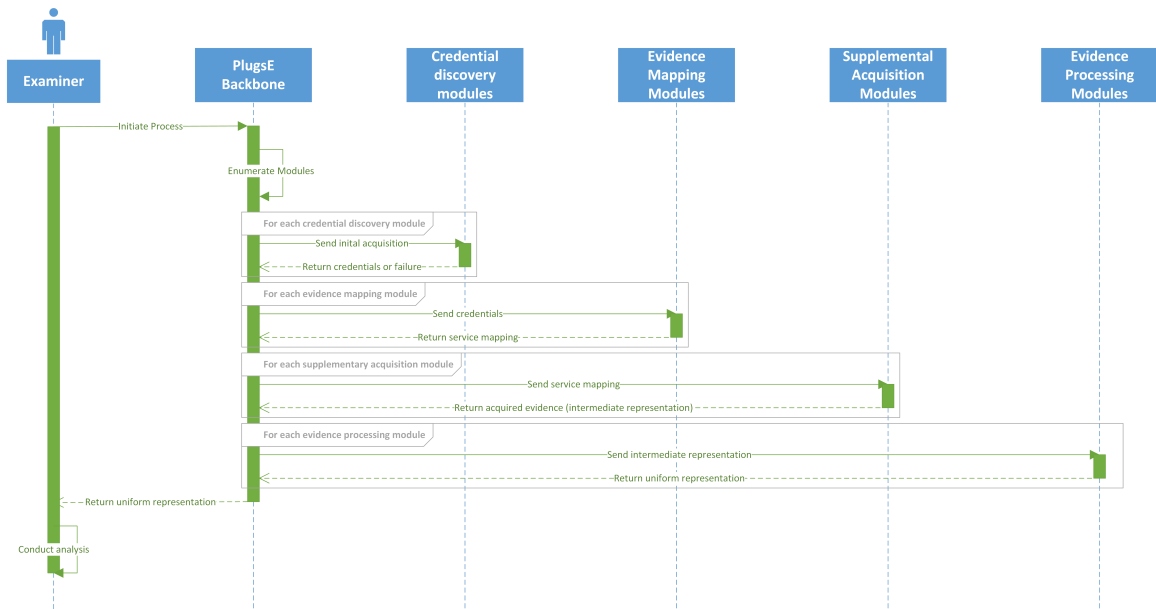


Figure 4.3: Sequence Diagram Showing the Information Flow Through the PlugsE Framework

the vector table. From here, PlugsE loops through each step in the forensic process, passing each module the types of data which it expects; this executes in a chain-like fashion where the results of one module are passed as inputs into the next. Finally, the uniform evidence representation of the email evidence is returned to the practitioner. This process can be seen in Fig. 4.3.

This modular approach offers a number of interesting benefits including that a developer can implement a number of data flows within the forensic process and a step in the forensic process may be offloaded to a remote server via RPC to a module provided by another organization in a SOA fashion, with the backbone and examiner being oblivious to the geographical location or implementation details of the web service. These qualities may benefit a distributed, collaborative approach to forensics, such as the one laid out in the CUFF framework [1]. Further, as modules implement only a single step in the forensic process, their structure can be greatly simplified, easing development and aiding in their validation and verification, as seen in Fig. 4.4.

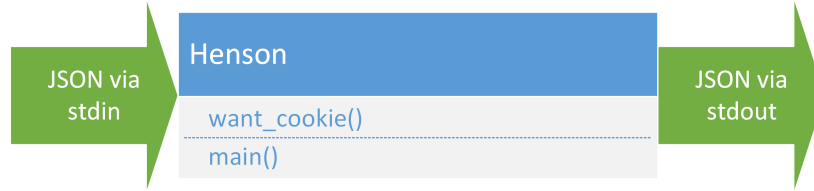


Figure 4.4: A High-Level Representation of Henson's Structure

```

C:\Windows\system32\cmd.exe - python plugse.py process "..\..\..\defense demo\demo.dfxml" d...
C:\Users\jpaglier\Dropbox\plugse\plugse\PlugsE\plugse>python plugse.py process "
..\..\..\defense demo\demo.dfxml" dfxml "..\..\..\defense demo\demo.raw" r
aw
finding credentials with Henson
handing credential to Crumbler

```

Figure 4.5: PlugsE's Command Line Interface Output

As a proof of concept, PlugsE's command line interface, as seen in Fig. 4.5, and the online evidence acquisition steps from Chapter 3 will be used to retrieve the contents of a Gmail account using cookies containing session information that is still valid.

4.2 Initial Acquisition

This implementation, makes the assumption that an examiner has already completed the work of initial acquisition (as described in Chapter 3.1) of a hard drive from a desktop computer and created a forensic copy. Ideally, this would be performed using a system such as the one presented in [1], which allows for the analysis of evidence to automatically begin after acquisition. As previously mentioned, this can be achieved through the use of tools such as dd or EnCase. Also, the modules in

Algorithm 1 A simple method that searches for a file by traversing the file system one directory at a time

```
rpaths ← {}  
for all q ∈ path_set do  
    rpaths ← rpaths ∪ { resolve(q) }  
end for  
for all p ∈ rpaths do  
    position ← root_dir ; not_found ← False  
    for all dir ∈ p do  
        if dir ∉ position then  
            not_found ← True  
            break  
        end if  
        position ← dir  
    end for  
    if not_found then  
        continue  
    end if  
    create_initial_mapping(p)  
end for
```

this implementation depend on the DFXML representation of the evidence, so the examiner (or the tools used by the examiner) must ensure its creation in this phase using the `fiwalk` tool.

4.3 Credential Discovery

With a forensic copy of the target device accessible, it is now possible to begin searching for credentials. By creating a PlugsE discovery module, Henson ², that searches for browser cookies utilizing a **Search known locations** approach, one easily discovers the cookies for the Chrome browser on a Windows machine at `%USERPROFILE%\AppData\Local\Google\Chrome\User Data\Default\Cookies`. While other browsers' cookies are also in known locations, this file is the focus of this proof of concept.

²Available at <https://bitbucket.org/jpaglier/henson>

Searching for the existence of a known path is a straightforward approach, taking as input a list of paths for which to search. First, it decodes all of the paths, meaning it resolves any Windows system variables to all matching explicit paths. Then it splits each path into its subdirectories and iterates through them, checking for their presence in a representation of the filesystem's structure created beforehand from the DFXML file. If the full path exists, this is recorded for later use. The complexity of Algorithm 1 is $O(n \cdot m)$, where $n = |\text{resolved_paths}|$ and $m = |\text{dir_contents}|$. It is assumed that the search space $|p|$ will be limited because it is searching for known paths of common programs, meaning it does not have the same capacity for expansion the way that n or m do. For example, in the case of Chrome cookie files in Windows 7, the path specified previously will resolve to `C:\Users\\AppData\Local\Google\Chrome\User Data\Default\Cookies`, which is a total of 9 directories before reaching the desired file. This step is achieved using `fiwalk`'s provided Python library, giving a programmatic interface to the files in the initially acquired storage media through the use of `FileObject` objects.

After discovering the cookie database, the last task the module performs is to store important information about the possible credential source in a JSON file for use in the Evidence Mapping phase. While the discovery module cannot map the credential to a service because it did not search the contents of the database for service-specific information, it stores the source, format, checksum, and time of discovery of the credential in a JSON file as illustrated in Fig. 4.6. With this, PlugsE's logging process has the information it needs and the relevant evidence mapping modules know which files to use when carrying out their discovery attempts. The module then returns the JSON file to the main PlugsE process which passes it to any modules registered for handling a cookie credential. Because of the modular approach of PlugsE, Henson

```
"start_time": "2013-09-10T16:07:00.853000",
"version": 1.0,
"end_time": "2013-09-10T16:07:01.426000",
"credentials": [
  {
    "checksums": {
      "sha1": "2a9d2f492f9d4ce4f902c2f0b4a80fb84b0a1d50",
      "md5": "b6ea39052c7e425b2743c385a1f006c9"
    },
    "errors": [],
    "contents": "...",
    "filename": "Users/pag/AppData/Local/Google/Chrome/User Data/Default/Cookies"
  },
  ...
]
```

Figure 4.6: An Abbreviated Sample of Henson Output

was implemented in 175 lines of code (including comments) and using simple, freely available libraries..

4.4 Evidence Mapping

Now that the cookies have been discovered, PlugsE invokes all evidence mapping modules registered to work with cookie databases, passing the possible credential sources to each of them. In some cases, it may be necessary at this point for the examiner to manually map the credentials to a service, as mentioned in Chapter 3.3. PlugsE will determine that this is the case when one of two things happens: 1) no mapping module has been registered to work with the source and format of a credential, or 2) none of the registered modules were successful in mapping it to a service. For this proof of concept, a tool named Crumbler³ was written in Python to attempt to map existing cookies to a live Gmail session. Crumbler imports the cookie database from its native SQLite format to a custom subclass of the common `Cookie` Python object.

In this example, identifying the cookies for a Gmail account is straightforward because the fields shown in Fig. 4.7 will be present in the cookie database passed to this module as input when opened using Python's `sqlite3` library. The mapping

³Available at <https://bitbucket.org/mmabey/crumbler>

module for PlugsE searches for these fields and uses the `cookiecutter` library to load them into a browser powered by the Selenium web driver ⁴. Using Selenium, the cookies are reused to attempt to gain access to the Gmail inbox — a successful access is counted as a detection. Upon detection PlugsE creates an entry in the mapping table which identifies this cookie database as containing credentials for Gmail. Fig. 4.8 shows what this entry looks like. Although the complexity of this module depends on the efficiency of the Python `sqlite3` ⁵ library, it only searches for a constant number of keys in the target and stops searching when any key is not present, which means it contributes no more than $O(1)$ complexity to the library’s operations.

Interestingly, Gmail allows users to grant other Gmail addresses access to their account without reentering their password. The account that is granted access is called a “delegate” and can read all the emails in the granter’s account as well as send emails on their behalf. While Gmail does not provide IMAP access to delegate accounts, creating the delegate prolongs access to the target account past the two week expiration date of the cookies, allowing for any needed follow-up acquisition. Because of this, as long as the cookies are still valid Crumbler performs each of the steps for adding a delegate as outlined in the Google help pages ⁶, which takes $O(1)$ time. Once this process has completed, the grantee can access the target account by logging in to Gmail, clicking on their email address in the top right hand corner of the screen, and selecting the target account from the list of accounts to which they have access.

creation_...	host_key	name	value	path
13003520...	accounts.google.com	GALX		/
13003520...	.accounts.google.com	_utma		/
13003520...	.accounts.google.com	_utmb		/
13003520...	.accounts.google.com	_utmc		/
13003520...	.accounts.google.com	_utmz		/
13003520...	mail.google.com	S		/mail
13003520...	accounts.google.com	GAPS		/
13003520...	accounts.google.com	RMME		/
13003520...	.google.com	NID		/
13003520...	.google.com	SID		/
13003520...	accounts.google.com	LSID		/
13003520...	.google.com	HSID		/
13003520...	.google.com	SSID		/
13003520...	.google.com	APISID		/
13003520...	.google.com	SAPISID		/
13003520...	.mail.google.com	GX		/mail
13003520...	mail.google.com	GMAIL_AT		/mail
13003520...	.google.com	PREF		/
13003520...	mail.google.com	gmailchat		/mail

Figure 4.7: All the cookies created by logging in to Gmail

4.5 Supplemental Acquisition

As mentioned in Chapter 3.4, tool developers must determine the best practice for acquiring data stored by a distinct online service based on the type of credential(s) discovered previously and the service’s features. For this proof of concept with Gmail, it had to be discovered what features are apparent when only the browser cookies are available to be used to log in. While the optimal acquisition method for retrieving a copy of all emails is to do so via IMAP, cookies are specific to the HTTP protocol and will not work to authenticate through IMAP. If plain text credentials (user name and

⁴<http://seleniumhq.org/>

⁵<http://docs.python.org/2/library/sqlite3.html>; complexity of individual operations not provided.

⁶<http://support.google.com/mail/answer/138350>

```
"start_time": "2013-10-10T18:14:00.942000",
"version": 1.0,
"end_time": "2013-10-10T18:27:21.626000",
"credentials": [
  {
    "checksums": {
      "sha1": "2a9d2f492f9d4ce4f902c2f0b4a80fb84b0a1d50",
      "md5": "b6ea39052c7e425b2743c385a1f006c9"
    },
    "errors": [],
    "contents": "...",
    "service": "gmail"
  },
  ...
]
```

Figure 4.8: An Abbreviated Sample of Crumbler Output

```
Delivered-To: burdenedreflect@gmail.com
Received: by 10.220.174.193 with SMTP id u1csp57274vcz;
      Fri, 8 Nov 2013 13:35:06 -0800 (PST)
X-Received: by 10.224.32.66 with SMTP id b2mr27904352qad.80.1383946506094;
      Fri, 08 Nov 2013 13:35:06 -0800 (PST)
Return-Path: <update+o6hhc6of@facebookmail.com>
Received: from mx-out.facebook.com (outmail010.ash2.facebook.com. [66.220.155.144])
...
```

Figure 4.9: An Abbreviated Sample of Spatula’s Intermediate Representation

password) were discovered, acquisition via IMAP would be possible. Further, since only the browser cookies are available to work with in the proof of concept, there is a limited ability to change any account settings that will help the process of acquisition.

The final challenge to acquiring data from Gmail is that the only method for retrieving the raw email data is to essentially “screen scrape” the pages returned during a web session, parsing through the HTML and using regular expression patterns or searching through the Document Object Model (DOM) for the desired elements. A tool, Spatula ⁷, was developed for the purpose of downloading the contents of a Gmail account, again using Selenium. This tool navigates into each email present in the user’s account and uses the “Show Original” option to provide an mbox representation of the evidence. This view is then stored as mbox, which is an intermediate representation as mentioned in Chapter 3.4 as seen in Fig. 4.9. The output is then returned via `stdout` — a sample is shown in Fig. 4.10. Special care had to be taken when developing this

⁷<https://bitbucket.org/jpaglier/spatula>

```
"start_time": "2013-09-26T09:11:00.029000",
"version": 1.0,
"end_time": "2013-09-26T09:40:01.728000",
"credentials": [
  {
    "checksums": {
      "sha1": "9c35cb5e1b21a98f7285d6565e8e3e8cf838a0",
      "md5": "5c7e4b81e8a7a89ecb3ee5656db133d1"
    },
    "errors": [],
    "mailbox": "ICAgICAgICAgICAgICAgICAgICAgICAgICAg..."
  },
  ...
]
```

Figure 4.10: An Abbreviated Sample of Spatula’s Output

tool, as it was discovered that Gmail limits the number of requests you can make to a single account within a certain time period; too much activity is flagged as “unusual” and results in a lockout period anywhere from one to twenty-four hours. As such, a mandatory wait period is enforced in between each page of email scraped to decrease the average rate of outgoing requests. The messages should then be processed into a standard format, as discussed in the following section.

While it is clear that a few circumstances have to be ideal in order for this acquisition process to work, namely that the owner of the credentials is always signed in, that the cookies have not yet expired and are discoverable by some means, and that the notification banner of having added the delegate account will not compromise the investigation. It is inevitable to retain these circumstantial dependencies. However, one might also assert that those investigations for which they do not hold have not lost access to evidence that otherwise would have been accessible, while those for which they do hold have gained access to a great source of information ⁸.

⁸Either way, to achieve comprehensive forensic analysis on email evidence, it is believed such an approach is necessary and beneficial.

4.6 Evidence Processing and Authentication

To demonstrate the evidence processing phase, the framework must convert our intermediate representations to a well-structured format which follows the current best practices of the forensic process.

4.6.1 Evidence Representation

During the development of this proof of concept, a survey of the strengths and weaknesses of existing formats was conducted and it was concluded that the mbox format [31] was the best suited to the purposes of the process. The mbox format is a flat-file, plain text representation of email which is easy to parse and human readable; these traits greatly reduce the time needed for examination of evidence and development of tools, both of which are costly in terms of resources and time. Furthermore, normally mbox stores attachments in some form of directory structure related to their messages so that attachment analysis could be started as part of an automated process, separate from the email data. Using mbox is also useful even when processing the common PST format as tools from libPST⁹ provide the conversion. Finally, mbox is valid for use as a forensic copy format as it is “. . . output readable by sight, shown to reflect the data accurately . . .” and thus “. . . is an original” [3, pg. 162], so long as the acquisition process used was sound.

A current trend in digital forensics is the use of XML as a data representation format, allowing for a firm layer of abstraction “. . . between feature extraction and analysis . . .” and “. . . a single, XML-based output format for forensic analysis tools . . .” [21]. For evidence representation in existing methodologies, DFXML is a standard to represent disk, partition, file system, and file data in a unified manner [22].

⁹<http://www.five-ten-sg.com/libpst/>

A major benefit of DFXML is the generality of its representation; regardless of disk geometry or forensic copy format, the evidence is represented in the same manner.

When analyzing email evidence, the most significant metadata is contained within the header of the email itself. Email headers include information such as the sender and recipient (**From** and **To**), unique message identifiers (**Message-ID**), reply addresses (**Reply-To**), and more [32]. Even without considering the content or body of emails, these headers have been shown to be useful in forensic investigations as a means to achieve author attribution, detect attempts to obfuscate sequences of events during a time period of interest [24], as well as identify communication flows and perform social networking analysis.

Although DFXML is quite efficient for representing massive amount of data from a filesystem, it is ill-suited for storing the header information of emails, as file system metadata is not closely related to the analysis of evidence contained within email messages. While it could be extended to fit this purpose, the resulting format would become overly encumbered and its size efficiency greatly reduced. Because of this two new representations which are more suitable for email forensics, but maintain some of the standard elements introduced in DFXML (such as byte runs of discrete pieces of evidence) were defined. These new formats are known as Email Forensics XML (EFXML) and Email Forensics Resource Description Framework (EFRDF). While these two representations are functionally equivalent, they were designed with different purposes in mind. While EFXML lends itself mostly to syntactic reasoning methods, EFRDF is meant to be used when examining the highly semantic nature of information contained within email evidence, as discussed in Chapter 3.6.2. More importantly, investigations utilizing multiple bodies of email evidence which, when combined, may reveal complex information flows through social interactions may benefit from semantic

```
(126FB=3553193)(126FC=55914899)(126FD=30f2)(879D=Ubuntu Webserver)
(879E=4FBBFD99.7030003@gmail.com)(879F=4fbbfd99)(87A1
=1402698178870255137)(126FE=3556285)(126FF=55927429)(87A2
=4FBBFDF5.1020909@gmail.com)(87A3=<4FBBFD99.7030003@gmail.com>)
(87A4=4fbbfdf5)(87A6=1402698281367293275)(12700=3557382)(12701=55931778)
(87A7=4FBBF6D5.4060203@heypete.com)(87A8=4fbbf6d5)(87AB
=1402699518649060034)(12702=3558521)(12703=55936289)(9AE5=16f2)
(87AC=CAJJCUitGw3eJVdyK5o1XDuH=n0a_gg2UTGVaH6U8L9kk0+Ua3w@mail.gmail.com)
(87AD=<4FBBFD99.7030003@gmail.com> <4FBBFDF5.1020909@gmail.com>)(87AE
=4fbbf6fc)(87B1=1402699557081490979)(12704=3559eb2)(12705=55942834)
(87B2=4FBC07DD.1040100@gmail.com)(87B3
```

Figure 4.11: A Sample of a Thunderbird Mail Summary File

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<mailbox type="mbox">
  <message>
    <Subject>&lt;Re: problem with dovecot&gt;</Subject>
    <Date>
      <year>2003</year>
      ...
    </Date>
    <From>
      <sender>
        <alias>&lt;Jesse Peterson&gt;</alias>
        <email>jpeterson275@attbi.com</email>
      </sender>
    </From>
    <non-standard>
      <X-Original-To>dovecot@proctrol.fi</X-Original-To>
    </non-standard>
    <byte_runs file_offset="1101673" len="2159"/>
    <checksum>
      <md5>8e4bb7462b991183cf5b2adc87970227</md5>
    </checksum>
  </message>
</mailbox>
```

Figure 4.12: An Abbreviated Sample of the EFXML Format

analysis. EFRDF is based upon the Resource Description Framework standard [33], a subset of XML often used for semantic representations of data.

As email is text-based and can be easily represented in XML without complicated encoding, EFXML and EFRDF present many of the same benefits presented by Alink et al. [21] and Garfinkel [22], including easy searching and classification of information. Non-XML indexes of email metadata do exist in the current software ecosystem, but are not tailored to use in email forensics. For example, the Thunderbird email client stores metadata as Mail Summary Files (MSFs) in a format known as Mork.

```

<?xml version='1.0' encoding='ISO-8859-1'?>
<mailbox:Class>
  <mailbox:type>mbox</mailbox:type>
  <message:Class>
    <message:Subject>&lt;Re: problem with...&gt;</message:Subject>
    <message:Date>
      <Date:year>2003</Date:year>
      ...
    </message:Date>
    <message:From>
      <From:sender>
        <sender:alias>&lt;Jesse Peterson&gt;</sender:alias>
        <sender:email>jpeterson275@attbi.com</sender:email>
      </From:sender>
    </message:From>
    <message:non-standard>
      <non-standard:X-Original-To>...</non-standard:X-Original-To>
    </message:non-standard>
    <message:byte_runs file_offset="1101673" len="2159"/>
    <message:checksum>
      <checksum:md5>8e4bb7462b991183cf5b2adc87970227</checksum:md5>
    </message:checksum>
  </mailbox:message>
</mailbox:Class>

```

Figure 4.13: An Abbreviated Sample of the EFRDF Format

However, this indexed data is neither human readable nor easily parsed [25] and is of limited use, as seen in Fig. 4.11. As an added benefit of using an XML-based format, EFXML and EFRDF have clearly defined schemas which can verify the output from tools that generate these formats. As an example of their differing, yet equivalent, representations, a To field reading:

```
To: jsmith@gmail.com
```

would yield the EFXML element:

```

<To>
  <recipient>
    <email>jsmith@gmail.com</email>
  </recipient>
</To>

```

or the EFRDF element:

```
<message:to>
  <to:recipient>
    <recipient:email>
      jsmith@gmail.com
    </recipient:email>
  </to:recipient>
</message:to>
```

which accurately represent the “To” header field in a much more structured manner, allowing for easily focusing on or excluding messages based upon their apparent recipients. Similarly, to reflect the data extraction capabilities provided by the “byte_runs” element in DFXML, it includes a simplified element which details the span of bytes within the mbox file where the email resides and can be extracted from using tools such as those designed for DFXML, the Unix command `dd`, or other comparable programs. While line numbers would be equally useful with regard to the mbox format, it was decided to use the “byte_runs” field in each representation to follow existing standards. Abbreviated samples of the EFXML and EFRDF representations of a mailbox can be seen in Figs. 4.12 and 4.13, respectively.

4.6.2 Evidence Processing

This proof of concept includes two tools for parsing mbox files into corresponding EFXML and EFRDF representations, named `mbox2efxml`¹⁰ and `efxml2efrdf`¹¹. `mbox2efxml` was loosely based on Philip Guo’s `create_mbox_summary` tool [34], but with comprehensive support for the email headers specified in [32] and well-formed XML through the use of the `lxml` library¹².

¹⁰Available at <https://bitbucket.org/jpaglier/efxml>

¹¹Available at <https://bitbucket.org/jpaglier/efxml2efrdf>

¹²<http://lxml.de/index.html>

Algorithm 2 The process mbox2efxml follows to parse mbox files

```
procedure MBOX2EFXML(mbox_paths)
  known_headers  $\leftarrow$  RFC4021-Headers
  efxml = {}
  for all mailbox  $\in$  mbox_paths do
    for all message  $\in$  mailbox do
      headers  $\leftarrow$  extract_headers(message)
      for all header  $\in$  headers do
        if header  $\in$  known_headers then
          efxml  $\leftarrow$  efxml  $\cup$  process(header)
        else
          efxml  $\leftarrow$  efxml  $\cup$  header
        end if
      end for
    end for
  end for
  return efxml
end procedure
```

Because the mbox2efxml tool only works with mbox archives, it assumes that if the email was originally in a different format, some tool has already made the conversion to mbox. For example, libPST's readpst¹³ tool reads in a PST file and outputs a separate mbox file for each folder contained within the PST. These separate files are then iterated over and each message within them is processed. This tool runs in $O(m \cdot n \cdot q)$ time where $m = |\text{mbox_paths}|$, $n = |\text{messages}|$ for all mailboxes, and $q = |\text{headers}|$ for all messages, as outlined in Algorithm 2.

The efxml2efrdf tool works in a similar fashion to mbox2efxml by reading in an EFXML file using the lxml library and converting the contents to its EFRDF equivalent. While leaving out the body of the email may limit the scope of a semantic reasoning approach like was discussed in Chapter 3.6.2, explorations into the costs and benefits of its exclusion are left to future work, where further investigation will be able to discover the subtleties of a semantic approach. The structure of EFXML and EFRDF as specified in their schemas helps overcome one shortcoming of mbox as

¹³<http://www.five-ten-sg.com/libpst/rn01re01.html>

it relates to forensics, which is that there is no way to add metadata to the file. Of particular interest are the fields which help maintain the chain of custody by storing information on the name of the program that created the mbox, EFXML, and/or EFRDF files; the version of the programs; the date and time of their creation; the target email address; the size of the mbox file; and MD5 and SHA1 checksums for the mbox file. With this information, it is possible to keep track of how the evidence was acquired, authentication information for the entire set of emails ¹⁴, and what programs handled the evidence at what time, all of which are required by the rules of evidence.

4.7 Analysis

This approach creates well-defined, structured, and verifiable representations of email data. Since they are XML formats, developers can easily craft tools and validate them using common XML parsing libraries to facilitate the analysis process, much like DFXML. Also, with the intermediate mbox format and the EFXML/EFRDF abstractions, forensic analyses using syntactic or semantic analysis techniques can be carried out while the forensic copy remains intact, regardless of the data source's original storage format. The development of analysis tools remains as future work.

¹⁴Unless the target account has been frozen by the provider, acquiring emails from the same account at two different times will likely yield two distinct data sets, preventing the checksums from matching. As such, the checksums are provided for integrity checks against the same mbox archive to ensure it does not change while being analyzed and not against past or future acquisitions.

Chapter 5

EVALUATION

To demonstrate the value of this approach and the proof of concept implementation, a number of experiments were run and their quantitative measures will be presented. First, the comprehensiveness of the approach will be measured by quantifying the number email messages acquired during a traditional forensic investigation and one using PlugsE, which implements this methodology. Second, the extensibility of PlugsE will be shown through an example where a manifest file is updated to instruct the framework to use a tool newly introduced into the system. The uniformity of the evidence container formats will be evaluated by comparing the Gmail representation of an email with the internal representation created when Thunderbird retrieves the same message using IMAP. The value of the approach and framework with respect to collaboration and distribution of work will be measured by examining its usefulness in an existing collaborative forensics framework known as CUFF [1]. The consistency of the approach will be measured by comparing multiple acquisitions of evidence against a source where known changes have been made, allowing for the effects of such changes to be measured and enumerated. Finally, the performance of modules will be measured when running in real computer environments. These measures accurately represent the effectiveness of this work with respect to the problems presented in Chapter 1.2.

5.1 Comprehensiveness

Since a major goal of this work is to provide more comprehensive view of email evidence, an evaluation of this trait of the approach must be presented. Given a system containing:

Email Evidence Acquired through Traditional Forensics

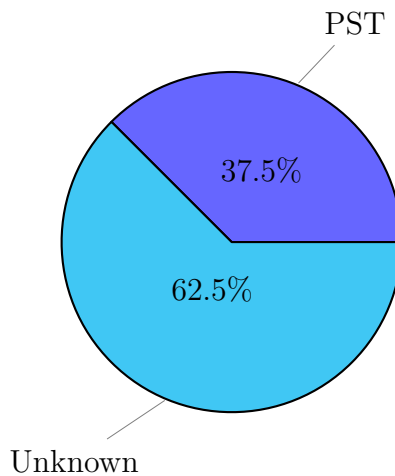


Figure 5.1: The coverage of email evidence provided by traditional forensics

- A PST file which holds 1,500 emails
- Credentials mapping to:
 - A Gmail account which holds 2,000 emails
 - A Yahoo account which holds 300 emails
 - A Hotmail account which holds 200 emails

for a total of 4,000 emails, using traditional forensics only 1,500 emails are directly available in the evidence, representing only 37.5% of the evidence, as seen in Fig. 5.1. By using PlugsE to achieve comprehensive results, a further 62.5% of the evidence was recovered as seen in Fig. 5.2.

As there is greater coverage of email evidence from this example, it is clearly shown that this process gives a more comprehensive approach to email forensics.

5.2 Extensibility

PlugsE was designed with extensibility in mind and its implementation reflects this strongly. At runtime, the PlugsE backbone parses four JSON manifest files to

Email Evidence Acquired through PlugsE

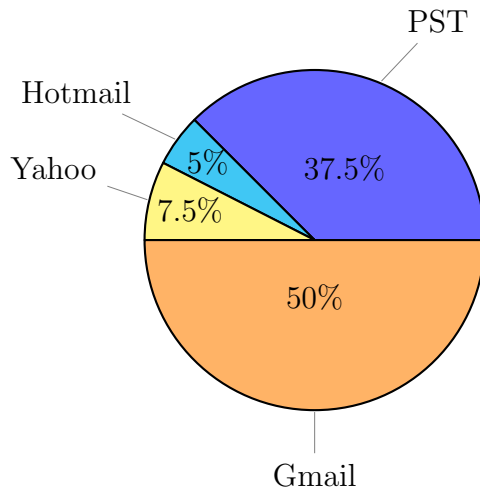


Figure 5.2: The coverage of email evidence provided by PlugsE's comprehensive process

load the plugins into the system, as described in Chapter 4.1. Each entry in this manifest file contains the tool name, the types of input that are required for this tool to run, the types of input that are optional for this tool to run (perhaps to increase the breadth of its results), the output type of the tool, whether it runs as a local command or remotely, the name of any required interpreters, and the access vector for the program (a URL or path to a program). Once these fields have been added to the manifest, PlugsE will automatically pass the proper inputs to it if all of the required input types are retrieved from the previous step. An example manifest entry can be seen in Fig. 5.3.

The use of the manifest files allows for system administrators to simple handle the installation of tools and basic configuration editing instead of needed to know the intricate implementation details of the PlugsE backbone itself, which would require programming knowledge. In this way, PlugsE is a pluggable framework with a focus on extensibility.

Since it has been shown that PlugsE addresses the need for extensibility in an environment handling email forensics, a need to represent the comprehensively acquired


```


{
  "Henson":
  {
    "version": 1,
    "required_input": ["dfxml", "raw"],
    "optional_input": [],
    "output_type": "chrome_cookie",
    "run_type": "cmd",
    "interpreter": "python",
    "access_vector": "/usr/bin/henson.py"
  }
}

```

Figure 5.3: An Entry in a Manifest File

[oss-security] CVE Request: additional fix for CVE-2012-2825 libxslt crash

 Inbox x oss-security x

 **Marcus Meissner** <meissner@suse.de>
to OSS ▾

Hi,

Our QA found that the reproducer in CVE-2012-2825 (magic.xsl and magic.xml) also expose another libxslt crash in older libxslt versions.

https://bugzilla.novell.com/show_bug.cgi?id=849019

This bug was fixed in libxslt 1.1.25 with this commit:
<https://gitorious.org/libxslt/libxslt/commit/7089a62b8f133b42a2981cf1f920a8b3fe9a8caa>

commit 7089a62b8f133b42a2981cf1f920a8b3fe9a8caa
Author: Martin <gzlist@googlemail.com>
Date: Wed Sep 16 19:02:16 2009 +0200

Crash compiling stylesheet with DTD

* libxslt/xslt.c: when a stylesheet embeds a DTD the compilation
----- could not determine the

Figure 5.4: Gmail’s Presentation of Email — Internal Representation is Unknown

data uniformly arises when handling emails from many sources which each handle email differently.

5.3 Uniformity

As mentioned in Chapter 3.4, different web services and email clients will represent email in differing ways, both internally and externally. For example, Gmail and

```
Delivered-To: jpaglier@email.asu.edu
Received: by 10.227.204.7 with SMTP id fk7csp144504wbb;
      Tue, 5 Nov 2013 04:50:50 -0800 (PST)
...
Date: Tue, 5 Nov 2013 13:50:09 +0100
From: Marcus Meissner <meissner@suse.de>
To: OSS Security List <oss-security@lists.openwall.com>
...
Subject: [oss-security] CVE Request: additional fix for CVE...
...
Hi,

Our QA found that the reproducer in CVE-2012-2825...
```

Figure 5.5: Internal Mbox Representation of Email Used by Thunderbird

Thunderbird will represent emails differently as seen in Figs. 5.4 and 5.5. These differing representations may pose a barrier to a forensic investigation in the sense that the task of email analysis may become tightly coupled with the representation being handled. To mitigate this hurdle, a uniform evidence representation must be defined to shift the focus away from the representation of email and towards the actual analytical methods being employed. To achieve this, the use of EFXML and EFRDF to represent that data can be shown to be beneficial.

After supplemental acquisition, the evidence processing modules in PlugsE convert intermediate representations of email (which are influenced by the best practices established for supplemental acquisition of a given service) into EFXML/EFRDF representations. This yields the same representation no matter the source of the data, as seen in Fig. 5.6.

Now that a uniform evidence container is used to represent emails from various sources has been shown to be useful, the benefit of aiding in collaboration and distribution of forensic tasks can be explored.

5.4 Collaboration/Distribution

Forensic investigations are becoming increasingly collaborative in a number of ways — multiple organizations may be conducting a joint investigation with a need

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<mailbox type="gmail">
  <message>
    <Subject>&lt[oss-security] CVE Request: ...&gt;</Subject>
    <Date>
      <year>2013</year>
      ...
    </Date>
    <From>
      <sender>
        <alias>&lt;Marcus Meissner&gt;</alias>
        <email>meissner@suse.de</email>
      </sender>
    </From>
    <To>
      <recipient>
        <alias>&lt;OSS Security List&gt;</alias>
        <email>oss-security@lists.openwall.com</email>
      </recipient>
    </To>
    ...
  </message>
</mailbox>
```

Figure 5.6: uniform Representation from Either Data Source

to share evidence and workload, proprietary tools may be shared using a service-oriented architecture approach (such that the implementation details of their tools aren't revealed), and the combination of those two issues lends itself to cloud based implementations of forensic platforms such as CUFF [1]. This architecture provides a unique approach to collaborative and distributed forensics in which stored evidence can be shared across organizational boundaries and forensic tasks are distributed among cloud nodes to provide scalable collaborative forensics. However, CUFF has not yet been tailored to a specific use case and tools must be developed in order to leverage it. In an environment like CUFF, PlugsE would be extremely powerful as it can easily leverage the distributed nature of the system. While the PlugsE backbone may reside on a single Analysis Node Instance, different steps in the forensic process (i.e. collections of PlugsE modules) may reside on other nodes or specific tools may only be available on certain Analysis Node Instances due to licensing or hardware issues. As PlugsE manifests are easily updated by a system administrator without programming knowledge, PlugsE can easily be managed within such a dynamic system. Furthermore, the ability of PlugsE to use RPC to communicate with modules that are not run

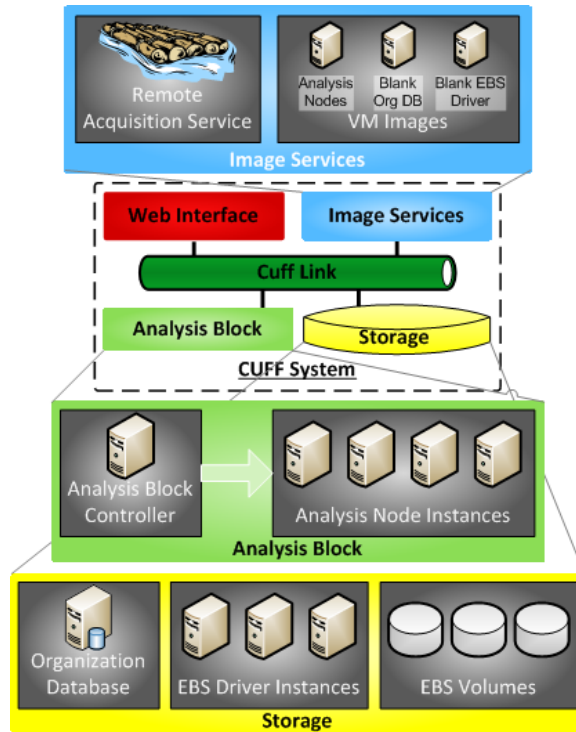


Figure 5.7: An Overview of the Cuff Architecture [1]

locally, the framework can easily be used in such a cross-organizational cloud-based environment. An added benefit of this RPC approach is that modules not deployed within a CUFF system may still be used by invoking tools in other systems using the same RPC methods. The use of uniform, language-agnostic representations and standard web communication models grants a great deal of freedom when designing a collaborative or distributed forensic environment.

PlugsE’s pluggable architecture and use of a uniform evidence representation adds the clear benefit of collaboration and distribution of labor among actors in a forensic investigation. While this is extremely useful, due to the rules of evidence later discussed in Chapter 6.2, any approach must provide a consistent representation of the evidence so there is no question about the authenticity, admissibility, completeness, and reliability of the evidence.

Table 5.1: Set of checksums after emails have been deleted and/or received

Operation	Set of Checksums
First acquisition	ab357d3e9cfda62d51d9e39f8f1cad3d fe41c3cd42318d0ba21f75eb9b72cc5f 84964e521a39865bb138c0046b3af45f
Email deleted	ab357d3e9cfda62d51d9e39f8f1cad3d fe41c3cd42318d0ba21f75eb9b72cc5f 84964e521a39865bb138c0046b3af45f
Email received	ab357d3e9cfda62d51d9e39f8f1cad3d fe41c3cd42318d0ba21f75eb9b72cc5f 84964e521a39865bb138c0046b3af45f aaa59f984ba63e54f493c77b830e99ed
Email both deleted and received	ab357d3e9cfda62d51d9e39f8f1cad3d fe41c3cd42318d0ba21f75eb9b72cc5f 84964e521a39865bb138c0046b3af45f 66efa0786721811f6e04fb6d62c1b6d4

5.5 Consistency

The well-defined, process-driven nature of the approach presented in Chapter 3 and the automated nature of the PlugsE framework presented in Chapter 4 provide a consistent approach to email forensics, which holds great weight with regards to the rules of evidence, particularly in repeatability. Since PlugsE also takes checksums of the evidence which is acquired, during subsequent acquisitions it becomes simple to check if the state of the data source has changed as the checksums will not match. To evaluate this property of the system, an experiment was devised in which the elements of the set of MD5 checksums of email messages acquired during multiple acquisitions of the same data source were compared when an email was deleted, an email was received, and finally when one email was deleted and another was received, as seen in Table 5.1. From this information, it became trivial to determine which changes had been made

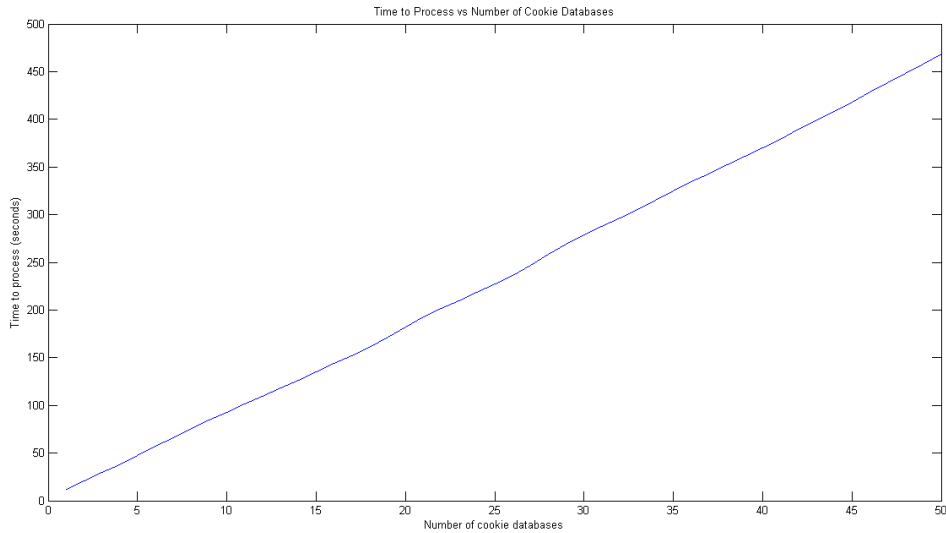


Figure 5.8: Time to discover known cookies in Chrome cookie database

to the system using basic reasoning. When the emails in a subsequent acquisition were a proper subset of those in the first supplemental acquisition, emails had been deleted and the converse signifies that new emails had been received. When neither case were true and the checksums still did not match, assuming the supplementary acquisition module functions properly, it could be inferred that emails have both been deleted and received. This information may be especially useful during intelligence operations where surveillance is the goal of the investigation rather than ex-post-facto examination.

Since the automation provided by PlugsE provides a consistent representation of a user’s email activities, the final evaluations needed are those related to the performance of the approach previously discussed.

5.6 Performance

As differing implementations of each step of the forensic process will vary based on data source, this evaluation will focus mainly on two factors: (1) the efficiency of

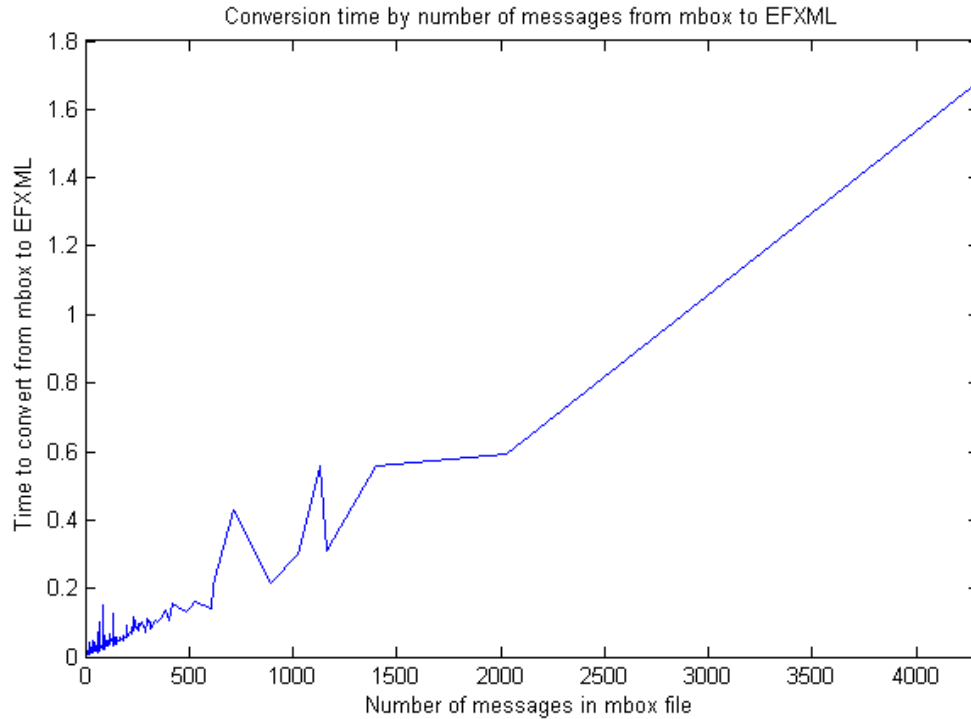


Figure 5.9: Conversion times of single mbox files in the Enron data set by number of messages

EFXML and (2) the running time of sample implementations of each step to show an example of the process functioning in a useful manner.

Initially, a test of the **Search known locations** algorithm was tested and was confirmed to be roughly linear, taking about 2 milliseconds for every 100 file objects listed in a DFXML record. When the process was carried through to evidence mapping, another 10 seconds per discovered Chrome cookie database containing Gmail cookies was added.

To evaluate the Email Forensics XML format and the evidence processing step, we designed an experiment which compares the size of Personal Storage Table (PST) files, their respective mbox files, and their EFXML representations. The PST files used were the publicly available Enron Corporation email data ¹, which measures 8.70GB

¹Acquired from <http://www.enrondata.org>

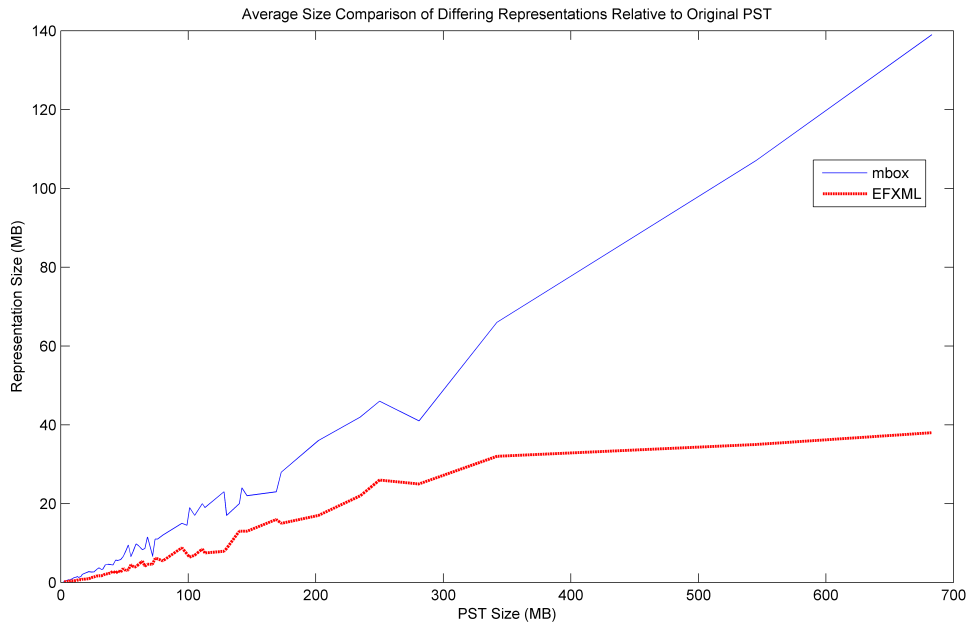


Figure 5.10: Average Size of mbox and EFXML Representations Relative to PST in the Enron Data (Normalized)

across 148 mailboxes, containing a total of 517,431 messages and 3,299 folders (once decompressed). This dataset does not include attachments or certain redacted portions of the original data. Each PST file was then converted into its respective mbox representation using the readpst tool available through libPST, yielding 1.2GB of text data. Then these mbox files were processed using the mbox2efxml tool discussed in Chapter 4.6 generating 614MB of data after approximately 3.5 minutes of processing on an Acer Aspire 4830T ².

When recording the time taken to process each mbox file individually, as seen in Fig. 5.9, the average processing time per message was 0.57ms. Fig. 5.9 also shows a roughly linear increase in processing time with regard to the number of messages, confirming the estimate of $O(m \cdot n \cdot q)$ running time. A comparison relative to mbox size, after conversion from PST, is depicted in Fig. 5.10, which visually demonstrates

²http://www.cnet.com/laptops/acer-aspire-timelinx-4830t/4505-3121_7-35029979.html

that the file sizes are positively correlated across this dataset. There was a notable decrease in file size when converting from PST into mbox, which may be explained by the block allocation scheme used in the PST format.

Following the step of processing the evidence into EFXML, a number of verification tasks were carried out including reproducing checksums and comparing counts of messages between the original and EFXML representations. Due to the nature of email data, it is possible to observe a size increase in particularly imperfect cases (e.g. where volume of header data exceeds the volume of body data) after the addition of the EFXML tags to the data ³ ; however, these evaluations point toward an average case which does not approach this situation. Potential size decreases could be seen when considering attachments in the body of evidence.

5.7 Summary of Results

The effectiveness of the approach and the PlugsE framework have been evaluated with respect to comprehensiveness of the evidence acquired, the uniformity of differing representations of the same pieces of evidence, the use in collaborative and distributed forensics systems, the consistent performance of the approach, and the performance of the modules developed. These evaluations represent a solution to the problems described in Chapter 1.2.

³In fact, such a case became apparent when using a sample acquired at <http://www.dovecot.org/tmp/dovecot-crlf>

Chapter 6

DISCUSSION

To build upon the previous evaluations, discussing this work in relation to the important topics of the rules of evidence, collaboration, and general use of the approach is important.

6.1 A Note on Legality

It must be emphasized that this approach requires special consideration to laws regarding the search and seizure of evidence. In many territories, it may be necessary to secure a subpoena or warrant before using an approach like ours. However, in the event that the necessary procedures have been followed and the service provider remains uncooperative, this method provides examiners with an alternative means of acquisition for the sake of prompt response, as discussed by Richard Littlehale ¹ in his testimony before the U.S. House Judiciary Subcommittee on Crime, Terrorism, Homeland Security & Investigations on March 19, 2013 [35]. It is critical that practitioners consult proficient legal counsel before utilizing the information contained herein.

6.2 Rules of Evidence

Any forensic framework must uphold *the rules of evidence* (authenticity, admissibility, completeness, and accuracy/reliability), which are the canonical guidelines for handling evidence. The following is a discussion of how this approach supports these important principles.

¹Assistant Special Agent in Charge, Technical Services Unit, Tennessee Bureau of Investigation.

The authenticity of evidence from online sources relies on the same arguments as other live acquisition methods, which by nature are difficult to verify [36]. While our framework is capable of facilitating the protection of evidence authenticity, it remains an issue for developers to implement sound acquisition plugins for PlugsE that capture an accurate representation of the acquired data.

Admissibility implies two key concepts: 1) that the evidence was acquired following proper procedures, and 2) that it was handled properly after acquisition. As discussed in Chapter 6.1, it is the responsibility of practitioners to understand their duties with regard to the first item. However, this framework facilitates the second item by using the EFXML and EFRDF files to record details about the tools that acquired and processed the evidence, which examiners can include in chain of custody forms when necessary.

For evidence to be complete, the format into which it is acquired needs to accurately reflect the original data, as mentioned in Chapter 4.6. Without cooperation from the service provider, it is impossible to know the exact structure of the original data, so focus instead be placed on retaining the available data using standards such as RFC 4021 [32] as a guide. Even though these tools may omit newlines, control characters, and other non-essential data during the processes of acquisition and conversion to mbox, they preserve all the header information, body text, and attachments, which are the portions of interest and which will hold sway in a legal setting. Furthermore, I reiterate that this approach provides examiners access to relevant evidence not available after a simple acquisition of a hard drive, aiding the examiner to form a more clear, complete, and well-informed report on the suspect. I do, however, feel that the algorithm discussed in Chapter 4.6 is costly and should be improved upon in future work. This may be achieved through defining and extracting only specific headers which reveal important information flows (reducing a linear factor

to a constant), using semantic reasoning to determine which messages to focus on, or other yet unexplored tactics. As EFXML and EFRDF are structured data formats, they may facilitate these future approaches by simplifying the development process of tools which utilize them and following current trends in the forensics community [21].

This implementation ensures the reliability and accuracy of evidence it handles by measuring the integrity of each message by taking its checksum during supplemental acquisition and evidence processing. With this, an examiner may verify the evidence has remained unchanged after each step of the process. Also, because there are schemas provided for EFXML and EFRDF, developers can ensure the reliability of their own tools more easily.

6.3 Collaboration

This approach facilitates collaboration in many ways. First, it presents a platform for tool developers that allows them to focus on building their algorithms and modules correctly without having to devote precious time to the acquisition of email data or handling its native format, much the same way that DFXML has done for disk forensics [37]. Second, the use of language agnostic formats such as JSON, EFXML, and EFRDF allows for interoperability between modules in the system, allowing for both collaborative development and the sharing of work among developers with different technical backgrounds as well as allowing for organizations to provide their implementations as SOA products without revealing the intimate details of their methodologies, which may reveal trade secrets or other proprietary information. The ability to share work and tools allows for scenarios where different examiners within one or more organizations can take responsibility for different steps in the forensic process, which could be further enhanced by use in a collaborative forensic system such as CUFF [1]. Finally, the use of PlugsE manifest files gives the system a degree

of autonomy, where an administrator need not have detailed programming expertise, and further increases interoperability by abstracting the details of how modules are accessed; to a practitioner the use of locally-hosted command line tools becomes no different from using a remotely-hosted SOA product accessed over an RPC protocol while logging information contained within their outputs maintains the chain of custody.

6.4 General Use of the Approach

While the dialogue in this thesis focuses mostly on forensic investigations in law enforcement settings, this is not the only environment in which the approach would be useful. As business intelligence and incident response processes mirror those of criminal investigations, this work may be used in industry settings when conducting inquiries into misconduct, misuse of resources, or any other violation of an employee's obligation to their employer. Further, since businesses typically own the machines from which evidence is acquired, the lack of an expectation of privacy on the part of their employees may present more relaxed requirements than those which law enforcement agencies face.

6.5 Limitations of the Approach

This process-driven approach is not without limitations which will now be discussed.

First, while this work does present a repeatable process for conducting email forensics, it does not present a universally applicable best practice for credential discovery, evidence mapping, and secondary acquisition of any given email service; that is to say that the best practice for applying this process to one email web service may differ from another. Along with this limitation, the PlugsE implementation of this framework may not present fully comprehensive results in every case which it

is employed as the results are dependent upon the modules available to it and their implementations of the best practices previously mentioned. As the ecosystem of email service providers grows and changes, tool developers must take special care to continuously implement and refine tools to handle challenges as they are introduced.

Second, in situations where a live acquisition approach has been taken against active email accounts practitioners must take special care to ensure this evidence is properly presented as a snapshot of a system in flux. In preparation for using a live acquisition methodology tool developers and practitioners must enumerate the limitations and effects of such an approach beforehand and show them to be within the restrictions placed upon the investigation by the rules of evidence. If these steps are not followed, the evidence may be ruled inadmissible and negatively affect the outcome of the investigation.

Finally, as previously mentioned in Chapter 6.1, there are many legal implications which must be taken into account when using this approach. First, focus must be placed on upholding the rights of the suspect during the investigation. Further, all modules being used during the investigation must go through thorough verification and validation processes to ensure their proper functioning and representation of the evidence acquired with respect to the rules of evidence. For example, the `mbox2efxml` tool presented for Chapter 4.6 may contain behavior which would misrepresent the evidence acquired during the supplemental acquisition phase of the methodology; as such, no modules presented in this work should be used directly in legal proceeding until they have been subjected to proper scrutiny. Failure to follow either of these guidelines may also invalidate evidence, negatively affecting the outcome of investigation.

Chapter 7

CONCLUSION

To conclude this work, a summary of the work at large, its contributions, and tasks left to future work will now be presented.

7.1 Summary

This work has defined a general methodology for carrying out email forensics and shown a proof of concept implementation with evaluation results. This approach broadened the definition of credentials, identified methods for discovering credentials, and demonstrated the need for a generic evidence representation. The implementation has shown an example of credential discovery for Gmail accounts, a method for reestablishing existing Gmail sessions, the steps needed to carry out a supplemental acquisition, and a completed evidence processing phase generating both an intermediate mbox representation of email evidence and proposed EFXML/EFRDF representations of email headers upon which further analysis can be carried out while addressing the need to facilitate collaboration amongst developers and organizations during the creation of tools for forensic investigations as well as the investigations themselves.

7.2 Contributions

This work represents contributions to the field of email forensics in three distinct areas: 1) the benefits of a new process-driven approach to email forensics 2) the benefits of using a well-structured and uniform evidence representation 3) and the benefits of PlugsE and its sample modules, a realization of the approach.

7.2.1 *Process-Driven Approach*

The proposed and evaluated process-driven approach provides a new method for conducting acquisitions of email for forensics where, instead of treating it simply as a task to be carried out during the evidence investigation, it is treated in much the same way as disk forensics where the evidence itself is the main focal point of the process as described in Chapter 3. While these results could previously be achieved through manual examination of disk evidence, there lacked a formal approach to the task. Further, the success of an examination using only manual examination hinged on the examiners technical skills (i.e. discover, map, and reuse credentials then acquire the data) as well as past experiences (e.g. previously having reused cookies to retrieve email evidence and concluding it is a worthwhile approach), where this process-driven approach leaves the task of creating a consistent approach to email forensics to the developers of forensic tools to allow examiners to focus their efforts on the actual analysis of the evidence. As this process can be automated, it increases the reliability of acquired evidence and repeatability of the process used to acquire it, which adds weight to the results of an examination according to the rules of evidence. Also, the comprehensiveness of the acquired evidence is also greatly increased by ensuring that evidence is acquired from all available data sources, rather than only the storage media initially acquired. Finally, by defining the distinct steps in the process (credential discovery, evidence mapping, supplementary acquisition, and evidence processing) best practices can be defined for each task required to conduct email forensics and provides a common vocabulary for researchers to use when discussing what these best practices entail.

7.2.2 Evidence Representation

Following the trends in the forensics community to use well-structured, uniform XML based evidence containers [21] such as DFXML [37], two new evidence containers tailored for use in email forensics, EFXML and EFRDF, were designed and developed as discussed in Chapter 4. In addition, schemas to accompany these evidence representations were developed to aid in the validation and verification of evidence. These evidence containers abstract away the technical details of how and where evidence was stored and allows for examiners to focus on the emails themselves as evidence and the tools which use them. Since EFXML is purely XML-based, it lends itself to easy syntactic reasoning approaches to email analysis, such as querying for specific header fields using the XPath query language. EFRDF, on the other hand, provides a format which can be utilized in semantic reasoning engines to draw conclusions about the evidence based upon the characteristics of these header fields, which may be used to reveal information flows which are not directly evident across multiple bodies of evidence. As these formats store all relevant forensic information including dates, checksums, and tool versions it lends itself to maintaining the chain of custody and aids examiners in verifying that they have followed the rules of evidence. Finally, the well-structured nature of these containers allows for forensic tool developers to verify the proper functioning of their tools and examiners to validate their output.

7.2.3 Realization of Approach

PlugsE as presented in Chapter 4 was developed as a realization of the process-driven methodology defined previously. PlugsE is a pluggable framework which allows for the system to be extended and grow as best practices emerge for various email service providers, fully realizing the comprehensiveness of the methodology. As it

provides interfaces for either local tools or remote tools accessible via RPC, PlugsE also greatly aids in collaborative and distributed forensic examinations such as those facilitated in systems like CUFF [1]. A proof of concept implementation of each step in the forensic process has also been provided, showing a useful approach to reusing credentials in the form of browser cookies to gain a more complete body of evidence through the automated processing of disk-based evidence. The following tools have been designed and implemented to support this framework: a credential discovery tool **Henson** which utilizes the Search Known Locations method to discover credentials stored in cookie databases, an evidence mapping tool **Crumbler** to detect the presence of active email sessions, a supplementary acquisition tool **Spatula** which uses these sessions to screen scrape the evidence, and two evidence processing modules **mbox2efxml** and **efxml2efrdf** which facilitate the proposed evidence representation schemas EFXML and EFRDF. The developed tools and framework facilitate novel approaches to the analysis of email evidence so examiners can further reveal information flows which may otherwise go unnoticed.

7.3 Future Work

Throughout the course of this work, numerous research problems presented themselves. First, since credentials are defined as any pieces of information which may reveal the existence of further email evidence, new approaches to discovering and reusing the various types of credentials which may exist on a system, such as those in the form of text files or encrypted keyring systems, will be investigated. By continuing research in this direction, the comprehensiveness of this process will greatly increase the value of following the methodology.

Next, further best practices for each step in the process will be investigated and established to increase the forensic soundness of the tool. This research should focus

greatly on following the rules of evidence while aiding examiners in maintaining the chain of custody and give legal staff solid ground from which to present the results of an investigation.

This process may also be useful in situations other than those of email forensics, business intelligence, and incident response — its applications in other realms will be explored. Particularly, this approach may be generalized to handle all kinds of remote services in the rapidly growing ecosystem of services and service providers. Remotely maintained storage and software applications are rapidly gaining traction in the consumer market and should also be addressed by the forensic community to create more and more comprehensive narratives of events during investigations.

Finally, tools and novel approaches to the analysis of the evidence acquired during the process which leverage the usefulness of the evidence container formats presented alongside it will be investigated and developed. In particular, the usefulness of semantic reasoning on email evidence will be explored past the simple examples previous presented. More thorough feature extraction based on the body of email evidence may also be aided during an analysis utilizing semantic reasoning techniques and the costs and benefits of including the body of the email in the container formats must be explored. As part of this work, improvements upon Algorithm 2 should be explored to reduce its cost. While analysis tasks are not addressed by this work, they are an important step in the forensic process. Utilizing the evidence containers presented herein may support the previously mentioned novel approaches to analytical tasks.

REFERENCES

- [1] M. Mabey and G.-J. Ahn, "Towards collaborative forensics: Preliminary framework," in *Information Reuse and Integration (IRI), 2011 IEEE International Conference on*, 2011.
- [2] Fox News, "Petraeus resigns after affair with biographer turned up in fbi probe, fox news confirms," <http://www.foxnews.com/>, November 2012.
- [3] B. Nelson, A. Phillips, F. Enfinger, and C. Steuart, *Guide to computer forensics and investigations*. Boston, Mass: Thomson Course Technology, 2008.
- [4] K. Jones, R. Bejtlich, and C. Rose, *Real Digital Forensics: Computer Security and Incident Response*. Addison Wesley Professional, 2006. [Online]. Available: <http://books.google.com/books?id=CXmWSAAACAAJ>
- [5] W. Kruse and J. Heiser, *Computer Forensics: Incident Response Essentials*. Pearson Education, 2001. [Online]. Available: <http://books.google.com/books?id=-qWa5Svv7BIC>
- [6] E. Casey, *Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet*. Elsevier Science, 2011. [Online]. Available: http://books.google.com/books?id=lUnMz_WDJ8AC
- [7] L. Volonino, R. Anzaldua, and J. Godwin, *Computer forensics: principles and practices*, ser. Security series. Pearson/Prentice Hall, 2007. [Online]. Available: <http://books.google.com/books?id=EWVGAAAAYAAJ>
- [8] J. Paglierani, M. Mabey, and G.-J. Ahn, "Towards comprehensive and collaborative forensics on email evidence," in *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 9th IEEE International Conference on*, 2013.
- [9] S. Greengard, "On the digital trail," *Communications of the ACM*, vol. 55, no. 11, pp. 19–21, November 2012. [Online]. Available: <http://doi.acm.org/10.1145/2366316.2366323>
- [10] G. Grispos, W. B. Glisson, and T. Storer, "Using smartphones as a proxy for forensic evidence contained in cloud storage services," in *46th Hawaii International Conference on System Sciences*, 2013, pp. 1–10.
- [11] D. Quick and K.-K. R. Choo, "Dropbox analysis: Data remnants on user machines," *Digital Investigation*, vol. 10, no. 1, pp. 3 – 18, 2013. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S174228761300011X>
- [12] "Aid4mail: Software for email forensic analysis and e-discovery." [Online]. Available: <http://www.aid4mail.com/email.ediscovery.forensic.software.php>

- [13] Radicati Group, Inc., “Email Market, 2012-2016,” <http://www.radicati.com/wp/wp-content/uploads/2012/10/Email-Market-2012-2016-Executive-Summary.pdf>, October 2012.
- [14] R. Hadjidj and et al., “Towards an integrated e-mail forensic analysis framework,” *Digital Investigation*, vol. 5, no. 3–4, pp. 124–137, 2009. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1742287609000036>
- [15] O. De Vel, A. Anderson, M. Corney, and G. Mohay, “Mining e-mail content for author identification forensics,” *ACM Sigmod Record*, vol. 30, no. 4, pp. 55–64, 2001.
- [16] A. Popescu and H. Farid, “Statistical tools for digital forensics,” in *Information Hiding*, ser. Lecture Notes in Computer Science, J. Fridrich, Ed. Springer Berlin Heidelberg, 2005, vol. 3200, pp. 128–147. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30114-1_10
- [17] “Forensic explorer - fact sheet.” [Online]. Available: <http://www.forensicexplorer.com/analysis.php>
- [18] “Properties of email examiner that help in recovering forensic email evidence.” [Online]. Available: <http://www.mailxaminer.com/email-examiner-feature.html>
- [19] “Intella - technical overview.” [Online]. Available: [https://www.vound-software.com/resources/files/Intella%20Fact%20Sheet%20\(FS008\)US-Online.pdf](https://www.vound-software.com/resources/files/Intella%20Fact%20Sheet%20(FS008)US-Online.pdf)
- [20] S. L. Garfinkel, “Digital forensics research: The next 10 years,” *Digital Investigation*, vol. 7, pp. S64–S73, 2010.
- [21] W. Alink, R. Bhoedjang, P. Boncz, and A. de Vries, “XIRAF–XML-based indexing and querying for digital forensics,” *Digital Investigation*, vol. 3, pp. S50–S58, 2006.
- [22] S. Garfinkel, “Digital forensics XML and the DFXML toolset,” *Digital Investigation*, vol. 8, no. 3–4, pp. 161–174, 2012. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1742287611000910>
- [23] F. Buchholz and E. Spafford, “On the role of file system metadata in digital forensics,” *Digital Investigation*, vol. 1, no. 4, pp. 298–309, 2004.
- [24] M. T. Banday, “Analyzing e-mail headers for forensic investigation,” *Journal of Digital Forensics, Security, and Law*, vol. 6, pp. 49–64, 2011. [Online]. Available: <http://www.jdfsl.org/subscriptions/abstracts/JDFSL-V6N2-column-Banday.pdf>
- [25] J. Zawinski, “Bug 241438 - please make history.dat easier to parse (i.e., not mork),” April 2004. [Online]. Available: https://bugzilla.mozilla.org/show_bug.cgi?id=241438#c0

- [26] V. Roussev and G. G. Richard III, "Breaking the performance wall: The case for distributed digital forensics," in *Proceedings of the 2004 Digital Forensics Research Workshop*, vol. 94, 2004.
- [27] G. G. Richard III and V. Roussev, "Next-generation digital forensics," *Communications of the ACM*, vol. 49, no. 2, pp. 76–80, 2006.
- [28] S. Garfinkel, P. Farrell, V. Roussev, and G. Dinolt, "Bringing science to digital forensics with standardized forensic corpora," *digital investigation*, vol. 6, pp. S2–S11, 2009.
- [29] S. Garfinkel and D. Cox, "Finding and archiving the internet footprint," in *Digital Lives Research Conference: Personal Digital Archives for the 21st Century*, February 2009.
- [30] S. Garfinkel, P. Farrell, V. Roussev, and G. Dinolt, "Bringing science to digital forensics with standardized forensic corpora," *Digital Investigation*, vol. 6, no. Supplement 1, pp. S2–S11, 2009, the Proceedings of the Ninth Annual DFRWS Conference. [Online]. Available: <http://www.sciencedirect.com/science/article/B7CW4-4X1HY5C-3/2/090ebc16025d598c775d87c8abbb7ae5>
- [31] E. Hall, "The application/mbox Media Type," RFC 4155 (Informational), Internet Engineering Task Force, Sep. 2005. [Online]. Available: <http://www.ietf.org/rfc/rfc4155.txt>
- [32] G. Klyne and J. Palme, "Registration of Mail and MIME Header Fields," RFC 4021 (Proposed Standard), Internet Engineering Task Force, March 2005, updated by RFC 5322. [Online]. Available: <http://www.ietf.org/rfc/rfc4021.txt>
- [33] W3C, "RDF primer," W3C Recommendation 10 February 2004, W3C, February 2004. [Online]. Available: <http://www.w3.org/TR/2004/REC-rdf-primer-20040210/>
- [34] P. Guo, "Email analysis scripts for mbox mailbox files," <http://www.pgbovine.net/mbox-analysis.htm>, Sep. 2006.
- [35] R. Littlehale. (2013, March) Hearing on ECPA part 1: Lawful access to stored content, written testimony of Richard Littlehale. http://judiciary.house.gov/hearings/113th/03192013_2/Littlehale%2003192013.pdf. Tennessee Bureau of Investigation.
- [36] B. Schatz, "BodySnatcher: Towards reliable volatile memory acquisition by software," *Digital Investigation*, vol. 4, Supplement, no. 0, pp. 126–134, 2007. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1742287607000497>
- [37] S. Garfinkel, "Automating disk forensic processing with sleuthkit, xml and python," in *Systematic Approaches to Digital Forensic Engineering, 2009. SADFE '09. Fourth International IEEE Workshop on*, 2009, pp. 73–84.