THE UNIVERSITY OF NEW HAVEN

CYBER FORENSICS NEEDS ANALYSIS AND SOLUTIONS ENTAILING ARTIFACT FORMALIZATION AND APPROXIMATE MATCHING

A THESIS
submitted in partial fulfillment
of the requirements for the degree of
MASTER OF SCIENCE IN COMPUTER SCIENCE

BY

Vikram S. Harichandran

Department of Electrical & Computer Engineering and Computer Science
Tagliatela College of Engineering

February 2018

CYBER FORENSICS NEEDS ANALYSIS AND SOLUTIONS ENTAILING ARTIFACT FORMALIZATION AND APPROXIMATE MATCHING

APPROVED BY

_____

[Frank Breitinger, Ph.D.]
Thesis Adviser

_____

[Ibrahim Baggili, Ph.D.]
Committee Member

_____

[Amir Esmailpour, Ph.D.]
Committee Member

_____

[Barun Chandra, Ph.D.]
Program Coordinator

_____

[Ronald S. Harichandran, Ph.D.]
Dean of the College

_____

[Daniel May, Ph.D.]
Provost

To my unconditionally supportive and artistically scientific family

Acknowledgements

I would like to thank my all-encompassing professors, mentors, and friends Dr. Frank Breitinger (thesis advisor) and Dr. Ibrahim Baggili for giving me the opportunities to publish and present these papers, enhance my professional focus, and personally grow into a more adept explorer of the sciences.

I would also like to extend a thank you to the members of the University of New Haven Cyber Forensics Research and Education Group for keeping me sane enough to stay on the good side.

Abstract

The cyber realm has been under fresh scrutiny in recent history due to the increase in successful large-scale breaches. This has added to the hindrances forensic professionals, especially, have to deal with to entrap and convict malicious actors. This document presents a series of four works (three published) which focus on the crux of how to better assist digital forensic investigators in regards to 1) education / training / certification, 2) resource allocations (including personnel, funding, and research focus), 3) organization, 4) tools / technology, 5) communication, 6) standards / semantics, 7) evidence curation / case tracking, 8) automated white / black-listing, and 9) template-content splitting. First, a general needs analysis survey was conducted that determined some of the primary roadblocks for various types of professionals linked with the sleuth process (addressed in 1-5 enumerated above). Major takeaways included a need for more cloud / mobile support, better communication, and standardization, among others – abstracts for each chapter are incorporated for a cursory reading of complete outcomes. By concentrating on these findings the thesis was distilled, cuing development of a database-reliant model for evidence curation; this second chapter covers a brief survey and usage analysis before presenting CuFA, Curated Forensic Artifact, as a more standardized definition for evidential items than those found. The University of New Haven Cyber Forensics Research and Education Group (UNHcFREG) created the Artifact Genome Project (AGP) in conjunction with this, helping to illustrate how objects similar to those of the Cyber Observable eXpression (CybOX) project could be applied. To assist this model with approximate matching a review of the subdomain was performed (third chapter) to attempt to address the lack of awareness and application found in the needs analysis survey. Finally, the use of approximate matching databases was looked into for faster document similarity recall. This fourth chapter illuminates how similarity algorithms could be used within a CuFA database for better automation, an unintended consequence being possible template/header carving. A `MongoDB` implementation, `SimFind`, was used to make defining similarity more customized. A CuFA database will arguably save time for forensic investigations.

Contents

List of Tables

List of Figures

Abbreviations

AGP	**A**rtifact **G**enome **P**roject
BBH	**B**lock-**B**ased **H**ashing
BBR	**B**lock-**B**ased **R**ebuilding
CPI	**C**orrelation **P**reserving **I**ndex
CTPH	**C**ontext **T**riggered **P**iecewise **H**ashing
CuFA	**Cu**rated **F**orensic **A**rtifact
CybOX	**Cyb**er **O**bservable e**X**pression
DESO	**D**igital **E**vidence **S**emantic **O**ntology
DFAX	**D**igital **F**orensics **A**nalysis e**X**pression
DFXML	**D**igital **F**orensics **XML**
ETC	**E**ducation/**T**raining/**C**ertification
F2S2	**F**ast **F**orensic **S**imilarity **S**earch
F3	**F**irst **F**orensic **F**orum
FBHMA	**F**ile **B**lock **H**ash **M**ap **A**nalysis
FRASH	**FR**amework to test **A**lgorithms of **S**imilarity **H**ashing
FTK	**F**orensic **T**ool**k**it
GRR	**G**oogle **R**apid **R**esponse
GUID	**G**lobally **U**nique **ID**entifier
IOC	**I**ndicator **O**f **C**ompromise
IoT	**I**nternet **o**f **T**hings
IRB	**I**nstitutional **R**eview **B**oard
ISI	**I**nternational **S**cientific **I**ndexing
LSH	**L**ocality-**S**ensitive **H**ashing
MHR	**M**alware **H**ash **R**egistry
NSRL	**N**ational **S**oftware **R**eference **L**ibrary
OCR	**O**ptical **C**haracter **R**ecognition
RLE	**R**un **L**ength **E**ncoding
SIF	**S**tatistically-**I**mprobable **F**eatures
STIX	**S**tructured **T**hreat **I**nformation e**X**pression
TAXII	**T**rusted **A**utomated e**X**change of **I**ndicator **I**nformation
TLSH	**T**rend Micro **L**ocality **S**ensitive **H**ash
UCO	**U**nified **C**yber **O**ntology
XIRAF	**X**ML **I**nformation **R**etrieval **A**pproach to digital **F**orensics

Chapter 1

Introduction

The never-old narrative of benevolence versus the nefarious has observed a spike in both media attention and civilian awareness in relation to cybersecurity and digital forensics in the last few years. Due to the sudden ubiquity of long-lasting batteries and seemingly metaphysical properties of mobile devices the world's behaviour has evolved in ways both foreseen and hidden. Daily interactions through the Internet have begun impacting minute habitual actions, to the extent people experience phantom notification sounds from their pocket and subconsciously make decisions based on data conveyed through their phone. Little would argue the world hasn't gone through a dramatic metamorphosis, and consequently much research is going into the psychological, sociological, economic, environmental, computational, and ethical effects of electronics. But resulting international and cultural appreciations aside, the detriments of a more connected globe have been enormous, particularly for cybercrime.

Privacy has become a central political debate because criminals have a much wider playground to exercise on. Those molded by the digital domain have long been aware of cyber threats but frequent usage by those less educated on the subject matter has resulted in a slew of new attacks, hacks, and complexly-linked physical danger (terrorism being only the most notable). The question for scientists, law enforcers, and judicial figures has become "What methods, policies, and mindsets must we establish to maintain stability and safety during this phase of scientific revolution?" Particularly, if a crime occurs it must first be fully comprehended what happened, how, and by whom – this is the role of digital forensics within cybersecurity (the role also includes representation of findings legally).

Scientific paradigm shifts like this are often multi-faceted and require adaptation (Kuhn, 1970), so reevaluation during this transition is important to keep pace with global delinquents. This document is a collection of four works of research that attempt to reevaluate and improve on the status quo in digital forensics by giving investigators a better way to extract, handle, and analyze evidence; titles and publication information below:

1. A cyber forensics needs analysis survey: Revisiting the domain's needs a decade later[1]

2. CuFA: a more formal definition for digital forensic artifacts[2]

[1] http://doi.org/10.1016/j.cose.2015.10.007 (last accessed 2017-05-02).

[2] http://doi.org/10.1016/j.diin.2016.04.005 (last accessed 2017-05-02).

3. Bytewise approximate matching: the good, the bad, and the unknown[3]

4. Usage of approximate matching databases for document similarity lookup and template detection[4]

The first chapter, a survey, served as a way to broadly understand the currently perceived problems in the field and update what forensic practitioners value (contrasted with research, legal, and law enforcement perspectives). Second, "CuFA: a more formal definition for digital forensic artifacts" proposed advancing how evidential items referred to as "artifacts" are defined and handled, based on some of the findings of the survey. Terminology was introduced (Curated Forensic Artifact) and a model proposed, as well as a database design/system (Artifact Genome Project, or AGP[5]) for implementation. Then, addressing another finding from the survey, approximate matching was reviewed (Chapter 4) to increase awareness of the subdomain and support its use in a model like the one proposed in the previous chapter to better find items. Finally, funneling the output from the prior chapters to an even smaller focus, Chapter 4 looked at primary obstacles with approximate matching: lookup speed, priority of similarity, and similarity scores. Hoping to encourage usage of databases to mitigate these obstacles was secondary to our higher-level vision of using one in parallel with a system like the AGP to automatically help investigators find evidence they would not have come across themselves

[3] `http://ojs.jdfsl.org/index.php/jdfsl/article/view/427` (last accessed 2017-05-02).

[4] Unpublished and will be revised before future submission.

[5] `https://agp.newhaven.edu`

natively (e.g., finding related items between cases). An originally unplanned and subsequent benefit of exploring this research was the discovery of a possible template/header carving capability. Note, each chapter may have slightly different linguistic, grammatical, and visual presentation due to their publication in different mediums.

Chapter 2

A cyber forensics needs analysis survey: Revisiting the domain's needs a decade later

2.1 Abstract

The number of successful cyber attacks continues to increase, threatening financial and personal security worldwide. Cyber/digital forensics is undergoing a paradigm shift in which evidence is frequently massive in size, demands live acquisition, and may be insufficient to convict a criminal residing in another legal jurisdiction. This paper presents the findings of the first broad needs analysis survey in cyber forensics in nearly a decade,

aimed at obtaining an updated consensus of professional attitudes in order to optimize resource allocation and to prioritize problems and possible solutions more efficiently. Results from the 99 respondents gave compelling testimony that the following will be necessary in the future: (1) better education/training/certification (opportunities, standardization, and skill-sets); (2) support for cloud and mobile forensics; (3) backing for and improvement of open-source tools (4) research on encryption, malware, and trail obfuscation; (5) revised laws (specific, up-to-date, and which protect user privacy); (6) better communication, especially between/with law enforcement (including establishing new frameworks to mitigate problematic communication); (7) more personnel and funding.

2.2 Introduction

With the rising presence of digital devices, information repositories, and network traffic, cyber forensics (a.k.a. digital forensics) faces an increasing number of cases having ever-growing complexity Al Fahdi, Clarke, and Furnell (2013). The large volume of data and the deficit of time needed for examination have placed pressure on the development of real-time solutions such as criminal profiling systems, triage automation, and tools capable of recovery-processing parallelization. These main challenges have rippled across the field introducing and building upon obstacles related to cyber forensic resources, education/training/certification (ETC), tools & technology, research, laws, and subdomains.

Needs analysis is one type of assessment tool used for identifying areas that members of a community view as challenging. Performing them periodically is essential to adjust for

fluctuating trends. Some of the benefits of needs assessment include improved resource allocation, better efficiency, informed decision-making, the generation of a professional consensus, and increased awareness of unaddressed problems and possible solutions.

The last general needs analysis survey on cyber forensics was conducted by (Rogers & Seigfried, 2004), titled "The future of computer forensics: a needs analysis survey." Since the publication of the original study, only subdomains have been assessed. Our study was performed to obtain an updated consensus of the cyber forensics community's opinions in order to more extensively identify and prioritize problems and solutions.

We present the feedback of 99 respondents to our 51-question survey which, among others, strongly motivates the need for more funding and personnel; better ETC, tools, and communication; updated laws; and research on cloud and mobile forensics. We further performed a direct comparison to the 2004 survey results.

This paper is divided into 6 major sections. First, we present a summary of challenges & recent findings in Section 2.3. In Section 2.4 we briefly outline the methodology, followed by a short survey section. The core of this paper is Section 2.6, which includes the results. Next, Section 2.7 discusses the implications of the results. Then the limitations are stated. Main findings and future follow-up can be found in Section 2.9.

2.3 Summary of challenges & recent findings

Despite a general public concern for cyber security, the technological response, frameworks, and support are lagging behind the escalating rate of crime. The PricewaterhouseCoopers (PwC) 2014 Global Economic Crime Survey reported 7% of U.S. organizations lost $1 million or more due to cybercrime in 2013, and 19% of U.S. entities lost $50,000 to $1 million. Global equivalents of these financial ranges of loss were 3% and 8%, respectively Mickelberg, Schive, and Pollard (2014).

Accumulation of financial loss is not the only worry as recently shown with the Sony hack in which a group of hackers calling themselves the Guardians of Peace gained access to Sony systems and threatened violence on company employees if the movie 'The Interview' was released. Threat of terrorism caused the banning of the movie in nearly 20,000 theaters in North America Dickson (2015). As mirrored in the PwCs Annual Global CEO Survey of 2014, CEOs and executive boards must now be concerned with such risks lest denial-of-service attacks and damaged image effect company survival Mickelberg et al. (2014). In the foreseeable future, the Internet of Things (IoT) will likely be just as much of a concern. Automobiles and other electronically enhanced devices will form new risks to individuals, businesses, and governments.

Due to such cyber incidents, the area of cyber forensics has gained ample publicity over the recent years as it becomes more important to analyze and understand breaches. In the following subsections we summarize the state of the art cyber forensics practices and

research.[1] First, however, the previous general needs analysis survey is described.

2.3.1 Rogers & Seigfried survey

The first and only needs analysis survey conducted by (Rogers & Seigfried, 2004) consisted of a single question asking participants to list what they viewed as the top five issues in computer forensics (computer forensics was more synonymous with cyber forensics then). Responses were tallied into high-order categorizations, exhibiting the following order of frequency (from most mentioned to least):

1. Education/training/certification
2. Technologies
3. Encryption
4. Data acquisition
5. Tools
6. Legal justice system
7. Evidence correlation
8. Theory/research
9. Funding

[1]For more background reading on the current standards see (Darnell, 2012).

10. Other

These categories were used in our study to directly assess how the understanding of challenges changed over the last decade. Since the time of the preliminary paper, many new challenges have emerged.

2.3.2 Resource allocation & education / training / certification

Although a variety of crimes can be committed using digital devices (violent crimes, terrorism, espionage, counterfeiting, drug trafficking, and illicit pornography to name a few) the largest driving force for cyber forensics is crime related to financial security. A survey conducted by the Ponemon Institute indicated the average cost of cybercrime for United States retail stores in 2014 ($8.6 million per company) was more than double that of 2013 Walters (2014). Estimations only account for publicly disclosed figures. We posit that reported numbers from surveys are likely to be underestimations because of their voluntary nature. Prevention is clearly not working, which reflects the accumulation of cases.

Consequently, the question of whether there are enough forensic practitioners arises, which can only be answered with speculation as no recent studies have attempted to quantify this; personnel, as a resource, was evaluated in our survey. General conjecture is that more cyber forensic scientists and professionals are needed to appease the amount of cases. This is partially backed by a 37% projected increase in employment of information security analysts (this category encompasses cyber forensics practice) from 2012 to 2022

Bureau of Labor Statistics, U.S. Department of Labor (2014). Funding may need to increase or be reallocated as these jobs are put on the market. Additionally, in part due to the variety of both cases that practitioners deal with and the methods they use to analyze the evidence, there is still no standardized certification for examiners Srinivasan (2013). Instead, professionals usually obtain tool-specific certifications. The same can be seen in law enforcement and judicial courts. A recent study supported by the National Institute of Justice (NIJ) indicated that first-responding officers often do not know how to properly secure digital evidence, and that prosecutors have a tendency to request all information from devices without considering their physical storage size Goodison, Davis, and Jackson (2015). Such a diverse and likely insufficient large pool of personnel urges the creation of faster and more efficient tools and technology to improve case processing.

2.3.3 Tools & technology

In the past, tools tended to be technology-oriented, inconveniencing non-technical users, and lacked user-friendly, intuitive interfaces Reith, Carr, and Gunsch (2002). Today, investigating simple questions such as whether two people were in contact and which websites a person has visited still requires too much time and effort. Usually, following complex leads result in the case being handed on to more experienced, specialized investigators. Tool usability and reporting is an important issue because "misunderstanding that leads to false interpretations may impact real-life cases" Hibshi, Vidas, and Cranor (2011). Furthermore, recent work has illustrated that tools still lack standardized reporting mechanisms, and even though research has been conducted on this front, the

tools have not adopted a standard for digital evidence items Bariki, Hashmi, and Baggili (2011a). The young, incompletely explored open-source landscape needs further inquiry as well because there is powerful functionality to be gained from tools tested, validated, and constantly updated via communal repositories or trusted open-sourced centers Greek (2013).

Two other approaches being worked on to improve tool efficiency is implementation of automation and real-time processing technology. Triage automation is considered by some to be essential for dealing with the increasing number of cases S. L. Garfinkel (2013). The plethora of photos created every day illustrates the need for automation. Photo doctoring is becoming commonplace yet automation of image forgery detection is still not possible Birajdar and Mankar (2013). Automation could be critical in the future as instances of slander and false evidence increase. Indeed, earlier this year a new Stegosploit tool demonstrated malicious, self-executing code could be hidden within pictures Harblson (2015). On the other hand, mobile and flash memory devices (including video game consoles and eBook readers) have resulted in quick evidence deletion. To combat this, memory forensics, real-time detection, and parallel processing research has surfaced. Parallel processing would be most beneficial in these areas: traffic generation (network models), imaging and carving processes, and development of user history timelines Nance, Hay, and Bishop (2009). Finally, it is unclear how to separate user/owner privacy and identification from thorough investigations, and this seems to be a topic of increasing interest Aminnezhad, Dehghantanha, and Abdullah (2012).

2.3.4 Research

Research is an essential component in this period of changing focus but may not be achieving ideal output. Scientific journals within the field are relatively new, exhibiting "low ISI impact factors, circulation rates, and acceptance rates" - journals will need time to mature Beebe (2009). Experiments are rarely reproducible because of the lack of corpora, or standardized data sets, made available along with publications, which could also explain why mainstream journals lack interest in the domain's research S. Garfinkel, Farrell, Roussev, and Dinolt (2009). There is a disconnect between practitioners and researchers. Despite their different roles in the field, research should support the desires of those practicing evidence recovery and examination. A survey by (Al Fahdi et al., 2013) marked that practitioners were concerned with anti-forensics and encryption as future challenges while researchers worried about tool capability and social networking. Difference in opinion may be caused by the particular problems these two groups handle, but this disparity should not be present (ideally) when determining prioritization of research topics or funding. Whether this is currently the case is unclear. This disconnect can also be thought of as a failure for research to affect end users as discussed by (S. L. Garfinkel, 2010).

(Baggili, BaAbdallah, Al-Safi, & Marrington, 2013) studied cyber forensics research trends by analyzing 500 papers from the domain and categorizing them. The overall results indicated that the rate of publication in cyber forensics continues to increase over time. Additionally, results showed an overall lack of anti-forensics research where only

2% of the sampled papers dealt with anti-forensics. The results also showed that 17% of the samples were secondary research, 36% were exploratory studies, 33% were constructive and 31% were empirical. One important finding was the discovery of a lack of basic research, where most of the research (81%) was applied, and only 19% of the articles were categorized as basic research. Also, the results exemplified a shortcoming in the amount of quantitative research in the discipline, with only 20% of the research papers classified as quantitative, and the other 80% classified as qualitative. Furthermore, results showed that the largest portion of the research (almost 43%) from the examined sample originated from the United States. In summary, some of the identified challenges, and their associated needs, in cyber forensics research are not fully understood.

2.3.5 Law

Whatever the progress made in researching better solutions and improving tools, practitioners are limited by what they can and cannot do by law, and the evidence they find may not convict a criminal Hack In The Box Security Conference (2012); Dardick (2010). Ransomware is a prime example of the effective means criminals now possess to anonymously and rapidly cash out European Cybercrime Center (2014). In the case of cloud forensics, research needs to be conducted to show the true impact of clouds on cyber forensics before frameworks and guidelines can be established Grispos, Storer, and Glisson (2013). However, in most cases there is ample evidence showing laws are outdated. The subdomain of cloud forensics has proven the need for new laws related to proactive collection of data and multi-jurisdiction laws Ruan, Carthy, Kechadi, and Baggili (2013).

A comprehensive international decree, possibly headed by the United Nations (U.N.), is imperative. According to (Barwick, 2014), currently "data sovereignty laws hamper international crime investigations" and although the U.N. adopted a surveillance proposal in 2013 more forensic-oriented laws are still deemed necessary.

Of course, in addition to these laws judges themselves must be educated and trained, since they are responsible for determining what types of digital evidence are allowed in their courts and how they are used for incrimination. These decisions are mostly guided by three pieces of legislation: *Daubert v. Merrel Dow Pharmaceuticals, Inc.* (1993), *Kumho Tire v. Carmichael* (1999), & FRE Rule 702. A small, yet thorough study Kessler (2010) involving a survey and interviews established that judicial education systems are lacking for digital evidence; judges themselves rated their knowledge of computer forensics as less than computer and Internet technology.

2.3.6 Communication

In response to a survey of Australia's finance and insurance industry there was a high no-incident and no-response rate (around 83%) when companies were asked about their most significant computer security incident Choo (2011). This suggests victims may not be aware they have been successfully attacked or that private companies are reluctant to report victimization. If the latter is true, a framework for anonymous reporting may be useful. Also, as mentioned in Section 2.3.4, differences in opinion between practitioners, researchers, law enforcement, and non-forensic entities may occur natively due to their

different roles but little to no research has been conducted into whether this is due to faulty communication.

2.3.7 Urgency of closing the gaps

The end of the "Golden Age" of cyber forensics, as described by (S. L. Garfinkel, 2010), has quickly outdated many methods used by examiners, causing a paradigm shift with unclear direction. Some of the major concerns include standardization, researching new methods to speed up evidence recovery and analysis (proactive cyber forensics), support for non-traditional devices, and bringing about cheaper tools that support a wider variety of purposes (whether they are all-in-one or bolster a specific function). Although the field is on its way to making some changes, many obstacles such as the ones described in the above subsections prevail. Disregarding the rapidly increasing (successful) crime rate, the need to determine the direction of research, practice, and laws is vital. Cyber forensics' growth will continue to be stunted until these challenges are concretely addressed.

2.4 Methodology

To complete this work, the following high-level methodology was employed:

1. Performed a literature review (main findings are mentioned in Section 2.3) and survey design.

2. Obtained a category two exemption from the Institutional Review Board (IRB) at the University of New Haven (this meant that the survey did not record participant identification information or behavior, and posed no risk or harm to subjects not encountered in every day life).

3. Distributed the survey via list servers, LinkedIn cyber forensics groups, Twitter, and e-mail contacts.

4. Obtained data by exporting the coded responses to XLSX and CSV files from the Baseline survey system.

5. Analyzed the data using statistical probability, power tests, and crossing non-demographic questions with demographics and each other.

2.5 Survey design

The questions were formulated based on typical needs assessment topics, the (Rogers & Seigfried, 2004) survey, critical areas from the literature review that were unknown or deserved further investigation, and our interests. The survey went through three drafts, followed by a brief testing phase, in which three experts within the field were consulted to refine the wording, content, and formatting of the survey.

Needs assessment is a systematic process for determining gaps between the *status quo* and the desires of those within a community. Consequently, survey questions were designed

to identify unmet desires rather than explicitly obtain statistics on the current state of the field. The survey consisted of 51 questions:

- 28 Likert scale
- 13 multiple choice
- 7 multiple selection (checkbox)
- 2 free response
- 1 ranking

According to IRB practices at our institution, participants could not be forced to answer any single question.

A general cyber forensics audience was targeted for the survey because of the researcher-practitioner discrepancies mentioned before (Section 2.3.4), to obtain as unbiased and wholesome a perspective as possible (for instance, if asked a group may likely blame another group rather than themselves for poor communication, or state their area is underfunded), and to understand how motivated people are to join such areas (e.g. hypothetically, if academia/research was found to be underfunded less people might find the domain desirable, which would be a problem if more employees were needed in this domain).

2.6 Results

The survey was available online for one month before data were exported from the system. Ninety-nine participants submitted responses. The calculated required sample size was 76 indicating that the number was large enough to make inferences from and that statistical tests were unlikely to exhibit type II errors (two-sided t-test, alpha = 0.05, using a medium effect size of 0.5 and power of 0.99). It should be noted that we aspired to obtain a higher response rate, but taking into account the relative size of the cyber forensics domain compared to cyber security in general, we deemed the sample size acceptable.

We would like to point out that although websites (e.g., the Digital Forensics Training on LinkedIn or the First Forensic Forum[2] (F3)) have hundreds or thousands of members, we do not believe citing the number provides a good estimation of the size of the domain – people in LinkedIn groups often are there to observe or self-promote, and may not be active members of the community. A reasonable approach to analyze the number of practitioners would be to count all degree holders of organizations that provide certifications. However, organizations as such do not publicly release statistics on how many professionals are trained or end up as practitioners in the domain.

This section's structure reflects that of Section 2.3, preceded first by an overview of the demographics and a comparison with the 2004 survey. Figures and tables related to this section using percentages may not sum exactly to 100% due to rounding error.

[2] https://www.f3.org.uk (last accessed 2017-05-02).

2.6.1 Demographics

The results of the demographics questions are presented in Table 2.1. It shows that most respondents were American (54%), 25-54 years old, had 11 years or more of experience, and had most experience in computer (disk) and mobile forensics. Albeit ages of respondents were evenly spread between age groups from 25 to 54, the years of experience participants reported were uneven, showing peaks at 2-4 years and 11 years or more. Just over half of the respondents were practitioners and most belonged to private organizations not related to the government or law enforcement. Because 28% said they work within education/training/certification (ETC) and over 50% were practitioners there is a chance some trainers and educators may be practitioners as well.

2.6.2 Comparison to Rogers & Seigfried survey

One of the main purposes of our survey was to determine how the view of future challenges within cyber forensics had changed since the Rogers study (Section 2.3.1). A single question asked participants to rank the categories that were formed in the 2004 survey with the results showing the following order according to calculated average rankings:

1. Education/training/certification (ETC)
2. Technologies
3. Tools

TABLE 2.1: Not all participants answered the demographics questions but the lowest number of respondents for any one question was 94, meaning most did.

	Percentage
Region of residence	
North America	56.7
Europe	23.7
Middle East	7.2
South Asia	6.2
East Asia	2.1
Australia	2.1
Russia	2.1
Age	
18-24	6.3
25-34	25.3
35-44	24.2
45-54	25.3
55-64	14.7
65 or older	4.2
Gender	
Female	14.9
Male	85.1
Years of experience in cyber forensics	
0-1 years	11.5
2-4 years	25.0
5-7 years	16.7
8-10 years	11.5
11 years or more	35.4
Primary occupation	
Industry instructor	3.1
Law enforcement practitioner	20.8
Non-law enforcement practitioner	33.3
Professor	14.6
Researcher	16.7
Student	11.5
Occupation category	
Education/training facility/university	28.1
Federal/national law enforcement	7.3
State/local law enforcement	15.6
Military/national security	1.0
Legal system	3.1
Private organization that doesn't fit into any of the above	37.5
Public organization that doesn't fit into any of the above	7.3
Fields of expertise	
Crime scene investigation (first responding)	11.9
Cloud forensics	3.5
Computer (disk) forensics	28.8
Database forensics	1.5
Memory forensics	5.4
Mobile forensics	18.5
Multimedia forensics (audio, video, image, etc.)	5.8
Network forensics	13.5
Software/malware forensics	9.6
Non-traditional forensics (game consoles, printers, etc.)	1.5

4. Evidence correlation

5. Theory/research

6. Encryption

7. Legal/justice system

8. Data acquisition & Funding (tied)

The data were analyzed via a Friedman test[3], determining that there is a significant difference among the 9 categories within a 95% confidence interval. The top two categories (ETC and Technologies) did not change when compared to the earlier survey. However, encryption moved down 3 places, while evidence correlation and theory/research moved up 3 places; this may reflect the current interest in producing automated and new technologies in the field.

2.6.3 Resource allocation

As shown in Figure 2.1, the types of resources having the strongest correlation to being insufficient were personnel and funding. Interestingly, in the Likert scale questions ETC and funding had similarly strong correlations while the ranked question (Section 2.6.2) had a clear separation of the two resources (ETC at top and funding at bottom). This may be due to the question format.

[3] https://en.wikipedia.org/wiki/Friedman_test (last accessed 2017-05-02).

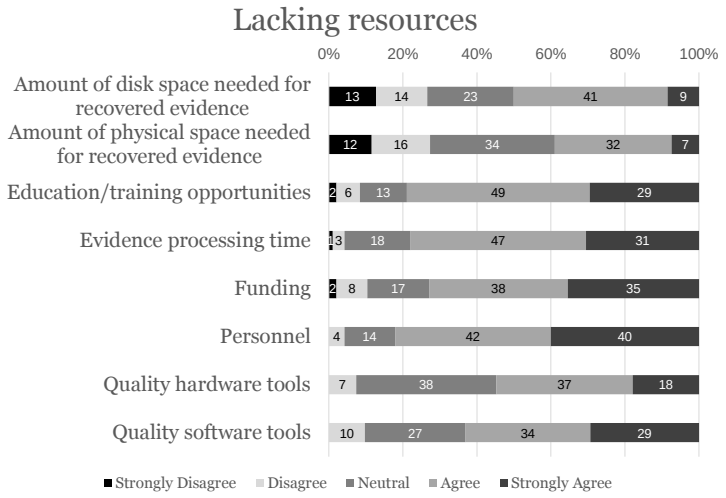

FIGURE 2.1: Each bar represents one Likert scale question. Approximate percentages are displayed for each answer selection.

Unexpectedly, 78% of those who thought public organizations needed more funding were non-law enforcement practitioners (Figure 2.2)[4]. About half of those who chose federal/-national law enforcement were law enforcement practitioners. No bias was observed in the other categories.

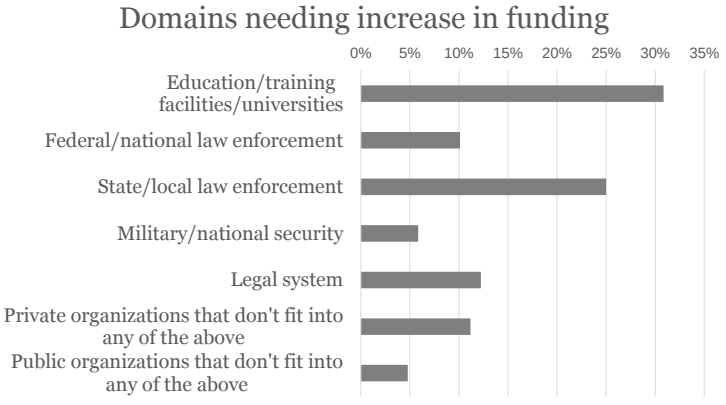

FIGURE 2.2: This multiple selection checkbox question allowed for respondents to select up to 2 of the answers shown above.

[4]Chi-squared test, $p = 0.005$.

Thirty-four people responded to the practitioner-oriented free response question asking to list types of cases personally encountered that need further support/attention. Cloud/-database forensics was mentioned 7 times, mobile forensics 6 times, and non-traditional devices 6 times (satellite, navigation, CCTV systems & game consoles; especially development of tools for these scenarios). Other concerns involved more support for Linux & Mac systems at law enforcement offices, timeline/profiling tools, and chip-off forensics. The concern for mobile support was echoed in another question asking which operating systems needed more support in respect to cyber forensic cases (about 24% selected each Android and iOS, while all other systems were below 12%).

2.6.4 Education/training/certification

As seen in Figure 2.3, the majority of respondents clearly thought state/local law enforcement needs more ETC opportunities. This was mirrored in the practitioner free response where a few respondents mentioned law enforcement & three letter agencies need more education on basics like Domain Name System (DNS) and working with Internet Service Providers (ISPs) or hosting companies.

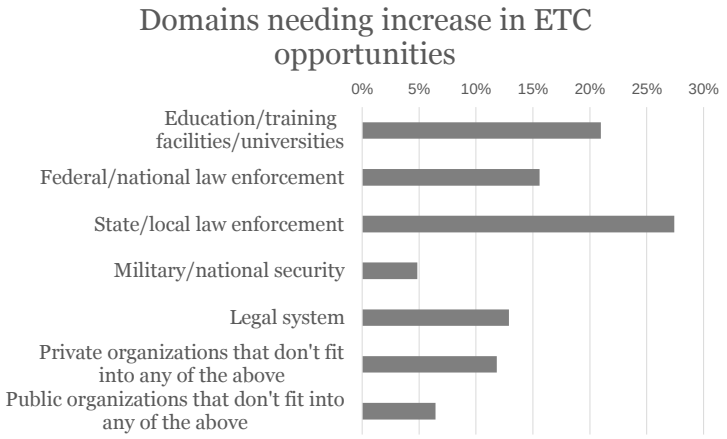

FIGURE 2.3: This multiple selection checkbox question allowed for respondents to select up to 2 of the answers shown above.

The high desire for more ETC opportunities for the education/training facilities/universities category can be interpreted as a need for more cyber forensics programs at universities and offered certifications at training facilities. This concern was less prevalent among Europeans than North Americans; the second most frequent choice for Europeans was federal/national law enforcement rather than ETC. Legal system was selected relatively frequently, which was an outstanding observation considering it was among the smallest occupational demographics.

Figure 2.4 implies practitioners need to know how to use tools, but this is only complementary to a thorough understanding of the forensic process/investigative skills. Reverse engineering also was expressed as a valuable skill for the future, possibly because the plethora of software being developed is increasing and may need some level of reverse engineering to help examiners gain access to evidence. This could be explained by reverse engineering being innately time intensive, thus requiring more experts to combat recovery

time. Or it may be a prerequisite for mobile forensics when encountering devices that are not supported by mobile acquisition and analysis toolkits given their proprietary nature.

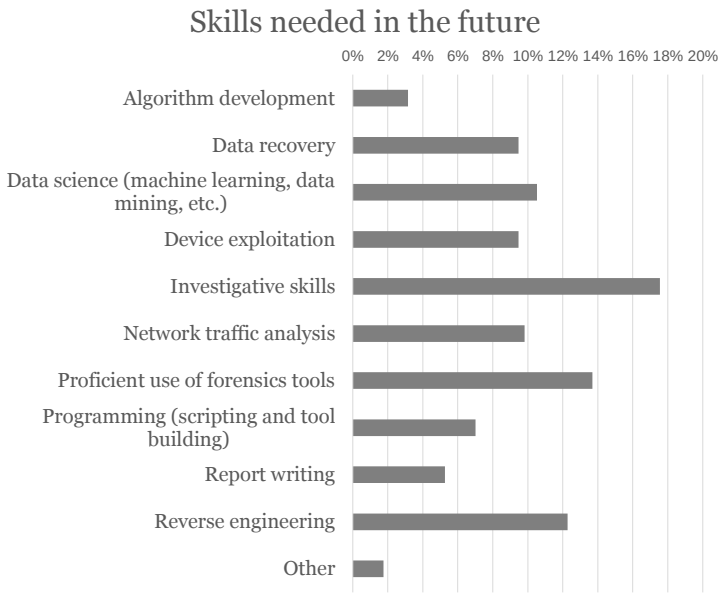

FIGURE 2.4: This multiple selection checkbox question allowed for respondents to select up to 3 of the answers shown above.

2.6.5 Tools & technology

The Likert scale questions in Figure 2.5 clearly show that open-source tools are not meeting the desires of professionals. They need to be both better and funded adequately. Most participants also indicated that commercial tools should be cheaper.

The checkbox question in Figure 2.6 shows that mobile and cloud forensic tools and technology need improvement most. North Americans were most concerned with mobile forensics while Europeans were most concerned with cloud forensics. This could be construed as a result of these domains being newer and currently experiencing rapid growth.

Alternatively, it could be that in Europe there is a larger concern with cloud forensics given their typically more stringent privacy concerns when compared to the United States.

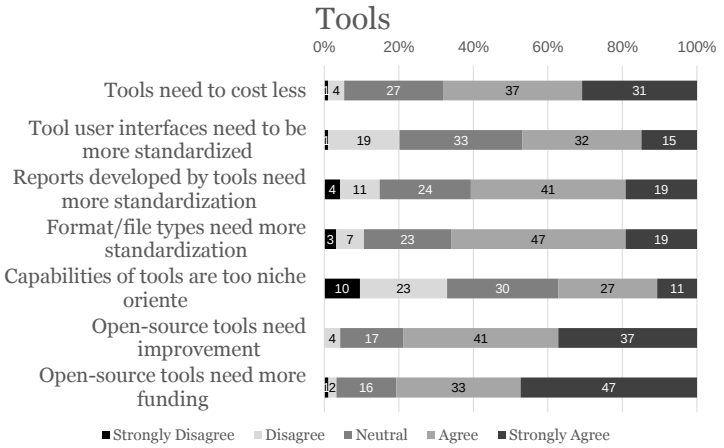

FIGURE 2.5: Each bar represents one Likert scale question. Approximate percentages are displayed for each answer selection.

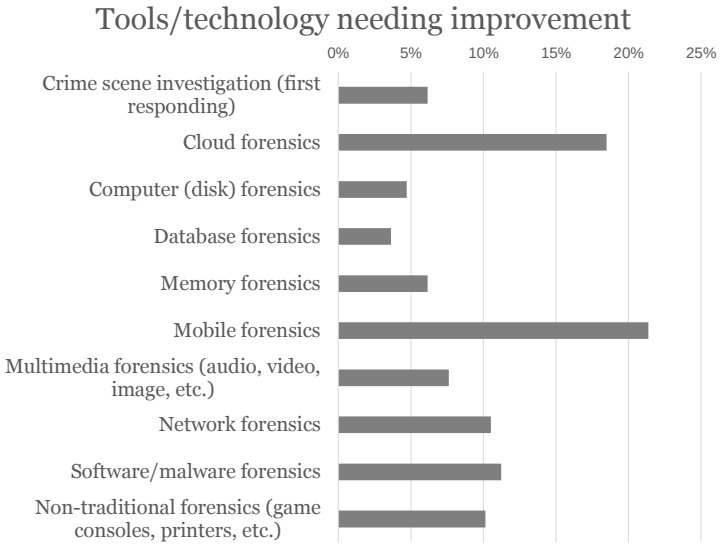

FIGURE 2.6: This multiple selection checkbox question allowed for respondents to select up to 3 of the answers shown above.

Two multiple choice questions were used to assess the use of hashing algorithms which are commonly used in digital forensics. About 42% of respondents claimed they use MD5 the most (another 40% split evenly between SHA1 and SHA-256). We noticed that older

respondents were more likely to use MD5 the most. Despite being considered flawed, practitioners most likely use it because it is fast, short, and the only option in some software. However, this phenomenon could also be due to being unaware of its flaws, or not regarding MD5's flaws significant enough for their purposes or being not aware of newer developments.

To proof our statement that many people are not aware of/avoid new technologies, we asked about approximate matching, a.k.a. similarity hashing or fuzzy hashing, which is a rather new field. Although a definition was only published in 2014 Breitinger, Guttman, McCarrin, Roussev, and White (2014), the first algorithm `ssdeep` came out eight years earlier Kornblum (2006). Only 13% of respondents use these algorithms on a regular basis while 34% only had used them a few times before. Thirty-one percent said they had not used them because they were not necessary for their purposes, 7% reported they are too slow for practical use, and 15% did not know what similarity hashing was. Europe is ahead in adopting this technology (68% of Europeans had used it before compared to only 40% of North Americans).[5]

2.6.6 Research

The Likert scale questions about research (Figure 2.7) indicated a strong need to research encryption, malware, and trail obfuscation countermeasures. Criminal profiling systems, data wiping, and evidence displayed opinions closer to a neutral standpoint.

[5]Chi-squared test, p = 0.025.

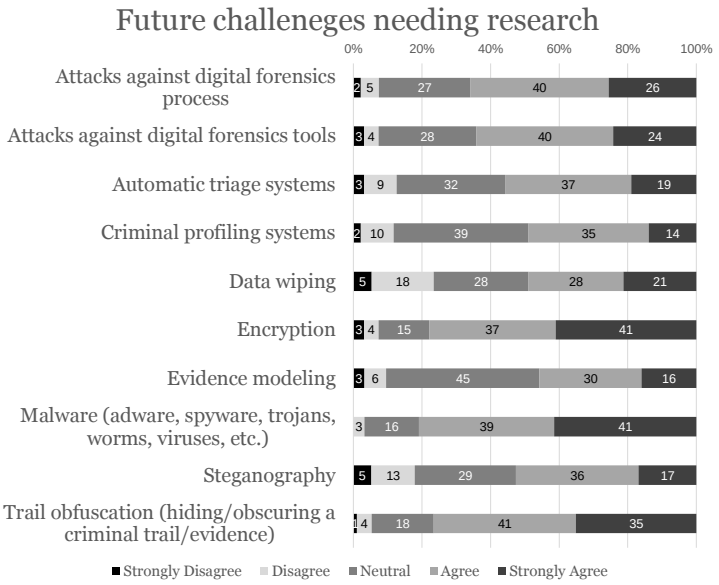

FIGURE 2.7: Each bar represents one Likert scale question. Approximate percentages are displayed for each answer selection.

A free response research-oriented question investigated what topics participants thought will be most important to research in the next 5-10 years. The 35 answers most frequently mentioned cloud forensics (10 times) and mobile forensics (6 times). Other common mentions were malware, encryption, solid state drives, and network forensics (the prior two concerns reflected in the Likert scale questions). A few respondents also expressed worry for the future of the Internet of Things/embedded devices.

2.6.7 Laws

There was an overwhelming consensus that laws pertaining to cyber forensics are out of date, as shown in Figure 2.8. The reasonably substantial evidence that user privacy needs to be protected more, along with these opinions, implies a strong need for overhaul since

most of the respondents were practitioners and likely deal with legal issues more directly than non-practitioners. As mentioned in Section 2.3.5, laws have seen sparse attention.

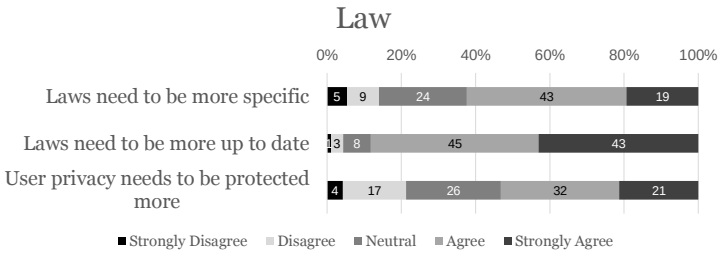

FIGURE 2.8: Each bar represents one Likert scale question. Approximate percentages are displayed for each answer selection.

2.6.8 Communication

The matrix in Figure 2.9 explicitly indicates that state/local law enforcement needs to communicate more effectively (13 occurrences where it was paired with federal/national law enforcement and 12 occurrences where it was paired with legal system). Overall federal/national law enforcement was chosen most for poor communication. Since ETC was also selected frequently, better/increased communication between practitioners and educators/researchers will need to occur.

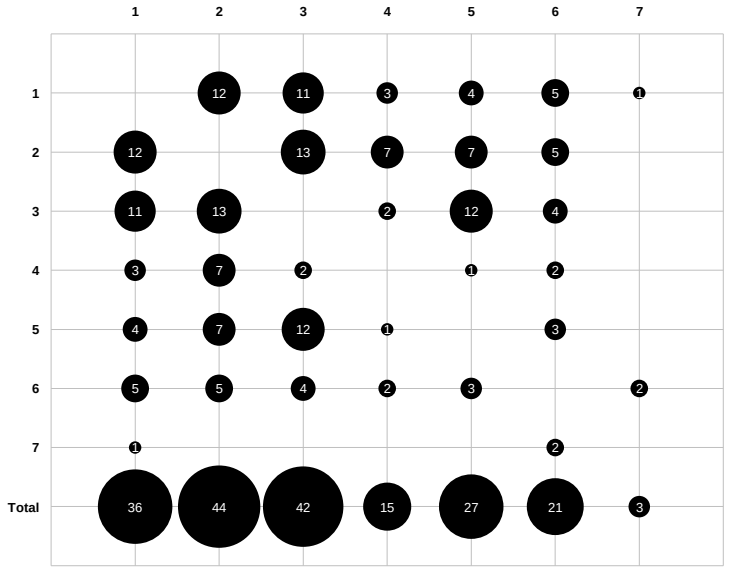

FIGURE 2.9: The matrix represents groups that need to communicate more effectively with each other (participants were asked to select a pair). The sizes of the circles are proportional to their numbers. The total in the bottom row demonstrates how much each group needs to improve on its communication, regardless of who they are communicating with. 1 = Education/training facilities/universities, 2 = Federal/national law enforcement, 3 = State/local law enforcement, 4 = Military/national security, 5 = Legal system, 6 = Other private organizations, 7 = Other public organizations.

Three yes or no questions asked respondents about communication with ISPs (e.g. AT&T, Verizon, and Comcast), online service providers (e.g. Google, Yahoo, Facebook), and computer and mobile manufacturers. In all three of these questions a unanimous call for new established frameworks for communicating with these organizations was observed (over 78% for each). Once again, the relative size of the private organization demographic and the polarized opinions of these questions may mean such frameworks are nonexistent or extremely weak.

2.6.9 Domain categorization

Two general questions were asked about the definition of cyber forensics. The first asked if the participants considered the field a (formal) science to which there was a resounding "yes" (77%). The second question asked whether cyber forensics is an engineering discipline. North Americans were undecided but European input tipped the scale toward an overall 63% affirmation.

2.7 Discussion

The results of the survey signed several things about cyber forensics. As in most technical domains, there is a relatively low number of females in the demographics; possibly the field is still unevenly gender balanced. A second eye-catching aspect was that no respondents were from Africa. This might have to do with the distribution method. Taking a closer look at the fields of expertise shows that only a few respondents had expertise in database forensics and only slightly more had cloud forensics expertise. These were top concerns for the future and although they are newer subdomains more experts must arise quickly if the field is to keep stride with criminals. Certainly, mobile forensics is extremely important as all types of cases involve mobile evidence Saleem, Baggili, and Popov (2014).

The upset with funding in the results can be rationalized by a low federal demographic; federal labs are more likely to "report analyzing digital evidence," implying that non-federal facilities are not equipped well enough to deal with cybercrime Durose, Walsh,

and Burch (2012). Such lack of cohesion within the field is now leading to the call for an official governing body Waziri and Sitarz (2015), something that may also require a federal and non-federal delegation; a distinct difference in concerns between demographics was solidified in a 2009 survey addressing top priorities: law enforcement selected best practice issues as most critical, government selected jurisdictional issues, commercial selected access & exchange of information Liles, Rogers, and Hoebich (2009).

Since the legal system was ranked a relatively low priority in the Rogers & Seigfried followup question one would assume it is not a top priority within the field. However, the Likert scale questions advocated major amendment. Since the legal system demographic was close to non-existent we think a follow-up study would be beneficial — one that targets the judiciary viewpoint. This would be helpful to pinpoint how laws are not specific enough or could be improved upon to protect users and effectively prosecute international criminals. A similar issue could be seen in federal/national law enforcement having the poorest communication of any occupational category demographic; it was the smallest. Perhaps a follow-up study might target federal opinions on the matter.

Many of the results supported recent findings: tools need to be better (quality, usability, & price) and standardization needs to increase across the board (laws, tools, education, & communication); all of these were repeatedly found in research presented in Section 2.3.

2.8 Limitations

The Likert scale questions may have exhibited acquiescence bias (agreement with the statements as presented), because many questions observed affirmation. However, the questions were worded in a way to avoid this and this may have simply been a result of grouping questions by topic. Bias by geographic region and other demographics was not observed other than where mentioned; low count of some demographics prevented drawing further conclusions.

2.9 Conclusions & follow-up

A divide still exists between what professionals desire and what is currently occurring within cyber forensics. ETC and technology still remain the highest priorities for change. State/local law enforcement and ETC facilities need more ETC opportunities (whether it be newer programs or revised curricula); in Europe federal/national law enforcement is also of concern. ETC requires more funding — as an example, the Regional Computer Forensic Laboratory facilities only used 2% of their funding in 2012 on training/educational material; in 2013 the number of trained employees was even lower than the previous two years Regional Computer Forensics Laboratory, U.S. Department of Justice (2012, 2013). Surprisingly, reverse engineering is viewed as a skill future certified examiners should have in addition to fundamental investigative skills and ability to use tools. The

most evident finding on tools was that open-source tools need much more support and improvement. Additionally, most respondents pointing out tools are too expensive.

One of the themes of the survey responses was the need to pay greater attention to cloud and mobile forensics. Not only are these subdomains in need of support (referring to technological repositories and communities useful for practitioners), but they also need more research. Cloud and mobile forensic tools are lacking when compared to other subdomains.

Another crux among the results was sluggishness to adopt newer technologies and ideas. MD5 is still used by most practitioners despite its flaws. Less than a fifth of professionals don't know what similarity hashing algorithms are; nevertheless usage is low among those who do know (especially among North Americans when compared to Europe). Laws are perceived to be out of date, not specific enough, and insufficiently protective of user privacy.

Thirdly, communication is a substantial problem. There appears to be a disconnect between educators/researchers and investigators, and ineffective communication between law enforcement and service providers/ISPs warrants the establishment of new correspondence systems. Setting this up will demand a stepwise coordinated implementation since federal/national law enforcement has problematic communication efforts at the moment (in the eyes of practitioners) and state/local law enforcement needs more funding.

Other significant results were that research has moved up in priority in the last decade (malware, encryption, and trail obfuscation now viewed as essential areas of focus) and

that the domain lacks personnel. A recent survey supports this, writing that forensic departments do not have enough personnel to process the high number of cases, no matter what tools are used Goodison et al. (2015). This study also strongly (the likelihood of incorporation was measured directly) supports the aforementioned need for ETC reform, recommending digital evidence training be incorporated into both law enforcement and judicial system curricula.

Followup Delphi-method-based studies and surveys would be extremely beneficial to target more narrow and well-defined solutions (such as those mentioned in the Discussion section).

Chapter 3

CuFA: a more formal definition for digital forensic artifacts

3.1 Abstract

The term "artifact" currently does not have a formal definition within the domain of cyber/digital forensics, resulting in a lack of standardized reporting, linguistic understanding between professionals, and efficiency. In this paper we propose a new definition based on a survey we conducted, literature usage, prior definitions of the word itself, and similarities with archival science. This definition includes required fields that all artifacts must have and encompasses the notion of curation. Thus, we propose using a new term – curated forensic artifact (CuFA) – to address items which have been cleared for entry into a CuFA database (one implementation, the Artifact Genome Project, abbreviated

as AGP, is under development and briefly outlined). An ontological model encapsulates these required fields while utilizing a lower-level taxonomic schema. We use the Cyber Observable eXpression (CybOX) project due to its rising popularity and rigorous classifications of forensic objects. Additionally, we suggest some improvements on its integration into our model and identify higher-level location categories to illustrate tracing an object from creation through investigative leads. Finally, a step-wise procedure for researching and logging CuFAs is devised to accompany the model.

3.2 Introduction

Currently, the use of the term "artifact," or "artefact" (United Kingdom spelling), in relation to digital information and cyber/digital forensics embodies a variety of meanings depending on the context used as well as the perspective of the user. The term has generally been adopted within the cyber forensics domain for items of interest that help an investigation move forward. Notwithstanding, the lack of a formal definition and sound ontology is holding the field back from forming standards to keep pace with cybercrime Brinson, Robinson, and Rogers (2006). Note that the term should not be confused with the software development interpretation of the word (a tangible by-product produced during software development, especially pertaining to such methods/processes)[1].

Without a systematic ontology, scientists and practitioners have different ideas of how knowledge is related within the context of their situations. Ontology provides an essential

[1] `http://forensicswiki.org/wiki/Computer_forensics#Artifact` (last accessed 2017-05-02).

"unifying map of concepts and relationships" (for more explanation on the importance and creation of ontologies/taxonomies see (Malafsky & Newman, 2009)). Chiefly, professionals (in different subdomains) cannot easily share evidence and often are forced to rely exclusively on their own past experience during investigations, which may cause missed evidence or leads. This becomes extremely important with the ubiquity of devices and software applications used today. Adopting an ontological system should increase the ability to bring to light connections investigators are unaware of, such as linked cases that involve the same criminal, or missing data in a specific location that indicates system tampering.

In addition to the ontology, it is important to develop a standardized taxonomy so that reports can be developed by software/investigators easily via a procedure for the process of researching and handling items (curation). By using dynamic (optional) and required fields, artifacts extracted using various tools would be directly comparable. Currently this is not the case for cyber forensics (e.g. "SerialNum" on Windows vs. "Serial Number" on OSX), even though such classification exists for biological forensics Brady, Overill, and Keppens (2014). The open-source CybOX project[2] is one increasingly popular attempt at standardizing such fields. Object types encapsulate these fields making items placed in them close to mutually exclusive, but like the prior example there lacks details that help experts enter data on cyber items (files, processes etc.). Conventions are especially lacking with respect to presentation of evidence in courts Bariki, Hashmi, and Baggili (2011b).

[2] `http://cyboxproject.github.io` (last accessed 2017-05-02).

Our contribution aimed to solve the aforementioned challenges (standardization/cohesive understanding and standardization of practitioner-oriented information exchanging). Primarily, a survey was designed to ask practitioners and researchers how they would define a "digital forensic artifact". Based on this, previous adoptions in academic literature, and the domain of archival science we accomplished the following:

1. Proposed a more concrete, unified linguistic definition, assigning it a new name: Curated (digital) Forensic Artifact (CuFA).

2. Using survey responses and our proposed definition, we designed an ontological model for curation of artifacts that involves a procedure and sets the requirements for an object to be considered a CuFA.

3. Presented a manner for implementing the higher-level ontology in conjunction with a low-level schema (CybOX) resulting in a searchable database organized by dynamic, taxanomic fields and tags/flags.

The structure of this paper is as follows: first, we cover past definitions/usage of the term "artifact" (Section 3.3.1), review previous ontologies (Section 3.3.2), and deliver a brief comparison to archival science. Next, our survey methodology (Section 3.4.1) and design (Section 3.4.2) are introduced, followed by the results in Section 3.4.3. Using these findings we propose a definition and an ontological model based on this definition in Section 3.5. Finally, discussion and suggestions for future work are presented to the reader.

3.3 Previous work

3.3.1 Developing a definition

This section reviews past definitions of the term "artifact" in the context of digital evidence, the types of items both researchers and organizations have used the term to describe (including the perspective that drives these usages), and previously proposed ontologies.

3.3.1.1 Definitions

All definitions listed below are word-for-word. (Merriam-Webster Dictionary, 2015) defines "artifact" as:

- An accidental effect that causes incorrect results.
- Something characteristic of or resulting from a particular human institution, period, trend, or individual.
- A product of artificial character (as in a scientific test) due usually to extraneous (as human) agency.

(Oxford Dictionaries, 2015) lists:

- An object made by a human being, typically an item of cultural or historical interest.

- Something observed in a scientific investigation or experiment that is not naturally present but occurs as a result of the preparative or investigative procedure.

(Dictionary.com, 2015)'s definitions include:

- Any object made by human beings, especially with a view to subsequent use.
- A substance or structure not naturally present in the matter being observed but formed by artificial means.
- A spurious observation or result arising from preparatory investigative procedures.
- Any feature that is not naturally present but is a product of an extrinsic agent, method, or the like.

More specific definitions were obtained from the CybOX project MITRE Corporation (2015):

- An object produced or shaped by human craft, especially a tool, weapon, or ornament of archaeological or historical interest.
- A phenomenon or feature not originally present or expected and caused by an interfering external agent, action, or process.

Another digital-scoped definition came from the Scientific Working Groups on Digital Evidence and Imaging Technology SWGDE/SWGIT (2015):

- Information or data created as a result of the use of an electronic device that shows past activity.

There were a few instances where papers made explicit attempts to bound their usage of the term, and therefore provided a definition. In one case it was stated that artifacts should not be confused with Indicators Of Compromise (IOCs), items that signify a system's compromise, as their intent is different and they represent pure data without logic, i.e. a system state rather than malware state Castle (2014b). The example Castle gave: an IOC might be an executable that contains a string "evil" or is signed "stolen cert," while an artifact could be the location where user runkeys are located (`HKEY_USERS\%%users .sid%%\Software\Microsoft\Windows\CurrentVersion\Run\*`). Wikipedia contradicted this by using the word "artifact" in the definition of IOC[3]. Castle's IOC definition also disagrees with the previous usages presented in Section 3.3.1.2. SysAdmin, Audit, Networking, and Security (SANS) defines an artifact as a "combination of description, location, and interpretation" Castle (2014a).

The commonality between these definitions appears to be observed artificiality/external force, antecedent temporal relation, and exceptionality (based on either accidental procurement, rarity, or a person's interest). Including the word "forensic" at the beginning of the term adds legality and science to this list. These cannot be used exclusively to form a definition, however – academic and community usage must be examined.

[3] https://en.wikipedia.org/w/index.php?title=Indicator_of_compromise&oldid=666037196 (last accessed 2017-05-02).

3.3.1.2 Perspectives and usage

Citations of the term "artifact" in cyber forensics have varied based on the professional goals of the users' subdomains and the tasks they were performing. Reviewed papers (see the table in the Appendix for the full list of papers and perspectives) used the word mostly in an ad hoc manner that reflected the concept of exceptionality via personal interest; thus, this perspective was the most variant and the term took a different specific meaning in each paper (e.g. log data in (Yasin & Abulaish, 2013) vs. installation/runtime/deletion behaviors in (Lim, Park, Lim, Lee, & Lee, 2010)).

A second trend recorded was that of investigators. Usages of the term in these papers emphasized looking generally for "what you want to know" in order to further an investigation and, consequently, had a broader intention than the academic standpoint. Between these two extremes laid the perspective of those who design, manufacture, and test tools. The primary motive behind this view is the objective of standardizing objects for tagging (or filling in fields/checkboxes), reporting, comparison (exporting to share), and increased investigative efficiency. Note that although this seems similar to the investigative stance, these papers detracted the logical, conceptual aspects described above and attempted to focus on the location and data itself.

Table 3.1 shows an excerpt from the table in the Appendix. Each paper's focus was categorized into one of the above perspectives, or the collection perspective described in the next section, so that the various mindsets could be weighted in our proposed model. The investigative ethos was usually used in conjunction with another one and therefore

cannot be found in the table; we declare the more prevalent view. Note that papers marked with an asterisk did not have explicit categorization of items and required the authors to devise educated groupings.

TABLE 3.1: Excerpt from the Appendix to exemplify the structure of the full table.

Items	**Category**	**Paper & perspective**
Apple system log; Crash reporter; Diagnostic messages; FSEvents API; Preference settings; Saved application state; Spotlight; Swap files/-paging/cache; Temporary data;	OSX	Sandvik (2013)*, Researcher
Prefetch; Thumbnail cache; Paging file; Registry; Windows search;	Windows	
Bash history; GVFS virtual file system; Recently used; X session manager;	Linux	

3.3.2 Outlining an ontological model

(Casey, Back, & Barnum, 2015) reviewed past ontologies and stressed that "querying data on the basis of high-level behaviors [...] can be more powerful than just searching for low-level digital artifacts". In this section, we highlight the advantages and disadvantages of these ontologies and introduce the CybOX model. These will be used to feature a basic ontological model, delivered in Section 3.5, stemming from our proposed definition.

Traditionally, "ontology" involves the study of existence, the categories of being, and their relationships. However, in computer science the "intention is distinct: to create engineering models of reality, artifacts which can be used by software, and perhaps directly interpreted and reasoned over by special software called inference engines, to imbue software with human level semantics" Poli, Healy, and Kameas (2010). This form of ontology is sometimes referred to as "Little o" ontology, while "Big O" ontology signifies the philosophy-centered definition. There is some overlap between the two, but we were

primarily concerned with "Little o". Note, hereafter we use the term "model" as an umbrella term for both an ontology (we define this as high-level) and a taxonomic schema (we define this as technical/low-level).

3.3.2.1 Ontologies

Before reviewing the conceptual ontologies and technical schemas proposed in the last few years it is important to understand what problems these models attempt to solve. Knowledge and correlation are the main concerns for investigators, i.e. knowing where artifacts are located, knowing how they can assist a case, and connecting artifacts across locations/devices when they may be recorded in different ways Brady et al. (2014). As stated in the Introduction, increasing the chances of finding leads an investigator is unaware of is also a top priority, whether this be an unfamiliar file type, missing data that indicates a system's state has been changed, or a criminal's modus operandi. A feature many of the following ontologies have is extensibility – the advantage of representing low-level taxonomic data in a way that can be utilized flexibly by high-level ontologies. We categorize these models as a fourth type of perspective in addition to the perspectives presented in Section 3.3.1, that of a (database) collector/designer.

The Forensic Wiki[4] represents a loose catalog of tools, types of digital objects obtained from them, and general cyber forensics topics. However, (Brady et al., 2014) identified that its "value would be further enhanced if it used some form of classification or tagging

[4] `http://forensicswiki.org` (last accessed 2017-05-02).

system that allowed examiners to readily access what artifacts were available and how these could be linked across its various categories". The authors suggested to solve these issues by proposing the Digital Evidence Semantic Ontology (DESO), an investigative perspective of data that views artifacts through the scope of location or type superclasses.

In DESO, superclasses inherit other subclasses and attributes. For example, the location class may inherit devices, file systems, and operating systems subclasses (describing specific categories of locations), while the type class may inherit device identifier and logical identifier subclasses. When attempting to further an investigation the two primary classes accompany each other. If an investigator has obtained a specific artifact they may use the location class to look for that artifact type across different devices, file systems, and operating systems; alternatively if the type class is the same for artifacts extracted from various locations they can be compared directly.

Other high-level ontologies include Structured Threat Information eXpression (STIX), Digital Forensics Analysis eXpression (DFAX), and Unified Cyber Ontology (UCO) Casey et al. (2015). STIX uses CybOX (presented in the next subsection) to represent technical details (e.g. malicious IPs) in a manner that mirrors subdomain-specific information such as threat actors. It is the predecessor to DFAX which also attempts to use a third-party schema within a broader ontology to capture more procedural aspects such as chain-of-custody, case management, or processing. Fields such as *ActionPattern* and *ActionLifecycle* allow users to adopt them for documenting the investigative process, and fields for recording event times allows piecing together timelines and collusion between criminal entities. UCO is an ontology illustrating even more abstract concepts that are

linked across the cyber forensics domain. It requires the usage of a lower-level schema and potentially could use more than one schema at the same time for different subdomains.

3.3.2.2 Schemas

In this subsection we briefly describe the pros/cons of the following schemas: XML Information Retrieval Approach to digital Forensics (XIRAF), Digital Forensics XML (DFXML), and Cyber Observable eXpression (CybOX). XIRAF was a prototype for an XML-based schema proposed by the Netherlands Forensic Institute, but its use of parent-child relationships between objects limited its flexibility and it did not gain anticipated prominence outside of the Netherlands. A subsequent XML-based proposition was DFXML, also an attempt to introduce a structure to the presentation of objects S. Garfinkel (2012). Although the format enabled cross-platform comparison and sharing, it still has not been adopted as a standard, perhaps because XML is a verbose language or because some think it is too limiting (without an accompanying ontology).

Recently, CybOX has gained popularity due to its open-source nature and rigorous classification scheme for objects. CybOX utilizes a long list of required and optional attributes for each object type, which it classifies mainly by where the artifact came from conceptually. Each object is given a Globally Unique Identifier (GUID) to make it easily searchable in a database. One concern CybOX addresses is recording the state of a system before and after an event (e.g. version of a file). Differences can be logged specifically (new file created, timestamp) or statistically (similarity digest). CybOx has begun to

be implemented in Trusted Automated eXchange of Indicator Information (TAXII) and other models Casey et al. (2015). Nevertheless, it lacks the ontological vantage point of high-level ontologies like DFAX – thus, we use it as the low-level building block for our model.

3.3.3 Archival science

Novel work by (Dietrich & Adelstein, 2015) made comparisons between the fields of cyber forensics and archival science. Both fields use procedures involving acquiring, authenticating, and preserving items in a way that minimizes alterations (and documents them). Aside from maintaining their integrity, items must be able to be easily retrieved for future examination and analysis. As stated by the authors, this is one area where cyber forensics differs from archival science: "most criminal forensic organizations have no long-term data preservation and maintenance policy beyond physical storage". Thus, this should also be considered when attempting to develop a definition; the aspect of curation is what gives items the name "artifact" and sets them apart from items not analyzed within procedures followed by experts.

3.4 Survey

3.4.1 Methodology

The following basic methodology was applied in carrying out the survey:

1. Performed comprehensive literature review which informed the researchers that there was neither a consensus in the usage of the term "artifact" nor a concrete definition.

2. Designed a survey around asking respondents to define the term and list possible categories/fields that would help organize such items.

3. Obtained a category two exemption from the Institutional Review Board (IRB) at the University of New Haven restricting the survey from recording participant identification information or behavior, and disclaiming that it posed risk or harm to subjects not encountered in everyday life.

4. Distributed the survey via list servers and LinkedIn.

5. Retrieved data by exporting the coded responses to an XLSX file from the survey system.

6. Analyzed the data for creation of the definition and ontology in conjunction with past work.

3.4.2 Survey design

Questions were formulated based on what the authors found in literature – a missing cohesive definition for the word "artifact" in the context of the domain, and absence of a comprehensive ontology for entering and organizing such items based on static and

dynamic fields. There were two iterations of the survey and a testing round before opening it to the community. The survey consisted of 70 questions:

- 54 Likert scale
- 12 free response
- 4 multiple choice

According to IRB regulations at our institution, participants could not be forced to answer any single question. The target audience was all professionals in the field who had encountered items referred to as "artifacts".

3.4.3 Results

A brief note about the Likert scale figures in this section: Each bar represents one Likert scale question; approximate percentages for each answer selection are displayed within their respective segment; and the number of respondents per question is visible in brackets to the right of each bar.

The survey was open for 2 months before data was exported from the survey system[5]. There were 87 respondents, but 50 of them were excluded for only answering demographic questions; the results summarized below only account for the other 37 participants. It is likely this occurred because these participants wanted to just view the survey questions. A

[5]Raw data, tabulation, graphs, and the survey itself are publicly available on our website: http://www.unhcfreg.com

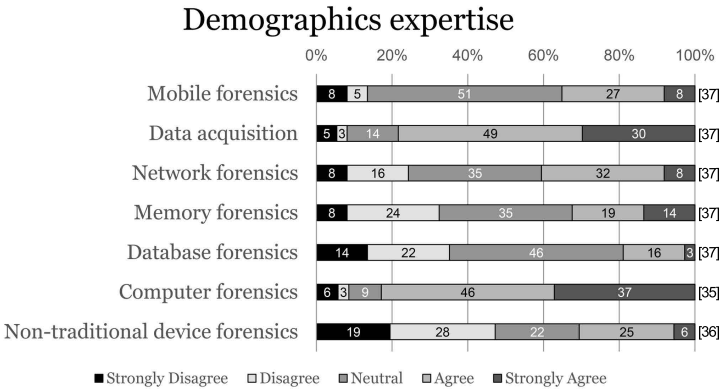

FIGURE 3.1: Results of demographics questions which asked respondents how much they considered themselves experts in the stated subdomain.

power calculation was not performed for the data because it is mostly descriptive/qualitative, lacking any statistical inferences; we recommend the results be used for determining the effect size for any followup work. As seen in Table 3.2, more than half of the respondents were Americans, had at least 7 years of experience in digital forensics, and were over the age of 34. Respondent expertise can be viewed in Figure 3.1.

Since there was no strong agreement on the definition of a forensic artifact, the responses in the survey spanned a myriad of positions. The general themes are shown below, many of which involve direct quotations. In the following results parenthetical numbers next to responses indicate the number of times a general idea or specific words were mentioned.

- Something that has "evidentiary value" in a legal proceeding (7).
- The results of "applying digital forensic (analysis) techniques" (4).
- Byte stream/sequence (2).
- Something of probative interest/yielding information about a digital device (2).

TABLE 3.2: Numbers in the table are rounded and thus may exhibit rounding error. The following was not disclosed: two people did not rate their expertise in computer forensics, one did not rate expertise for non-traditional forensics, two didn't describe their experience in the field, and one person did not disclose their gender. These percentages only account for the 37 that answered the non-demographic questions.

	Percentage
Age	
18-24	3
25-34	30
35-44	24
45-54	30
55-64	11
65-74	3
75 or older	0
Gender	
Female	16
Male	81
Other	3
Country	
Antigua and Barbuda	3
Argentina	3
Canada	5
Germany	8
India	5
Russia	3
Togo	3
Turkey	3
United Arab Emirates	3
United Kingdom	14
United States	51
Region	
North America	57
Europe	22
Asia	8
Middle East	5
Africa	3
Caribbean	3
South America	3
Years work experience in digital forensics	
1-3 years	24
4-6 years	22
7-9 years	16
10 years or more	38

- Digital item/data (2).
- Smallest unit of evidence that can make sense for a digital investigation (timestamp, database entry, etc) (2).
- Something used to reconstruct a crime/events (1).

- File states (1).
- An extraction with an established "data type" (1).
- Semantically annotated metadata (1).

As mentioned in the Introduction, it was important to find a procedure to research and process items. Two separate free response questions asked respondents to reveal their "investigative process" with familiar and unfamiliar artifacts. Four procedures were mentioned for the first question (familiar) and six procedures for the second (unfamiliar). Since mostly similar steps were stated in both questions, we combined these into a proposed procedure for general guidance, which we hope will serve as a method of standardization to be taught to training professionals:

1. Acquire (identify which tool the artifact came from).
2. Backup.
3. Check database to see if encountered before (this can be done by comparing hashes or fields).
4. If familiar, do quick search in artifact database to see if methods used previously are still applicable/effective. If they are, use them, then jump to step 8. If they aren't or the artifact is unfamiliar continue to next step.
5. Classify into a category using the proposed ontological model and catalog/extract artifact qualities (taxanomic fields used in schemas).

6. Attempt to use techniques effective for that category. If ineffective, repeat steps 4-6 until effective and skip to step 8.

7. If no effective techniques are encountered try reconstruction to see if the artifact can be recreated or reverse engineered. Usage of a hex editor may be useful.

8. After a technique is successful in analyzing, repairing, isolating, or rendering the artifact harmless the process should be documented (with all relevant artifact fields) and outputted to a report.

9. Examine the system for associated artifacts based on what was discovered/learned. This may involve searching the database for artifacts of the same type, or using the pointers in the artifact's database entry to browse potentially related artifacts in other locations.

10. Prepare the reports of each (type of) artifact for supporting a legal case.

Finally, the survey also aimed to devise a schema for organizing and archiving items through the identification of descriptive taxanomic fields (that can fit into high-order categories). The survey already presented files, databases, registry, and hardware as categories. The fields respondents mentioned were: files (6), network packets (6), memory/memory dumps (4), application data (3), registry entries (2), type & location (2), operating system (2), data structures (2), email & webpages (2), metadata (1), sockets (1), file system (1), hashes (1), stored/volatile (1), category matches between subfields

(1), external corroborating sources (1), processes (1), software (1), users (1), and device configuration (1). Some thought artifacts should be categorized by something more dynamic such as a tree (3), purpose/action (2), and physical/logical/data containers (1).

We decided that these taxanomic fields were well incorporated into CybOX and consequently should be used to improve it. Figures 3.4-3.7 illustrate that most fields from the survey were deemed important to document by the respondents. Some of these are present in CybOX objects already; others are not and should be incorporated in the future. Furthermore, Figures 3.2 (smartphone and laptop items) and 3.3 (hardware category) indicate that most professionals did not strongly support hardware classification as artifacts, when compared to the other responses.

3.4.4 Limitations

Although the sample size was large enough for the purposes of this paper, a larger sample would have been desirable. However, our size should be acceptable due to the modest size of the cyber forensics domain. Consulting sizes of organizations, forums, and groups, such as the Digital Forensics Training group on LinkedIn or the First Forensic Forum[6], is the only current measure of the target audience at the moment. Even so, these can still be considered small since a basic search will result in much larger populations for other domains.

[6] https://www.f3.org.uk (last accessed 2017-05-02).

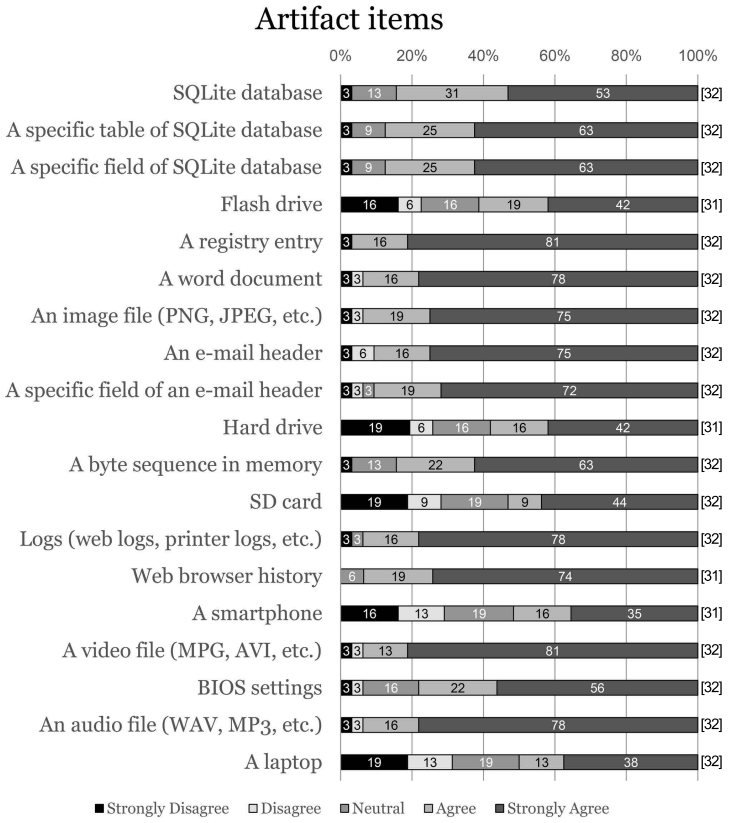

FIGURE 3.2: Results of questions which asked respondents if the stated item should be considered a forensic artifact.

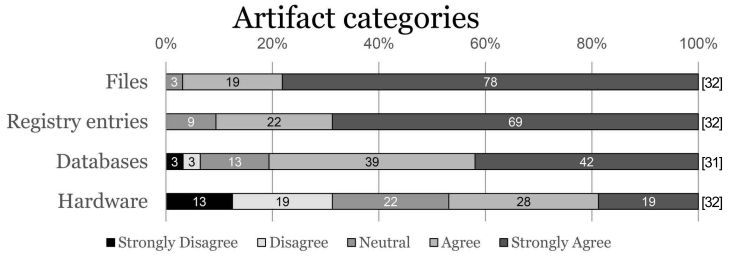

FIGURE 3.3: Results of questions which asked respondents if the stated category should exist for digital forensics artifacts.

Free response questions had a wide variety of answers due to different interpretations (and misunderstandings) of the questions' abstract nature. Many answered within the context of low-level implementation and schemas. These specific answers (not the respondent) had to be disregarded (often they were reiterations of things that already existed in prior work). This could have been eliminated by stating the scope more clearly in each

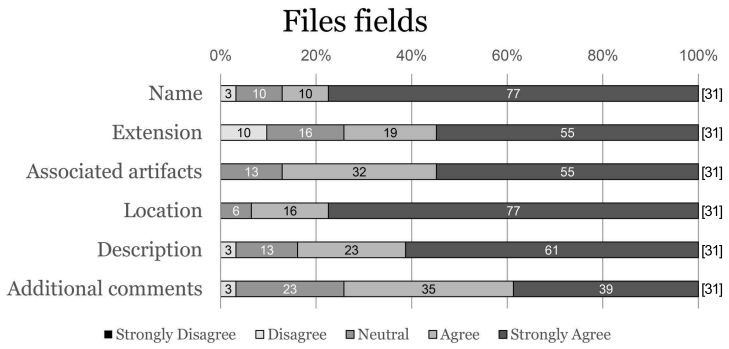

FIGURE 3.4: Results of questions which asked respondents if the stated field should exist for describing artifacts in the Files category.

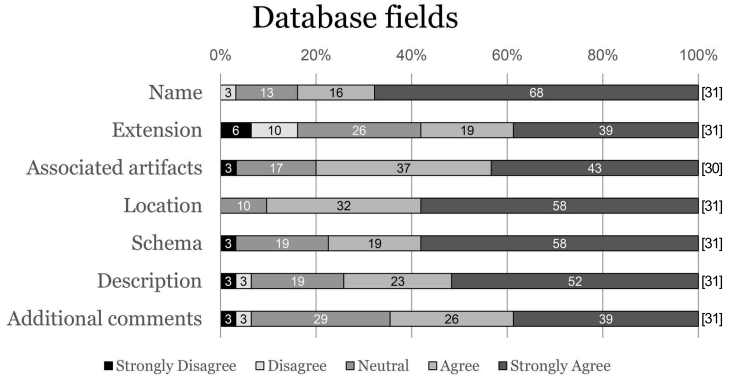

FIGURE 3.5: Results of questions which asked respondents if the stated field should exist for describing artifacts in the Database category.

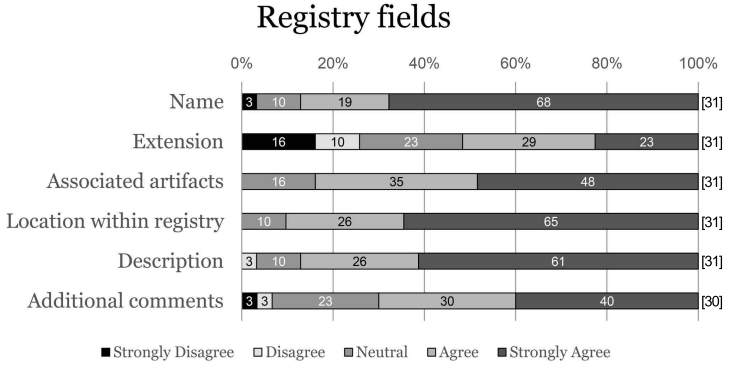

FIGURE 3.6: Results of questions which asked respondents if the stated field should exist for describing artifacts in the Registry category.

question, rather than simply in the disclaimer at the beginning of the survey, which most participants of surveys tend to skip. The survey design could also have been enhanced by having a "decline to respond" option to encourage response rate and minimize "missing

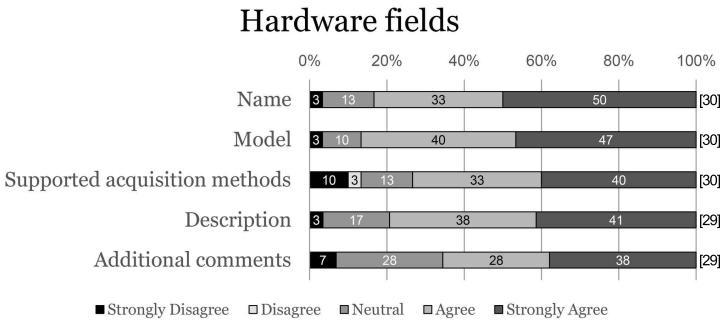

FIGURE 3.7: Results of questions which asked respondents if the stated field should exist for describing artifacts in the Hardware category.

data."

3.5 Proposed definition and model

This section describes our proposed definition, ontological model, and reiterates the importance of having a procedure like the one proposed in the Results section.

3.5.1 Definition

In creating a definition for a forensic artifact we consulted archaeology and archival science. The process of recovering, documenting, and storing objects defines items as artifacts in these domains. Consequently, we added the word "curated" to the term to make this explicitly understood by the community – Curated (digital) Forensic Artifact (CuFA).

By culminating the findings from the survey (only the top two most frequent themes from the bullet point list) presented in Section 3.4.3 and the summation of previous

definitions and common usages (Appendix) discussed in Section 3.3.1 we propose the following stipulations for the linguistic-conceptual definition of a CuFA:

- Must be curated via a procedure which uses forensic techniques, such as the one proposed in the 3.4.3 section.
- Must have a location in a useful format (when applicable).
- Must have evidentiary value in a legal proceeding.
- Must be created by an external force/artificially.
- Must have antecedent temporal relation/importance.
- Must be exceptional (based on accident, rarity, or personal interest).

Despite everything on a computer actually having a location, one must remember that the purpose of a CuFA is to find evidence on varying systems in order to improve future investigations. Therefore, location must be represented in a meaningful format that is most likely static between different devices; the most stable/default format would be disk-related. In other words, memory location is unreliable across devices due to their allocation and runtime usage being different. Disk partitions, sectors, and other representations such as the location of user runkeys mentioned by Castle in Section 3.3.1 are more likely to aid practitioners in looking for evidence across varying models and types of hardware (and maybe even different versions of the same operating system). If no useful format is possible we allow this requirement to be absent (all other requirements must still be met).

Since the definition demands the object be of "evidentiary value" we suggest this requirement be implemented with a tag/flag, which would indicate whether the specific CuFA had been successfully submitted and used in a court of law before or not. Although we found the researcher perspective to be the most common of usages, we prefer to leave this out of the formal definition since it is a consequence of usage without a standardized definition. Regardless, many of the items from the papers in the Appendix would remain identified as artifacts under the CuFA definition.

3.5.2 Ontological model

Based on the aforementioned definition and previous work, we established an ontological model shown in Figure 3.8. The requirements from the proposed definition attempted to unify the variety of items that would be present in a CuFA database; for something of interest to be considered a CuFA it cannot be missing any of these fields (except location). Thus, items of unknown significance should be referred to as potential CuFAs or simply items of interest.

As the results of the survey made evident, location is a consistently desired piece of information, so we determined that using a high-level categorization for it would be useful; hence the *Location type* field. In addition, it will be necessary to store the location of other related CuFAs that were found for a case when making a database entry. In other words, entries should be made after investigations are finished and types of items have been established as evidence, resulting in a linked list (of pointers) of unique CuFAs that

traces the leads investigators took. Although searching by location and type as (Brady et al., 2014) suggested to find new potential CuFAs will still be performed, we feel this extra amount of detail will help investigators better understand the course of action at a brief glance. Possibly, this will resolve the current uncertainty of not knowing how to categorize CuFAs that act as containers for other items, because it will allow layers of abstraction to be clearly understood. For this reason we think a mandatory tag should exist for all entries to show whether the CuFA was a container (found within another CuFA).

This cannot stand on its own though. We decided that CybOX was an excellent, concrete low-level schema to help discoverers curate their artifacts when uploading them to a database such as the AGP (the AGP is currently under development; see Future work for more information). Details of CybOX's design will not be discussed but one area of improvement was identified: CybOX should involve subfields. This is often a matter of design left to the programmer that creates the system. Still, breaking up location fields into disk sector/partition, filepath, key/value pairs, and so on would help to keep interpretation consistent and comparable across platforms and agencies. Similar subfields should be present for other ambiguous fields (e.g. *Device* field in Figure 3.8), and at least one of the subfields required to be completed.

Once more, location must be thought of in terms of the definition. If a CybOX object has a location-related field it may be used to satisfy the CuFA location requirement, as long as it represents a lowest-common denominator format which allows discovery of the CuFA across systems in the future. But it is not mandatory. Stating the physical

location of an item only existing in memory would not aid investigation because this would differ between devices. This does not mean all CybOX objects which satisfy the location requirement have an explicit location field. The *Windows Registry Key* object type has *key* and *subkey* fields which would help locate said objects on disk for different devices. Even though this data may be copied into memory, where an investigator may retrieve it from, the lowest-common denominator format (keys and subkeys on disk) would be logged as the CuFA requirement.

The goal of this type of schema is comprehensiveness along with flexibility. (Brady et al., 2014) mentioned that Encase Case Analyzer can document findings in a SQLite database, but the terminology is inconsistent. Our model could still be used alongside other models (CybOX could be replaced, accompanied, or altered) but would help standardize the items that are entered into databases, the way they are logged, and how investigators interpret the information (leveraging a more investigative viewpoint).

3.6 Conclusions

In this paper we identified requirements all items should have in order to be considered an artifact, and additionally stated that foregoing a curation process should be a new standard within the field. Thus, we suggested all items that meet these artifact requirements be named CuFAs.

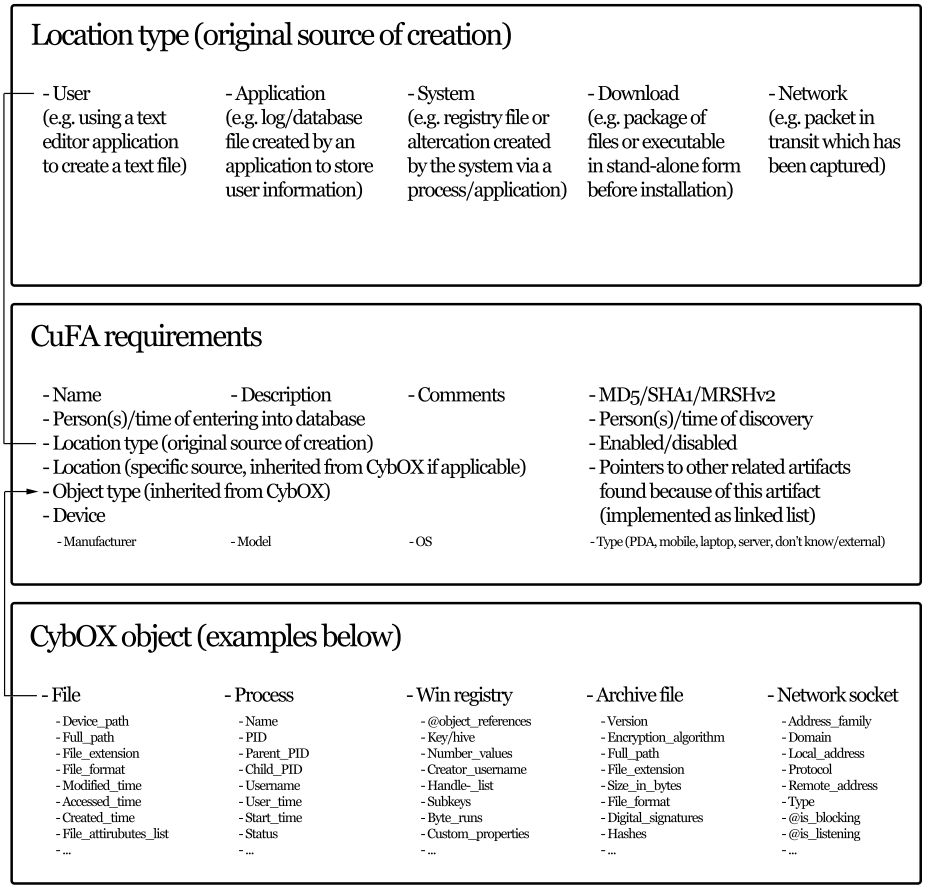

FIGURE 3.8: The proposed ontological model uses *CybOX object* to fill specific low-level fields while the *Location type* attempts to create high-level categorization. All requirements must be met for an object to be considered a CuFA (except location; see Section 3.5.2). The *Object type* requirement field at the end of the arrow illustrates inheritance from the CybOX object (beginning of arrow).

Location type was introduced, centered around the creation location of an item, to incorporate this definition into an ontological model. The model cannot stand on its own though and needs to be coupled with a low-level implementation.

When comparing the results of the survey with the list of items/fields on ForensicArtifacts.com and other sources, it became clear that current models, including CybOX, are still not comprehensive enough (metadata, hashes, file systems, and operating systems

were some missing fields) Castle and Metz (2015). We decided CybOX embodied the most comprehensive taxonomic objects/fields out of current options and offered a couple improvements (again, it could be substituted or used alongside another schema). First, dynamic fields need to be implemented in a manner which creates clear subfields, requiring the logger to choose one and input a correct format, enabling direct comparison of CuFAs. Second, using a linked-list might improve the investigative manner in which artifact databases are used even further, by allowing the logic of an investigation to be retraced from CuFA to CuFA.

Finally, a procedure for curation was identified via survey responses, creating guidelines users of a CuFA database should use to investigate, log, and search. These contributions will increase efficiency and allow better sharing of data, and may facilitate research on "digital evolution" over time/versions.

This initial work may not be enough to create standards immediately. A more comprehensive and larger survey would solidify findings. Thus, we call for collaboration between organizations to attempt to use our findings to develop a more inclusive mechanism for creating a standardized definition. Regardless, we hope the detailed-yet-flexible nature of our model and the obvious trends in definitions/usage will drive discussion and future work to mandate standards based on these ideas.

3.7 Future work

The example of Gene Ontology given by (Brady et al., 2014) supports the need for fields to be linked. Perhaps the easiest way to do this in the future would be to have two linkage boxes (in addition to a value box) for each field where one could select other fields from a dropdown menu to indicate relation. Currently, the Artifact Genome Project (AGP) is attempting to overcome some of the obstacles involved in incorporating a more standardized implementation. It is based on our proposed definition and model, and utilizes elements of CybOX (many of the CybOX objects have already been discarded due to absorption by other re-defined objects, lack of utility in forensics, and over-specificity). The AGP will attempt to create a repository of CuFAs available publicly for researchers, especially, and practitioners to log CuFAs and track investigations via the CuFA linked-lists (effective techniques for recovering, copying, quarantining, and so on may also be logged, in accordance with the procedure presented in the Results). Although many tools, such as GRR, exist to acquire and analyze artifacts the primary point of the AGP will be to create a centralized database regardless of type of tools used in the process. It will also help to standardize the way people upload and categorize CuFAs, helping to reduce wasted time searching for desired types of items. One way it will do this is to pull up possible CybOX objects based on the fields users enter, and then allow users to flag the object they select for different operating systems and levels of legality (has the object officially been used in a court of law); selecting inappropriate object types will become less common.

Accompanying this advancement should also be tools to facilitate the use of such databases. OSXAuditor was one example: it helped to automatically search locations on a running system/system image to find items of interest, extract them, and verify the reputation of files using VirusTotal, MHR, and Malware.lu (or your own database). It could also aggregate logs from locations and put them into a zipball. The final results could be rendered as text or HTML and sent to a Syslog server Roberts (2013). OSXAuditor is no longer maintained and has been set aside for OSXCollector, an automated toolkit oriented towards enabling analysts to answer questions, like how malware got onto an infected computer, quicker. Its single Python file creates a package of the collection (outputed to JSON) and useful files like system logs to pass off to analysts who can look through information on startup, quarantines, operating system info, browsers, downloads, kernel extensions, file timestamps, etcetera Yelp (2016). Tools such as this will be necessary in the future to expedite using CuFA databases and allow investigators to focus on higher levels of abstraction (at least when desired).

3.8 Acknowledgments

We would like to thank Jason Moore for the survey design & testing, as well as Kyle Anthony and Devon Clark for helping in the design of the ontological model based on the data & code they were working on for the AGP. The AGP and the material presented in this paper is supported by the U.S. Department of Homeland Security under Award Number 2009-ST-061-CCI001-05.

3.9 Appendix

For a full explanation of this Appendix see Section 3.3.1.2.

TABLE 3.3

Items	Category	Paper & perspective
User credentials, personal details, activities, location; Activity timestamps;	Databases	Azfar et al. (2015), Researcher
Images;	Media	
Opened/saved files; Email attachments; Skype log (chat & transfer); Index.dat (downloads);	File download	Goh (2014), Researcher
User assist (program launch); Last executed files by app; Run command executed; App compatibility cache; Taskbar jump list; Prefetch/service event logs;	Program execution	
Opened/saved fields; Last executed files by app; Recently opened files; Shellbags; Shortcut files (LNK); Taskbar jump list; Prefetch files; IE history files;	Files created & opened timeline	
Search assistant/history; Keywords search from Start Menu; Last executed files by app; Hidden files in dir (Thumbs.db); Recycle bin; IE history files;	Deleted files	
Current system timezone; Network history; IE cookies; Time website visited;	Physical location	
USB key identification; USB device plug & play times; GUID of mounted devices; Volume serial number; Drive letter & volume name; Shortcut link files (LNK); Plug & play event log;	Drive usage	
Last password change; Last login; Successful/failed login; Login types for account; Remote desktop usage;	Account usage	
IE history; IE cookies; IE cache files for web content; Automatic crash recovery; Local stored object & flash; Network history;	Browser usage	
Contact details & profile; Picture URLs; Photo uploads; Comments posted; Timestamps; Previously logged in users; Friends with active chat; Created albums; Pictures viewed with app; Mailbox/chat messages;	Facebook	Mutawa et al. (2012), Researcher
User names; Profile picture URLs; Tweets posted; Other activity (e.g. device);	Twitter	
Usernames/passwords; Post comments; Timestamps; Cookies & cache files;	MySpace	
Local folder; Metro apps; IE10 websites visited; Journal notes; Desktop tools; Metro app web cache; Metro app cookies; Cache; Cookies; Microsoft folder; Digital certificates; User contacts; Application settings;	Windows	Thomson (2012), Researcher
Ntuser.dat; SAM; System; USB storage devices; Software;	Registry	
Space carved from SSD; EFI-system objects from carving; Grub in boot sector;	Chrome operating system	Corbin (2014), Researcher
User directory of Chrome files; Google website history; Bookmarks; Cookies; Download, search, login history; Most visited websites; Cache;	Contained/inner items	
Temporary content; User content; System support; System updates; File timestamps;	Xbox One NTFS partitions	Moore et al. (2014)*, Researcher
Bit assignments; Browsing content records; Database files; Page files; Log files;	Private browsing	Chivers (2014), Researcher
Registry; Application data folder;	Google client-side	A. Gupta et al. (2013), Researcher
Keyword searches; Usernames/passwords; Most recently used/cache data;	Registry	Mee et al. (2006), Researcher
Text; Images; Sketches; Videos; Location data; Audio; Video;	Smartphone network traffic	Walnycky et al. (2015), Researcher
Chat logs; User info in SQLite files;	Local	
Application settings; Installation paths; Program compatibility assistant; Magnetic/registry key links;	Registry directory & key/value	Lallie and Briggs (2011)*, Researcher
IE integration; Statistics; Open with list; Windows routing service tracing; Remote access service tracing; File associations; Uninstallations;	Other BitTorrent association	
Apple system log; Crash reporter; Diagnostic messages; FSEvents API; Preference settings; Saved application state; Spotlight; Swap files/paging/cache; Temporary data;	OSX	Sandvik (2013)*, Researcher
Prefetch; Thumbnail cache; Paging file; Registry; Windows search;	Windows	
Bash history; GVFS virtual file system; Recently used; X session manager;	Linux	

Items	Category	Paper & perspective
Install path (install/delete); Registry keys (install/delete); Prefetch files (install/delete/runtime); VDF signatures;	Virtual disk encryption tool	Lim et al. (2010), Researcher
USB device database;	Database	Collie (2013), Researcher
User/attacker geoIP, source/private IP, SIP user agent, device, habits; Not found 401, 404; Options method;	Network traffic	Psaroudakis et al. (2014)*, Researcher
Frame time; Source IP address; Destination IP address; SIP from/to; SIP contact; SIP user agent via call-ID; Cseq; SDP owner, connection, session name, media attributes; Info request/response;	SIP/SDP header	
RAM; Swap files; Registry;	Accelerator Plus	Yasin et al. (2009), Researcher
Proxy settings; History of downloaded files; Files requested to download; Incomplete downloaded files; Password protected websites; Site grabber; Uninstall location;	Registry	Yasin et al. (2010), Researcher
Downloaded files; Site grabber; Uninstall process; Encrypted password storage;	Log files	
Log analysis; RAM analysis;	Digsby messaging client	Yasin and Abulaish (2013), Researcher
Registry keys/values; Directories & files;	Steganography	Zax and Adelstein (2009), Researcher
Antivirus/quarantine-related; Authentication; Web browser; Configuration/registry files; Containers for execution events; External media data/events; Log files; Memory/volatile data; Networking state; Running processes; Installed software; System-related; User-related file/type/location;	GRR & ForensicArtifacts.com	Castle and Metz (2015), Collector
Autorun locations; System preferences; System settings & info; Sleep/hibernate/swap image file; Kernel extension; Software installation; Miscellaneous system info; Networking;	System	Stirparo (2015)*, Collector
Autorun locations; Users; User directories; Preferences; Logs; User accounts; iDevice backup; Recent items; Miscellaneous;	User	
iCloud; Skype; Safari; Firefox; Chrome; Mail;	Application	
File downloads; Program execution; File opening/creation; Deleted file/file knowledge; File physical location; USB/drive usage; Account usage; Browser usage;	SANS cheat sheet	Lee (2015), Collector
File; Process; Win registry key; Win service; Win thread; Archive file; Mutex; URI; Domain name, address, & hostname; Port; Network socket; Link; DNS record; ARP cache; URL history; Email message; Socket address; Pipe; Win mailslot; Win memory page region; Win filemapping; Semaphore; Win event; Win critical section; Win handle; WHOIS;	CybOX objects	MITRE Corporation (2015), Collector
Apple serial number ID; Mobile phone handset ID; Network address ID; SIM card ID; USB device ID;	Device identifier	Brady et al. (2014), Collector
File system ID; IP address ID; SSID;	Logical identifier	
IP addresses/domains; Mutexes; Open(ed) files; Services; Registry keys/values/write times; System date; Process names/timestamps; Thread/network timestamps; UserAssist last run times;	Volatility Framework	Levy (2011), Tool
Event logs in XP; PE timestamps;	Volatility Timeliner	
PDF, TXT, RTF, Office, etc. files;	Document files	MAGNET (2015), Tool
USB devices; File system info; Network share info; Link files (shortcuts); User accounts; Startup items; OS info; Shellbags; JUmplists; Event logs; Prefetch files; Timezone info;	Windows	
Outlook web app & email client; Microsoft sharepoint; Mbox email;	Corporate email	
Microsoft Lync/OCS;	Instant messaging	
Calendar; Call logs; Contacts; iMessage/SMS; Native notes;	iOS backup	
SMS & voicemail; Browser; Cell.cache & Wifi.cache; Maps; Pictures; Notes; Contacts & call logs; Downloads; Email; Application snapshots; iOS owner info, notes, wifi/Bluetooth info, user word dictionary, spotlight searches, calendar events, installed applications;	Phone apps	
Network connections; Running processes; Connected network shares/drives; Alert on remote connections; Network interfaces; Logged on users; Scheduled tasks; Services;	Triage	
Instant messaging chats; Media; P2P file sharing; Social networking sites; Webmail applications; Web-related activities; Webpage recovery; Xbox;	Internet	

Chapter 4

Bytewise approximate matching: the good, the bad, and the unknown

4.1 Abstract

Hash functions are established and well-known in digital forensics, where they are commonly used for proving integrity and file identification (i.e., hash all files on a seized device and compare the fingerprints against a reference database). However, with respect to the latter operation, an active adversary can easily overcome this approach because traditional hashes are designed to be sensitive to altering an input; output will significantly change if a single bit is flipped. Therefore, researchers developed approximate matching, which is a rather new, less prominent area but was conceived as a more robust counterpart to traditional hashing. Since the conception of approximate matching, the

community has constructed numerous algorithms, extensions, and additional applications for this technology, and are still working on novel concepts to improve the status quo. In this survey article, we conduct a high-level review of the existing literature from a non-technical perspective and summarize the existing body of knowledge in approximate matching, with special focus on bytewise algorithms. Our contribution allows researchers and practitioners to receive an overview of the state of the art of approximate matching so that they may understand the capabilities and challenges of the field. Simply, we present the terminology, use cases, classification, requirements, testing methods, algorithms, applications, and a list of primary and secondary literature.

4.2 Introduction

It is no secret that the number of networked devices in the world continues to increase alongside the complexity of cyber crimes, size of storage devices, and amount of data. We are beyond the point where investigators can manually analyze all cases. These advances have also been complemented with an increase in processing power. Speaking in numbers, while 80-200 GB HDDs, 2-4 GB of RAM memory, and dual core were the quasi standards for machines in 2011, nowadays they are 512 GB SSDs, 8-16 GB RAM, and multicore architectures; (external) storage devices may have several terabytes of storage. Furthermore, if privately owned computing resources are insufficient for a task, one may shift it to the cloud. In short, practitioners need tools and techniques that are

capable of automatically handling large amounts of data since time in investigations is of the essence.

A common forensic process to support practitioners is *known file filtering*, which aims at reducing the amount of data an investigator has to manually examine. The process is quite simple: (1) compute the hashes for all files on a target device (2) compare the hashes to a reference database. Based on the signatures in the database, files are whitelisted (filtered out / known-good files, e.g., files of the operating system) or blacklisted (filtered in / known-bad files, e.g., known illicit content). This straightforward procedure is commonly implemented using cryptographic hash functions like MD5 (Rivest, 1992) or an algorithm from the SHA family (FIPS, 1995; Bertoni, Daemen, Peeters, & Assche, 2008).

While cryptographic hashes are well-established and tested, they have one downside – they can only identify bitwise identical objects. This means changing a single bit of the input will result in a totally different hash value. Subsequently, the community worked on a counterpart for (cryptographic) hashing algorithms that allows similarity identification – *approximate matching*. Although this is a practically useful concept, a recent survey (Chapter 2) with 99 participants showed that only 12% of the forensic experts polled use this technology on regular basis. Detailed results are provided in Table 4.1.

TABLE 4.1: Answers to the survey question: Have you ever used approximate matching/similarity hashing algorithms?

Answer	**in %**
Yes, I use them on a regular basis.	12.50
Yes, a few times.	34.38
No, they are too slow for practical use.	7.29
No, they are unnecessary for my purposes.	31.25
No, I am unaware of what it is.	14.58

4.2.1 Contribution.

In this paper we aim to address the almost 15 % that have never heard of approximate matching by providing them with a comprehensive literature survey, and the 31 % (unnecessary for my purposes) by illustrating a multitude of applications for approximate matching. Accordingly, we address the following key points:

- Terminology, use cases, classification, requirements, and testing.
- High-level description of existing algorithms including strengths and weaknesses.
- Secondary literature that enhances / assesses existing approaches.
- New applications that employ approximate matching, e.g., file carving and data leakage prevention.
- Current limitations and challenges, and possible future trends.

Since it is low-level (is directly concerned with the structure of everything digital), and may be the most impacting / implemented type due to its usage for automation, we focus on *bytewise* approximate matching.

4.2.2 Differentiation from previous work.

When writing this article, there were three articles similar to this survey. The first was the SP 800-168 from the National Institute for Standards and Technology (NIST, Breitinger,

Guttman, et al. (2014)). While this article provides an overview of the terminology, uses cases, and testing, it does not include any algorithm concepts, applications, or critical discussion. Moreover, a reader is not provided with a long list of references. Martínez, Álvarez, and Encinas (2014) is a purely technical paper and focused on the full details of the algorithms and their implementations. Thirdly, the dissertation from Breitinger (2014) contained almost all of these topics but is extremely lengthy. Ergo, our intention during writing was to make this publication the primary source for researchers / practitioners to grasp a cursory bird's-eye view of bytewise approximate matching.

We summarized the most important elements of these works in a condensed and direct manner to increase the awareness of approximate matching. Extra texts are also shared for each algorithm in Sec. 4.5.

4.2.3 Structure.

The remainder of this paper is organized as follows: Sec. 4.3 provides the historical background of approximate matching. Concepts are outlined, including use cases, types, requirements (this subsection describes the core principles of algorithm design), and testing. Then, after traversing 8 of the most popular algorithms in Sec. 4.5 we mention newly explored prospects in Sec. 4.6. Limitations and challenges precedes a brief listing of future areas of research in Sec. 4.8.

4.3 History

Many of the approximate matching algorithms designed to solve modern-day problems in digital forensics rely fundamentally on the ability to represent objects as sets of features, thereby reducing the similarity problem to the well-defined domain of set operations (Leskovec, Rajaraman, & Ullman, 2014). This approach has roots in the work of the early Swiss 20^{th} century biologist Paul Jaccard, who suggested expressing the similarity between two finite sets as the ratio of the size of their intersection over the size of their union (Jaccard, 1901, 1912): if A and B are sets, then the Jaccard index J is defined as $J(A,B) = \frac{|A \bigcap B|}{|A \bigcup B|}$. It has been widely adopted as a method for quantifying similarity and is still used mainly within computer linguistics for plagiarism detection.

Nearly a century later, Broder (1997) proposed using the Jaccard index as part of his algorithm for identifying similar documents. Broder suggested a distinction between two commonly used types of similarity: 'roughly the same' (resemblance) and 'roughly contained inside' (containment). While he recommended using the Jaccard index for resemblance, he introduced a variation to approximate containment which "indicates that A is roughly contained within B": $c(A,B) = \frac{|A \bigcap B|}{|A|}$. Additionally, Broder described the `MinHash` algorithm, an efficient method for estimating these similarities (Broder, Charikar, Frieze, & Mitzenmacher, 1998).

On the other hand, Manber (1994) presented `sif`, an implementation used to correlate text files. "Files are considered similar if they have a significant number of common pieces, even if they are very different otherwise." Due to the complexity of comparing

strings directly, he utilized Rabin fingerprinting to hash and compare substrings (Rabin, 1981).

A first step towards approximate matching as we use it today was `dcfldd`[1] by Harbour in 2002, which was an extension for the well-known disk dump tool `dd`. His tool divided the input into chunks of fixed length and hashed each chunk. Therefore, Harbour's approach is also called *block-based hashing*. While this approach works perfectly for flipped bits, it has a limited capacity to detect similarity in strings where the deletion or insertion of bits creates a shift that changes the hashes of all blocks that follow. Theoretically, the shift of even a single bit at the beginning of a file could cause nearly identical objects to appear to have nothing in common, much like the naive (traditional) file hashing approach.

Although this weakness can present a problem for file-to-file comparison, it may be acceptable in some scenarios. For example, if the goal is to determine which parts of a disk image might have been changed during a cyber attack, Harbour's technique remains useful. Likewise, in the case of an analyst scanning for blacklisted material across a drive or a collection of drives, the loss of a few block matches may be a worthwhile trade-off for gains in speed and simplicity, particularly because a single block is often sufficient evidence to demonstrate the presence of an artifact, or at least to warrant closer inspection.

Several efforts have been made to further leverage this technique for detecting similar material by matching fixed length file fragments. Collange, Dandass, Daumas, and Defour (2009) coined the term "hash-based carving" to describe this method of scanning for

[1] `http://dcfldd.sourceforge.net` (last accessed 2017-05-02).

blacklisted material, since it can be used to extract content without aid from the file system, provided the targets are known beforehand.

Key (2013)'s File Block Hash Map Analysis (FBHMA) EnScript and Simson Garfinkel's tool frag_find (S. L. Garfinkel, 2009) provided practical implementations that automated the process for forensic examiners, though searches were limited to a few files at a time. S. Garfinkel, Nelson, White, and Roussev (2010) described the implementation and evaluation of *frag_find* in detail, noting a particular difficulty in storing and searching billions of hashes at practical speeds. S. Garfinkel and McCarrin (2014) later succeeded in scanning a drive image for matches of 4096-byte blocks across a set of nearly one million blacklisted files stored in the custom-built database, *hashdb*.

4.4 Concepts

While approximate matching (a.k.a. fuzzy hashing or similarity hashing) started to gain popularity in the field of digital forensics in 2006, it was not until 2014 that the National Institute for Standards and Technologies (NIST) developed standard definitions, publishing *Approximate Matching: Definition and Terminology* (NIST SP 800-168, Breitinger, Guttman, et al. (2014)). Subsections below briefly summarize this work's principles.

The 'Purpose and Scope' section of the NIST document defines approximate matching as follows: "Approximate matching is a promising technology designed to identify similarities between two digital artifacts. It is used to find objects that resemble each other

or to find objects that are contained in another object."

4.4.1 Use cases

In investigative cases, approximate matching is used to filter known-good or known-bad files while using a reference approximate matching hashed data set, either on static data or data in transit over a network. The primary use cases for approximate matching are presented below:

- Similarity detection correlates related documents, e.g., different versions of a Word document.
- Cross correlation correlates documents that share a common object, e.g., a DOC and a PPT document including the same image.
- Embedded object detection identifies an object inside a document, e.g., an image inside a memory dump.
- Fragment detection identifies the presence of traces/fragments of a known artifact, e.g., identify the presence of a file in a network stream based on individual packets.

A lecture at DFRWS USA 2015 decided to break down uses into 6 categories instead, from the perspective of the bytestreams matching: R is identical to T, R contains T, R & T share identical substrings, R is similar to T, R approximately contains T, and R & T share similar substrings (where R and T are two sequences) (Ren & Cheng, 2015).

Notwithstanding, approximate matching may not be appropriate when used to whitelist artifacts since such content can be quite similar to benign content, e.g., an SSH server with a backdoor would look analogous to a regular entry point (Baier, 2015).

4.4.2 Types

Regardless of the use cases, approximate matching can be implemented at different abstractions. Usually we distinguish between the following three abstraction categories:

Bytewise: matching operates on the byte level and uses only the byte sequences as input only (also known as *fuzzy hashing* and *similarity hashing*).

Syntactic: matching also works on the byte level but may use internal structure information, e.g., one may ignore the TCP header information of a packet that is parsed.

Semantic: matching works on the content-visual layer and therefore closely resembles human behavior (also called *perceptual hashing* and *robust hashing*), e.g., the similarity of the content of a JPG and a PNG image where the image file types / byte streams are different, but the picture is the same.

Furthermore, there are 4 cardinal categories of algorithms (see Sec. 4.5 for the inner workings) (Martínez et al., 2014):

- Context-Triggered Piecewise Hashing (CTPH).

- Block-Based hashing (BBH).
- Statistically-Improbable Features (SIF).
- Block-Based Rebuilding (BBR).

4.4.3 Requirements

There are multiple ways to interpret substring matching. For example, "ababa" and "cdcdc" might be considered similar because they both have five characters ranging over two alternating values, or they might be treated as dissimilar because they have no common characters. Thus, algorithms must define the lowest common denominator of its interpretation - a *feature* - including how they are derived from an input. When two features are compared the outcome is binary, match or no match.

A *feature set* refers to a set of distinct features found in the entire bytestream of a file or file fragment. An algorithm may choose to only include some features in this set and must outline the method/criteria for inclusion. Feature sets are then used to produce a match/similarity score, representing the amount of similarity between sets of target files rationally (an increasing monotonic function).

Bytewise algorithms have two main functions. A *feature extraction function* identifies and extracts features from objects to convert them to a compressed version for comparison. Then, a *similarity function* performs the comparison between these compressed versions to output a normalized match score. This comparison usually involves string formulas such

as *Hamming distance* and *Levenshtein distance*; Martínez, Álvarez, Encinas, and Ávila (2015) and Li et al. (2015) have proposed new algorithms for specific uses. Normalized scores may be calculated by weighing the number of matching features against the total number of features for both objects (for resemblance), or by ignoring unmatched features in the container object (if concerned with containment).

In addition to the above, these traits must be satisfied to be considered a valid approximate matching algorithm, according to NIST:

Compression: actual storage of features is usually implemented as a one-way hash known as a *similarity digest*, *signature*, or *fingerprint*); length is shorter than the original feature/input itself.

Similarity preservation: similar inputs should result in similar digests.

Self-evaluation: authors should state the confidence level for the circumstances/parameters used to produce the match score and what the scale is (e.g., 0 = no features matched, 1 = all exact match).

Time complexity/runtime efficiency: speed should be stated via theoretical complexity in O-notation as well as the runtime speed; for bytewise algorithms it is preferable to know the isolated speeds of the feature extraction and similarity functions.

4.4.4 Testing bytewise approximate matching

Testing algorithms is an important task, so researchers set out to create a test environment for bytewise approximate matching. The first step was taken by Breitinger, Stivaktakis, and Baier (2013), called FRamework to test Algorithms of Similarity Hashing (FRASH).

It tested *efficiency*, *sensitivity and robustness*, and *precision and recall*. This last category can be divided further into synthetic data vs. real world data. While synthetic data provides the perfect ground truth (further described below), it does not coincide with the real world, and vise versa.

Synthetic data test results were published (Breitinger, Stivaktakis, & Roussev, 2013) in addition to real world data (Breitinger & Roussev, 2014). The complete results are too complex to be presented in this article but can be found in chapter 6 in Breitinger (2014). In the following subsections we briefly summarize how approximate matching algorithms can be evaluated and FRASH's results. The main findings were:

- `sdhash` and `mrsh-v2` outperform other algorithms.
- `mrsh-v2` is faster and shows better compression than `sdhash`.
- `sdhash` obtains slightly better precision and recall rates than `mrsh-v2`.

Therefore, the final decision for selecting an algorithm depends on the use case.

4.4.4.1 Efficiency.

As with cryptographic hash functions, compression and runtime efficiency are important, but approximate matching algorithms involve additional concerns; several do not output fixed length digests. Thus, researchers usually report compression ratio, $cr = \frac{\text{digest length}}{\text{input length}}$.

The community distinguishes between the following for runtime:

Generation efficiency: time needed to process an input and output the similarity digest.

Comparison efficiency: summarizes the theoretical complexity (in O-notation) to compare digests against an existing data set / database; again, often stated in units of time for implementations for bytewise approximate matching.

Space efficiency: calculated by dividing digest length by input length.

4.4.4.2 Sensitivity and robustness.

Sensitivity refers to the granularity at which an algorithm can detect similarity, i.e., how minute the feature is. At some threshold making a feature too fine causes almost all objects to appear common, however, and therefore the algorithm designer must strike a balanced sensitivity to optimize utility and time efficiency.

Robustness is a metric of how effective an algorithm can be in the midst of noise and plain transformations such as fragmentation and insertion/deletion into the target byte sequence.

As outlined by FRASH, these attributes are tested by creating manual mutations of the target fragments/files:

Alignment robustness: inserts blocks of various sizes at the beginning of an input; this should simulate scenarios like growing log files or emails.

Fragment detection: identifies the smallest fragment of a byte sequence that still matches by cutting it; this feature is important for network traffic analysis (see Sec. 4.6.2) and hash-based file carving (see Sec. 4.6.1).

Single-common block correlation: analyzes the minimum amount of correlation between two files, e.g., two word documents that share an common paragraph.

White noise resistance: is a probability-driven test that introduces (uniform) random changes into a byte sequence (via insertion, deletion, & substitution); a viable scenario is source code where a developer renamed a variable.

4.4.4.3 Precision and recall on synthetic data.

Precision can be thought of as a measure of false positives (possibility of counting objects as similar that in actuality are not) while recall refers to the false negatives (omitting objects that should be tallied as similar). These attributes are an indication of an algorithm's reliability.

In order to quantify them, the initial step is to analyze synthetic data. First, random byte sequences (the FRASH paper used Linux `/dev/urandom`) are generated. Next, mutations

are created through methods like those mentioned in the previous subsection. Finally, the comparison is executed and the results are analyzed.

4.4.4.4 Precision and recall on real world data.

Testing on real world data is a bit more complex because there is no definition for similarity and no ground truth (publicly available data sets for testing that involves explanations of what the expected similarity is between different files). To define the ground truth, the community developed *approximate longest common substring* (aLCS) which estimates the longest common substrings of two files. According to this, two inputs are declared as similar, if their aLCS is sufficient (e.g., 1 % of the total input length, or at least 2 KiB).

4.4.5 Security

Most approximate matching algorithms currently implement few-to-zero security features to guard against active, real-time attacks. One staple is inherently built into approximate matching algorithms: hash functions are one-way, even for similarity, and therefore prevent reverse engineering the original input sequence of a fragment / file. The few other tolerances exhibited by the algorithms are stated in their individual subsections under Sec. 4.5.

However, we posit that for most uses of approximate matching, security features are not essential. As pointed out by Baier (2015) these algorithms are most likely to be used for blacklisting. Why would an active adversary want to create files that match a blacklist

of static (not in transit) data? Researchers must find an answer to how easy it is to avoid matching files. Maybe in the future we should classify security for approximate matching algorithms by the minimum amount of changes that are necessary between two files in order to produce a non-match. A question that needs fresh exploration, though, is what practices criminals can use to bypass certain (types of) algorithms, use cases, and applications; a rigorous analysis of this has not been performed partially due to missing standards / ground truth.

4.4.6 Extending existing concepts

One of the major challenges that comes with approximate matching is related to the nearest neighbor problem, i.e., how to identify the similarity digests that are similar to a given one. More precisely, let's assume a database containing n entries. Most algorithms require an 'against-all' comparison which equals a complexity of $O(n)$.

Winter, Schneider, and Yannikos (2013) presented an approach to diminish this complexity for `ssdeep` named F2S2. Generally speaking, instead of storing the complete Base64 encoded similarity digest in the database, they stored n-grams using hash-tables. In order to lookup single digests they first looked for the n-grams which reduced the overall amount of comparisons. For the final decision, the `ssdeep` comparison function was used. As a result, they reduced the comparison time of 195,186 files against a database containing 8,334,077 records from 442 h to 13 min (boosted by a factor of about 2000), a 'practical speed'.

However, this approach works only for Base64 and hence for none of the other approaches like `sdhash` or `mrsh-v2`. Therefore, Breitinger, Baier, and White (2014) presented a concept that could speed up the process via Bloom filter-based approaches. They suggested using one single huge Bloom filter to store all feature hashes, which results in a complexity of $\sim O(1)$. Their approach overcomes the drawback of comparing digests against digests but loses precision. That is, it allows for only yes or no decisions: yes means there is a similar file in the set; no equates to none of the files being similar above the chosen threshold. It does not allow for the returning of the matched file(s).

Consequently, the authors presented an enhancement which simply uses multiple large Bloom filters to generate a tree structure that results in a complexity of $O(\log(n))$ (Breitinger, Rathgeb, & Baier, 2014). But these are only assumptions – while there is a working prototype for the first approach, the latter concept only exists in theory.

4.4.7 Distinction from locality-sensitive hashing (LSH)

It's critical to note that sometimes people confuse *Locality-Sensitive Hashing* (LSH) (e.g., Rajaraman and Ullman (2012)) with approximate matching. Therefore, we included this section. LSH is a general mechanism for nearest neighbor search and data clustering where the performance strongly relies on the used hashing method. Two popular algorithms are `MinHash` (Broder, 1997) and `SimHash`[2] (Charikar, 2002).

[2]Note, SimHash is a common term and is used several times literature. Accordingly, it is also used twice in this article. Besides this section it is also used in Sec. 4.5.6 where it describes an approach from Sadowski and Levin (2007).

This does not necessarily coincide with the idea of approximate matching. Specifically, while LSH aims at mapping similar objects into the same bucket, approximate matching outputs a similarity digest that is comparable.

We would like to note here that the following section mainly focuses on bytewise approximate matching.

4.5 Introduction to algorithms

As previously mentioned, approximate matching started to gain attention in 2006 with the concept of context triggered piecewise hashing and its first implementation, `ssdeep` (Kornblum, 2006). In the following years, new algorithms were proposed and published.

We will introduce the eight known approximate matching algorithms. While the first three algorithms are still extended and relevant, the last four algorithms are less promising from a digital forensics perspective for various reasons, e.g., precision and recall rates, runtime efficiency and detection capabilities. The last algorithm (TLSH) is more related to LSH than approximate matching and is included for completeness.

This section is a high-level summary of the current algorithms. Throughout each subsection references are cited for deeper reading.

4.5.1 ssdeep

CTPH is the technique used by `ssdeep` and was presented by Kornblum (2006). Roughly speaking, it is a modified version of the spam detection algorithm from Tridgell (2002–2009) generalized to cope with any digital object.

In CTPH the approach is to identify trigger points to divide a given input into chunks/blocks. This breakup is performed using a rolling hash that slides through the input, adds bytes to the current context (think of it as a buffer), creates a pseudo-random value, and removes them from the context after a set number of bytes are completed. The context is then used as a trigger – whenever a specified sequence is created the current context is hashed by the non-cryptographic FNV-hash function (Fowler, Noll, & Vo, 1994–2012). To create the similarity digest, the FNV-chunk-hashes are reduced to 6 bits, converted into a Base64 character and concatenated; this is done continuously as the trigger outputs FNV hashes.

At the time of this article, `ssdeep` was still an active project with version 2.13 and is freely available online[3]. Over the years, several extensions and performance improvements have been published that mostly focus on the efficiency of the implementation (Chen & Wang, 2008; Seo, Lim, Choi, Chang, & Lee, 2009; Breitinger & Baier, 2012b). However, a security analysis conducted by Baier and Breitinger (2011) showed that CTPH cannot withstand an active attack.

[3] `http://ssdeep.sourceforge.net` (last accessed 2017-05-02).

4.5.2 sdhash

Similarity digest hashing was published four years later by Roussev, Richard, and Marziale (2008); Roussev (2010) and is also still active. The SIF algorithm extracts statistically improbable features that are determined by Shannon entropy (not the ones with the highest / lowest entropy but the ones that seem unique, Roussev (2009)). In `sdhash`, a feature is a byte sequence of 64 bytes that is then compressed by hashing it with SHA-1. Finally, the author developed a way to insert the hashes into a Bloom filter[4] (Bloom, 1970).

The original version was extended several times, now supporting GPU usage for calculation and a block-based hashing mode (Roussev, 2012). The current version (3.4) is available online[5].

A comparison between `ssdeep` and `sdhash` showed that the latter algorithm outperforms its predecessor (Roussev, 2011). In addition, a security analysis showed that `sdhash` is much more robust and difficult to overcome (Breitinger & Baier, 2012c).

[4] A Bloom filter is a space efficient data structure to represent a set. Bloom filters will not be discussed in this article but more details can be found online.

[5] `http://sdhash.org` (last accessed 2017-05-02).

4.5.3 mrsh-v2

This algorithm was published by Breitinger and Baier (2013) and is a combination of `ssdeep` and `sdhash`[6]. Like the aforementioned implementations, `mrsh-v2` is still supported[7]. The algorithm uses the feature identification procedure from `ssdeep`, then hashes the feature using the non-cryptographic FNV (Fowler et al., 1994–2012) and proceeds like `sdhash`, consequently overcoming the weaknesses of `ssdeep` and becoming faster than `sdhash`. The precision and recall rates are slightly worse than `sdhash`.

4.5.4 bbHash

Building block hashing is a completely different approach and is based on the concept of eigenfaces (biometrics) and de-duplication (data compression). Contrary to expectation, its type (see Sec. 4.4.2) is not eponymous but rather BBR. The main difference is that this approach utilizes an external reference point – the building blocks.

A set of 16 building blocks (random byte sequences) is used to optimize representation of a given file. In order to find this representation the algorithm calculates the *Hamming distance*, which is time consuming and slow for practical usage (e.g., it takes about two minutes to process a 10 MB file) (Breitinger & Baier, 2012a).

[6]It was also inspired by multi-resolution similarity hashing (Roussev, III, & Marziale, 2007).

[7]`http://www.fbreitinger.de` (last accessed 2017-05-02).

4.5.5 mvHash-B

Majority vote hashing, another BBR type, was published by Åstebøl (2012); Breitinger, Åstebøl, Baier, and Busch (2013). It transforms any byte-sequence into long runs of `0x00`s and `0xFF`s by considering the neighboring bytes of a specific byte. If the neighborhood consists of mainly 1s, the byte is set to `0xFF`, otherwise to `0x00`. Next, these runs are encoded by Run Length Encoding (RLE). Although this proceeding is very fast, it requires a specific configuration for each file type.

4.5.6 SimHash

`SimHash` was presented by Sadowski and Levin (2007) and embodies the notion of counting the occurrences of certain predefined binary strings called "Tags" within an input. In their BBR implementation, the authors used 16 8-bit Tags, i.e., a possible Tag could have been 00110101. Subsequently, the tool parses an input bit by bit, searching for each Tag. The total number of matches is stored in a *sum table*. A hash key is computed as a function of the sum table entries that form linear combinations. Lastly, all information (including file name, path, and size) is stored in a database.

To identify similarities, a second tool named `SimFash` is used to query the database. The hash keys are used as a first filter to identify all possible matches. Next, the sum tables are compared and a match is found if the distance is within a specified tolerance.

The authors clearly state that "two files are similar if only a small percentage of their raw bit patterns are different. ... [Thus,] the focus of SimHash has been on resemblance detection" (Sadowski & Levin, 2007).

4.5.7 saHash

Another SIF type, `saHash` uses Levenshtein distance to derive similarity between two byte sequences. The output is a lower bound for the Levenshtein distance between two inputs. Akin to `SimHash` (Sec. 4.5.6), `saHash` allows for the detection of only near duplicates (up to several hundred Levensthein operations).

A unique characteristic of this approach is its definition of similarity. While all other approaches output a number between 0 and 1 (not a percentage value), `saHash` actually returns the lower bound of Levenshtein operations (Ziroff, 2012; Breitinger, Ziroff, Lange, & Baier, 2014) to convert one file into another.

4.5.8 TLSH

TLSH belongs to the category of locality-sensitive hashes, published by Oliver, Cheng, and Chen (2013), and is open source[8]. It processes an input byte sequence using a sliding window to populate an array of bucket counts, and determines the quartile points of the bucket counts. A fixed length digest is constructed which consists of two parts: (i)

[8] `https://github.com/trendmicro/tlsh` (last accessed 2017-05-02).

a header based on the quartile points, the length of the input, and a checksum; (ii) a body consisting of a sequence of bit pairs, which depends on each buckets value in relation to the quartile points. The distance between two digest headers is determined by the difference in file lengths and quartile ratios. Meanwhile, the bodies are contrasted via their approximate Hamming distance. Summing these together produces the TLSH similarity score.

According to the authors, the precision and recall rates are robust across a range of file types. Additional experiments (Oliver, Forman, and Cheng (2014)) showed that TLSH can detect strings which have been manipulated with adversarial intentions[9]. TLSH is also effective in detecting embedded objects depending on the level of object manipulation. Despite these advantages, it is less powerful than `sdhash` and `mrsh-v2` for cross correlation.

4.6 Applications

Originally, approximate matching was designed to support the digital investigation process via the use cases stated in Sec. 4.4.1; search for target file(s)/fragments or reduce the volume of data needing investigation. Recently, tools such as EnCase, X-Ways Forensics, and Forensic Toolkit (FTK) have incorporated similar object detection technologies (Breitinger, 2014). Researchers have now identified additional working areas where these

[9]Tolerance of manipulation was one of the design considerations for TLSH.

techniques or tools can have practical impact, e.g., for file carving (see Sec. 4.6.1), data leakage prevention (see Sec. 4.6.2 and 4.6.4) and Iris recognition (see Sec. 4.6.5).

4.6.1 Automatic data reduction and hash-based file carving

As sifting through data has become cumbersome, pre-processing schemes have risen. Extracting data in bulk is arguably the most sought after application of approximate matching. One perspective that should be fruitful is hash-based file carving.

This alienated area of work was presented by S. Garfinkel and McCarrin (2014). In their paper, the authors combined techniques from file carving and approximate matching to search on "media for complete files and file fragments with sector hashing and hashdb". Instead of focusing on the complete file and comparing it against a database, the authors use individual data blocks. They utilized a special database named `hashdb` (Allen, 2015) to obtain high throughput.

The evaluation proved their strategy works, although they had to solve the problem of non-probative blocks that emerged "from common data structures in office documents and multimedia files". To filter out such artifacts, the authors presented several 'tests' that alleviated the problem.

4.6.2 Network traffic analysis

V. Gupta (2013); Breitinger and Baggili (2014) demonstrated preliminary results when using approximate matching on network traffic for data leakage prevention. The question was (since approximate matching can be used for fragment detection) whether network packets could be matched back to their original files.

Design was similar to its traditional counterpart: create a database of known-object signatures (most likely files) and identify these objects, but instead of analyzing a hard drive the researchers used a network stream (single packets). This work illustrated approximate matching's utility in data leakage prevention, a formerly untouched application.

Beginning with modifying the original `mrsh-v2` algorithm to handle the small size of 1460 bytes per packet, the authors showed that this method works robustly on random data (true positive rate 99.6 %, true negative rate 100.0 %) having a throughput of 650 Mbit/s on a regular workstation.

Regardless, they faced several unsolved problems for real world data. One obstacle was that many files share the same structural information (e.g., file header information; this is equivalent to the non-probative blocks problem from the previous subsection) which led to false positive rates of around 10^{-5} – too high for network traffic analysis.

4.6.3 Malware

Innately, similarity hashing is ideal for grouping things together, but it was not until 2015 that it was rigorously tested when applied to malware clustering (Li et al., 2015). Faruki, Laxmi, Bharmal, Gaur, and Ganmoor (2015) developed AndroSimilar, a syntactical detection algorithm for Android Malware that falls into the SIF category. While Zhou, Zhou, Jiang, and Ning (2012)'s DroidMoSS, a CTPH algorithm, was developed to also detect mobile malware, a comparison between the two could not be performed due to unavailable code.

Polymorphic malware families are on the rise, often hosted on servers that automatically alter inconsequential segments of a file (before sending across a network) to bypass cryptographic detection tactics (Security, 2013). An intriguing paper by Payer et al. (2014) expands on ways criminals can circumvent the use of similarity-based matching for spotting malicious binary code, which often diversifies itself during recompilation. Thus, it is imperative that malware receives more attention.

4.6.4 Data leakage prevention

With respect to data leakage prevention, approximate matching may also be utilized for printer data inspection, e.g., MyDLP[10] and Symantec (2010) Data Leakage Prevention. If a document is protected, the software can discard the print (implemented by MyDLP).

[10] `https://www.mydlp.com` (last accessed 2017-05-02).

A similar experiment was also run by our research group[11] which created a virtual secure printer that analyzed a sent document before forwarding it to an actual printer.

Note, according to Comodo Group Inc. (2013), these software solutions often call their technology partial document matching, unstructured data matching, intelligent content matching, or statistical document matching, all synonyms for approximate matching.

4.6.5 Biometrics

Biometrics is another independent domain employing approximate matching with promising results (Rathgeb, Breitinger, & Busch, 2013; Rathgeb, Breitinger, Busch, & Baier, 2013). In their work, the authors demonstrated the feasibility of using techniques from approximate matching for biometric template protection, data compression and efficient identification. According to Breitinger (2014), there are three improvements:

- "Template protection: the successive mapping of parts of a binary biometric template to Bloom filters represents an irreversible transformation achieving alignment-free protected biometric templates."
- "Biometric data compression: the proposed Bloom filter-based transformation can be parameterized to obtain a desired template size, operating a trade-off between compression and biometric performance."

[11] The project was done by Kyle Anthony, a member of the UNH Cyber Forensics Research & Education Group

- "Efficient identification: a compact alignment-free representation of iris-codes enables a computationally efficient biometric identification reducing the overall response time of the system."

4.7 Limitations and challenges

Bytewise approximate matching has some intrinsic limitations. First and foremost, it cannot pick up similarity at a higher level of abstraction, such as semantically. For instance, it cannot meaningfully match two image files that have the same semantic picture but are different file types / formats due to different binary encoding. However, placing it in tandem with other approaches will still help; Neuner, Mulazzani, Schrittwieser, and Weippl (2015) include it as a critical component of the digital forensic process. Doubly crucial when it comes to applications like malware that employ databases is lookup time. Winter et al. (2013) outlined a faster approach to conduct similarity searching using a database but this method will not work effectively for all approximate matching algorithms. We will not belabor the point since this was discussed in Sec. 4.4.6.

Evidently, the first challenge to confront is awareness and adoption of approximate matching, a possible indication that more research needs to be conducted to understand the needs for the community. As we outlined in the introduction, 15 % of professionals are unaware of approximate matching – an unacceptable number. Conversely, 7 % criticize that algorithms are too slow for practical use. On the other hand, inquiry needs to be made into why 35 % have used approximate matching only a few times. Our hope is that

having the NIST SP 800-168 definition, a technical classification of algorithms (Martínez et al., 2014), and this more universal outline will improve awareness and adoption.

Yet another major obstacle is the lack of a standard definition of similarity (Baier, 2015). As addressed by S. Garfinkel and McCarrin (2014), not all kinds of byte level similarity are equally valuable as there are some artifacts (e.g., structural information, headers, footers, etc.) that are less important or lead to false positives. Hence, we need a filtering mechanism to prioritize matches. One possibility could be to extract the main elements (like text or images) and compare those, meaning including a pre-processing step before the comparison.

Aside from initial efforts to test approximate matching algorithms, there are currently no accepted standards and reliable testing frameworks. FRASH is not easy to implement, a deterrent to practitioners. This is one reason this paper avoids giving absolute comparisons between all the (types of) algorithms. More bothersome is the lack of an accepted ground truth for real world data that would support implementation assessments like whether algorithms scale effectively. The ground truth should embody the four use cases (see Sec. 4.4.1) and algorithm types, even if different data sets are required for each one. Once this is done, the algorithms will be directly comparable (e.g., embedded object detection: A better than B better than C; speed: B faster than A faster than C). We posit that it is critical for practitioner efficiency to know which algorithms solve which potential problems. At the moment the National Software Reference Library (NSRL[12])

[12] `http://www.nsrl.nist.gov` (last accessed 2017-05-02).

has built the most prominent software database and is piecing together a test corpus (its usefulness was demonstrated in Rowe (2012)).

4.8 Future focus

Future research should, in accompaniment to prior comments, pursue areas that branch out from digital forensics, even if completely detached. Below we name some possible applications that could use enhancement, and areas that approximate matching may be able to be enhanced by (this is not an exhaustive list):

- Bioinformatics: This field already uses exact matching methods, albeit using bytewise or semantic approximate matching alongside today's methods could conceivably increase efficiency.
- Text mining: Identifying patterns in structured data to gain high-quality, semantic value.
- Templates and layouts: Semantically identify document layout and separate content from template automatically.
- Deep / machine learning: Automated forms of learning need to be able to process information fast, store it efficiently space-wise, and be able to differentiate similarities and differences; semantic hashing is a known aspect of deep learning and ergo might be able to strengthen approximate matching.

- Source code governance: Manage shared code better, especially for open source software.

- Spam filtering and anti-plagiarism: These have already been looked at but might be behooved by deeper scrutiny.

Ultimately, approximate matching is an alienated domain and its increased adoption will strongly support other domains. Researchers looking to improve on the slowness of indexing and searching may also benefit to look into other domains such as compression and programming. Once more people are aware of approximate matching we might identify more fields where the technology would be relevant.

4.9 Acknowledgments

We would like to thank Michael McCarrin for reviewing and editing some paragraphs, especially those that are related to his research. Additionally, we would like to thank Jonathan Olivier for providing us with the summary for TLSH.

Chapter 5

Usage of approximate matching databases for document similarity lookup and template detection

5.1 Note about authorship

Note that most of the Appendix was done by Dr. Frank Breitinger, including the `Python-MySQL` development and testing; only writing and some theoretical calculations were edited. The `Python-MySQL` version is not the focus of this chapter but the Appendix has been included for more detailed understanding of rationale behind design decisions and the approximate matching approach itself. Dr. Breitinger was also responsible for

the non-template literature review and the initial template-content analysis presented in this paper.

5.2 Abstract

Traditional approximate matching approaches suffer mainly from three roadblocks: 1) lookup times are slow, 2) all features are usually weighted the same (this is a problem because in many cases bytestream padding is not of interest to investigators), and 3) similarity scores have varying margins of error and interpretation (between different algorithms as well as different use cases and data sets). To sidestep these we propose using databases for approximate matching applications. In this paper we present `SimFind`, a `C++-MongoDB` implementation that can be used to increase recall speeds and more clearly define what similarity means, not only through the use of extra fields/metadata stored in the database but possibly through template-content detection (thus allowing weightings for features to be dictated and extracting some of the ambiguity from similarity scores).

5.3 Introduction

One of the main practical challenges in digital forensics today is coping with massive amounts of data. Paralleled with the increase in the amount of data is the exponential increase in cybercrimes and digital evidence. Consequently, most (federal) agencies

have computer investigation backlogs often resulting in processing delays of 8-12 months (Baggili, Marrington, & Jafar, 2014).

In order to speed up the investigation process it becomes necessary to *automatically* reduce the amount of data for each case. This is commonly done by utilizing cryptographic hash functions for *known file filtering*, where a tool automatically hashes all files from a seized device and compares them against a reference database. A widely used database is the National Software Reference Library (NSRL)[1], (Fisher, 2002; NSRL, 2013).

However, cryptographic hashes come with a major downside. Due to security requirements, they allow practitioners to identify exact duplicates only – even a single bit change results in a completely new fingerprint. Hence, researchers developed a counterpart called approximate matching (also known as fuzzy hashing or similarity hashing) that also allows users to correlate similar files. According to the definition of the National Institute for Standards and Technology (NIST), "approximate matching is a promising technology designed to identify similarities between two digital artifacts" (Breitinger, Guttman, et al., 2014).

Regardless of the algorithm there are currently several major problems with existing approaches (ordered from most to least critical):

[1] "The National Software Reference Library (NSRL) is designed to collect software from various sources and incorporate file profiles computed from this software into a Reference Data Set (RDS) of information." `http://www.nsrl.nist.gov` (last accessed 2017-05-02)

Lookup performance. If $|S|$ is the amount of files of a set S stored in a database, approximate matching requires an 'against-all' comparison to look up a single digest (complexity of $O(|S|)$).

Priority of similarity. Recent bytewise implementations rate all features equally although there can be vast differences in their quality. For instance, similarities based on header, footer, structural information, or long runs of zeros should be weighted less while unique features should have higher weight. Put simply, two *Word* documents not having a single word in common might be less relevant for practitioners.

Similarity score. According NIST's definition, "the score represents a normalized estimate of the number of matching features in the feature sets *corresponding to the [objects] from which the similarity digests were created*" (Breitinger, Guttman, et al., 2014). Thus, it is an evaluation benchmark that differs case-to-case (varying margins of error) and is, as some authors describe it, more equivalent to a certainty score. There may be scenarios that require a more precise statement, e.g., a minimum common length of 2 KiB.

In this paper we present `SimFind`, an approach for identifying similarities among digital objects that illustrates a method to alleviate the existing problems of database lookup performance and scalability, by offering varied levels of similarity searching, indexing of features, user-specified threshold, and a database design for speedy queries. Additionally, we explored using metadata to separate structural / template information from content,

thereby mitigating the similarity priority problem. Source code is available to the public (https://github.com/vikhari/sim-find).

`SimFind` has an overall complexity between $\omega \cdot O(1)$ and $\omega \cdot O(\log_2 |B|)$ where ω is equal to the amount of features of an object and $|B|$ is the number of objects containing a given feature. Tests produced a 30.99 minute query time on average for a 1.78Gb corpus to query itself.

The remainder of this chapter is organized as follows: Background and related work provides the context for relevant approximate matching and details different kinds of work that has been done on separating template from content. Then, Sec. 5.5 outlines the design decisions made and the query options available (the first `MySQL-Python` prototype is further indulged in the Appendix – this is prior work which explains some aspects in more detail but is not essential to the crux of this paper). After giving test set information Sec. 5.6 evaluates `SimFind` (hereafter unless otherwise mentioned `SimFind` refers to the second `C++-MongoDB` prototype). This leads to our exploration of detecting template and content at the byte level in the following section (5.7). Finally, we discuss Limitations, Conclusions, and Future.

5.4 Background and related work

Approximate matching customarily consist of two functions. The *feature extraction function* identifies and extracts features which are then compressed and result in a similarity

digest. The second function is the *similarity function* which allows the comparison of two digests and outputs a similarity score.

Depending on the algorithm, feature extraction operates either on the byte level or the semantic level (more details are given in (Zauner, 2010)). This paper centers on bytewise similarity; two objects are similar if they share common byte sequences. Howbeit, our general approach is flexible and may be adapted for semantic scenarios – most likely only the feature extraction function would need changing.

Over the past years, several different algorithms, tools, comparative papers, and testing software were published in this domain. The upcoming subsections focus on the most promising published algorithms to date. For a brief review of the algorithms see Chapter 4.

5.4.1 `ssdeep` and the F2S2 software

Context triggered piecewise hashing (CTPH) with its implementation, `ssdeep`, was introduced by Kornblum (Kornblum, 2006) and originated from the spam detection algorithm by Tridgell (Tridgell, 2002–2009). This seems to be the most prevalent algorithm in the cybersecurity industry, e.g., `ssdeep` is a file attribute for searching malware on VirusTotal. The workflow for `ssdeep` is simple: split an input into blocks, hash each block independently and concatenate all hashes to a final similarity digest. In order to split an input into blocks, the algorithm identifies trigger points utilizing a pseudo random function called a rolling hash which considers the current context of seven bytes. Next, all blocks

(approximately 64) are hashed by the non-cryptographic hash function *FNV* (Fowler et al., 1994–2012). For additional compression, CTPH only takes the least significant 6 bits (one Base64 character).

Researchers identified ways to improve the algorithm with respect to both efficiency and security (Chen & Wang, 2008; Seo et al., 2009). One of the most important improvements was presented in 2013 by Winter et al. They developed a solution for the lookup complexity problem called F2S2 (Winter et al., 2013). The authors showed an improvement of a factor of almost 2000 which is now 'practical speed'. In their example, they decreased the time for verifying 195,186 files against a database with 8,334,077 entries from 364 h to 13 min.

5.4.2 Approximate matching approaches based on Bloom filters

Two newer approaches for approximate matching that outperform `ssdeep` with respect to precision and recall are `sdhash` and `mrsh-v2`. In contrast to CTPH, these approaches use Bloom filters (Bloom, 1970) for the similarity digest representation instead of Base64 encoding and thus F2S2 cannot be used.

5.4.2.1 Bloom filters

A Bloom filter is commonly used to represent elements of a finite set S. A Bloom filter is an array of v bits initially all set to zero. In order to 'insert' an element $s \in S$ into the filter, k independent hash functions are needed where each hash function h outputs

a value between 0 and $v-1$. Next, s is hashed by all k hash functions h. To insert, the bits at the positions $h_0(s), h_1(s), \ldots h_{k-1}(s)$ of the Bloom filter are set to one.

To answer the question if s' is in S, we compute $h_0(s'), h_1(s'), \ldots h_{k-1}(s')$ and analyze if the bits at the corresponding positions in the Bloom filter are set to one. If this holds, s' is assumed to be in S, however, we may be wrong as the bits may have been set to one by different elements from S. Hence, Bloom filters suffer from a non-trivial false positive rate. Otherwise, if at least one bit is set to zero, we know with certainty that $s' \notin S$. It is obvious that the false negative rate is equal to zero.

Due to the peculiarities of Bloom filters, they can only store a specific amount of features as the false positive will be too high otherwise. Therefore, both implementations have a maximum of features per Bloom filter. If this limit is reached, a new Bloom filter is created. As a consequence, the final similarity digest for a file is not a single Bloom filter but a sequence of them. To identify the similarity between two digests, all Bloom filters of fingerprint A are compared against all Bloom filters of similarity digest B which is performed by measuring the Hamming distance. Ultimately, bloom filters address the issue of space effectively while raising other issues, further described below.

5.4.2.2 `sdhash`

This algorithm was proposed by Roussev (Roussev, 2010) and attempts to pick characteristic features with a length of 64 bytes. Each feature is unlikely to appear by chance

in other objects, supported by empirical study. The baseline implementation hashes the features with *SHA-1* (Gallagher & Director, 1995) and inserts them into a Bloom filter.

5.4.2.3 `mrsh-v2`

Breitinger and Baier (Breitinger & Baier, 2013) published a newer algorithm that is based on `ssdeep` (Kornblum, 2006) and multi-resolution similarity hashing (Roussev et al., 2007). Like `ssdeep`, the algorithm divides an input into blocks using a rolling hash and hashes each block using *FNV*. Instead of building a Base64 similarity digest, `mrsh-v2` inserts these block hashes into a Bloom filter (identical to `sdhash`).

5.4.2.4 Lookup problem

Breitinger et al. discussed this issue and presented two ideas to overcome this problem. In (Breitinger, Baier, & White, 2014) the authors reduced the complexity of a query to $\omega \cdot O(1)$ (ω is the amount of features of an object). The downside is that only set-queries are possible. In other words, it can only answer the question if a similar object is in the underlying dataset by *yes* or *no* – it cannot return the matched objects(s). Their latest improvement (Breitinger, Rathgeb, & Baier, 2014) shows a theoretical solution having a complexity of $\omega \cdot O(\log_2 |S|)$. The two downsides with this approach are that it requires a copious amount of RAM, e.g., a dataset of 256 GB needs approximately 50 GB.

5.4.3 Separating template from content

There has been research on separating structural information, hereafter referred to as template, from file content, but none appear to have done so bytewise using approximate matching and without the use of file-specific information. We define "template" as a ubiquitous layout for a set of files (e.g., a company may use a template document with their logo or name on each page), in addition to the typical file-specific header / structural significance (e.g., Word document or E-mail). Most research in the domains of cybersecurity and cyber forensics focuses on problems concerning the latter. However, effective solutions for complex problems often involve assembling cross-domain approaches. Therefore, this section will review literature attempting to address the overall template-content objective regardless of application or definition, as well as give information about related topics. We hope this also provides the reader with contextual information be the case he/she wishes to combine our bytewise database approach with other techniques for future work.

A primary operation of approximate matching has been identification of text-based similarity. Bär, Biemann, Gurevych, and Zesch (2012) procured a relatively comprehensive review of matching methods: longest common substring, character / word n-grams, semantic graphs, use of lexical semantic resources (e.g., Wiktionary), textual entailment, correlations based on a thesaurus, lexical substitution, statistical translation, and structure / style identifiers (i.e., parts of speech, word order, frequencies, etcetera). Figure 1 of Potthast, Barrón-Cedeño, Stein, and Rosso (2011) exhibited a taxonomy for text

plagiarism types and approaches. Of course, the most relevant angles for our work is hash-based techniques. Retrieval using locality-sensitive hashing, fuzzy-fingerprinting, and hash-based search have been explored as early as 2007 (Stein, 2007). Using these techniques side-by-side with template separation methods would help make them more effective.

Second is the layout similarity track. Inclusion of semantic layout history is pertinent because a visually / semantically consistent feature can be categorized as a template. Obviously image similarity is a hot topic in the Information Age, with Google's machine-learning Vision API and Facebook's (now openly available) deep-learning DeepMask, SharpMask, and MultiPathNet only scratching the surface of the most notable. But our focus was on document similarity and use of approaches similar to approximate matching. Indeed, older work done by Lucent Technologies Bell Labs attempted to combine edit distance with Optical Character Recognition (OCR) and clustering to achieve the goal (Hu, Kashi, & Wilfong, 2000). Since then a myriad of ways to solve aspects of the problem have been scrutinized. An Italian group used modified X-Y trees to represent layout for faster retrieval (Cesarini, Marinai, & Soda, 2002). Chao and Fan (2004) came up with a manner to extract PDF style and logical attributes, expressing them as entries in an XML file. There was a similar invention proposed to deconstruct documents into sets of related glyphs (words, paragraphs, graphs, etc.) and then display user-contingent relationships and clustering (Rosiello, Kirda, Ferrandi, et al., 2007). Despite a similarity approach for detecting website phishing not being concretely linked to our problem, the use of graph properties to define template could also be useful for combinatorial future work.

As declared above image similarity was not our primary concern. On the other hand, usage of such techniques can certainly help to identify duplicate graphics with less error, thereby improving template detection under the prominence definition. Thus, we would like to suggest these works as starting points for ideas: Forsyth and Fleck (1999), Stauffer and Grimson (2001), Deselaers and Ferrari (2011), and Dekel, Oron, Rubinstein, Avidan, and Freeman (2015) (possibly most comparable to approximate matching). Likewise, clustering is another method proposed over a decade ago that could be applied (Steinebach, 2012). Places to begin include an ontology-driven capability for reducing features from the database to a "core" set, and a publishing on correlation preserving indexing (CPI) that maps document components to space before calculating distance (Fodeh, Punch, & Tan, 2011; Zhang, Tang, Fang, & Xiang, 2012).

Alas, we have not found an approximate matching bytewise approach. We hope, a bytewise approach could perhaps funnel its results into other approaches to pan out more refined feature classifications and achieve better recall times, or be used in parallel. Potentially, the work by Aleman-Meza, Halaschek-Wiener, Sahoo, Sheth, and Arpinar (2005), that ropes in different knowledge base data using the Semantic Discovery project, is the nearest example to how a multi-faceted template approach could behoove national security / digital forensics.

5.5 Design decisions

In this section we describe our approach which starts by providing a general overview. Next, we present the database structure (in Sec. 5.5.2) and how to detect object similarity (in Sec. 5.5.3).

5.5.1 Solution overview

From a general perspective, approximate matching has three main steps: (1) feature extraction, (2) feature compression, and (3) feature representation. Hence, the similarity between objects relies on the overlapping of features. `SimFind`'s current feature extraction is that of `mrsh-v2` and explained in Sec. 5.12.1 of the Appendix. The feature compression is done using *SHA-1* (Gallagher & Director, 1995); this was done because more approximate matching algorithms use it than *MD5*. After extracting and compressing the relevant information, everything is committed to a database. The actual similarity between two objects can then be calculated based on database queries. Bear in mind, while (1) and (2) are very similar to existing approaches (see Sec. 5.4), (3) is completely different and the focus of this paper. Also keep in mind the prior mechanisms can be switched for another algorithmic process.

There are two main drawbacks for this approach. First, it requires more storage space (not worrisome today). Second, it needs a database environment (but as mentioned in the literature review this may be useful for other approaches). Alternatively, this approach

adopts all properties of cryptographic hash functions featuring an efficient indexing strategy and few to zero false positive rates/high certainty, depending on the type of query.

5.5.2 Database structure

Initial design of the database was done with a relational SQL schema in mind. A `Python` prototype was implemented with `MySQL`. The `MySQL` database structure for this prototype are found in Sec. 5.12.2 of the Appendix, including implementation details in Sec. 5.12.4 and a discussion of possible enhancements to the database schema in Sec. 5.12.5. These sections can be read for further understanding of pros/cons related to the relational approach but is not necessary. See Fig. 5.1 for the most germane design information.

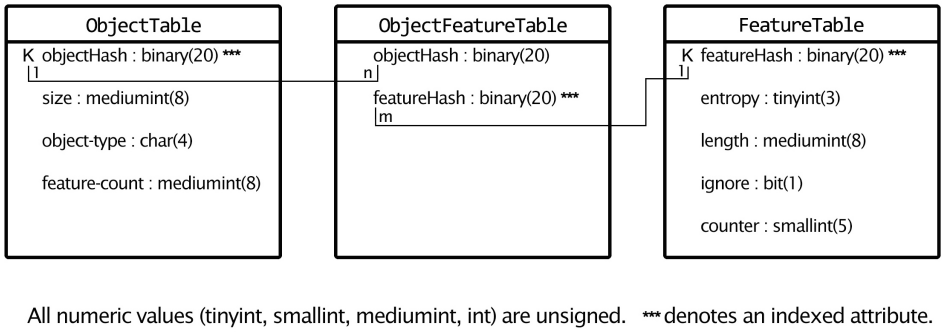

FIGURE 5.1: Basic schema overview.

To attempt to increase speed implementation was shifted to `C++` for creation and `MongoDB` for querying (`Redis`, a faster non-SQL database, was considered but was not flexible enough because changing "business logic" later would be cumbersome[2]). Transferring

[2] `http://stackoverflow.com/questions/5400163/when-to-redis-when-to-mongodb` (last accessed 2017-05-02).

to a document-oriented model allowed for elimination of the intermediary ObjectFeature table; it must be noted that this may not necessarily be faster than a `MySQL` approach due to caching - this should be tested in future work. Rather a feature document contained documents for each object where the feature was present, discarding the need to query a database multiple times per feature. Fields can be seen in the example below (Fig. 5.2) where the cryptographic object hashes (*SHA-1*) serve as the indices for the embedded documents.

```
{
  "_id" : ObjectId("58fa9f731d41c89dae39f614"),
  "featHash" : "1fdd2d3997ec23c67634c4e092fbfd505fd9c7cc",
  "featSize" : 640,
  "featInstances" : 5,
  "4193486b8546aa8149ee1408b46c35354ffc5bbc" : {
    "Name" : "003984.xls",
    "Length" : 15872,
    "NumFeatures" : 19,
    "Count" : 1
  },
  "fe02d7bde4276309018a2d73a18dd844cd6a5da0" : {
    "Name" : "004370.xls",
    "Length" : 427520,
    "NumFeatures" : 1180,
    "Count" : 3
  },
    "e442be63615b72a04054ed20254b659578de8210" : {
    "Name" : "000404.xls",
    "Length" : 58880,
    "NumFeatures" : 234,
    "Count" : 1
  }
}
```

FIGURE 5.2: Example `MongoDB` entry (feature document with embedded object documents).

Indexing of the *featHash* field using hashing was employed to try to optimize querying, and object documents were iterated via their *SHA-1* hashes. Generally speaking, iterating through a list in memory is quicker than obtaining such information from a relational

database because of overhead operations[3], so eliminating the ObjectFeature table was expected to increase speed. Totals of all *Count* fields comprise the *featInstances* field[4] and the *NumFeatures* field signifies the total (non-unique) features in an object's feature set.

The `C++` version calculates entropy (but does not insert it into the database because we wanted to present base runtime without "extras") and does not calculate other attributes that could prioritize similarity. This is another reason a document-oriented database was used. Addition of fields is very easy and does not dramatically increase overhead costs. It should be noted that the effective *hashdb* was not used for this second version because of less ease adding fields, the need to hook in the approximate matching feature creation function, and less inherent stability (the user manual states database integrity can be effected by incorrect commands, for instance) – and accordingly not considered (S. L. Garfinkel & McCarrin, 2015).

5.5.3 Types of queries

With the first (`Python-MySQL`) prototype, to compare a new object against the entire database set there were 3 query options. These are listed in Sec. 5.12.3 of the Appendix. The second version (`C++-MongoDB`) has analogous query options but with more customized

[3] `http://dba.stackexchange.com/questions/76973/what-is-faster-one-big-query-or-many-small-queries,http://stackoverflow.com/questions/1390757/which-is-more-expensive-for-loop-or-database-call` (last accessed 2017-05-02).

[4] Average for this field using the *t5*-corpus was 1.06.

output (screenshots in Sec. 5.12.10). A general query equivalent to the first version is available (1) which gives a yes-or-no answer if there is any feature in the database. Albeit, since the design is slightly different for the second prototype in terms of atomic retrieval (list of objects already retrieved) there is also another option to output all objects in the database that contain the matched feature (2). The third option requires iteration of a query object's entire feature set and will report the percentage that was found. Again, there is another option (4) that will list all related objects for each feature; this also outputs number of instances of the feature per object and the *More-than-ten* tally (Sec. 5.7). The fifth and sixth options take a user-inputted threshold (0-1) which specifies the percentage of query object features (approximately, it is actually $floor(f * n * t)$ as described in the next paragraph) that must be found to be reported. Both options state the number of features that were matched between the query file and the database file (*Feat-count* from Sec. 5.7). Option (5) only reports the first encountered object to meet the threshold while option (6) reports all objects.

The total time (T) of each option for the `MongoDB` version can be represented as follows (for each query object), where q is the time needed to query the database, f is the time needed to query for a single feature, n is the number of features in a query object's feature set, t is the user-specified threshold (decimal), and i is the time needed to iterate through (a potentially varied number of) objects of each feature in memory. Total time is equal to query time plus iteration time ($T = q + i$) and i is zero for options 1-4.

- Option 1/2 minimum: $q = (f)$

- Option 1/2 maximum: $q = (f * n)$
- Option 3/4 min/max: $q = (f * n)$
- Option 5/6 minimum: $q = (f * n * t)$
- Option 5/6 maximum: $q = (f * n)$

To sum up this section and the extra details from the Appendix, collecting different attributes and allowing for different types of queries allows us to define similarity at a finer granularity thereby improving flexibility.

5.6 Database assessments

In the subsequent subsections we will highlight the runtime efficiencies we observed from tests with real-world data and report `SimFind`'s ability to detect similarity accurately, using `sdhash` for comparison.

5.6.1 Test Set

Before putting `SimFind` through experimental trials the theoretical limits should be understood based on the test data and database setup. The results are based on the *t5*-corpus[5], a common set of real-world, various-typed files for digital forensics (Roussev,

[5] `http://roussev.net/t5/` (last accessed 2017-05-02).

2011). It is a subset of `govdocs` which "were obtained by performing searches for words randomly chosen from the Unix dictionary, numbers randomly chosen between 1 and 1 million, and randomized combinations of the two, for documents of specified file types that resided on web servers in the *.gov* domain using the Yahoo and Google search engines" (Corpora, 2013). The method utilized in gathering these documents leads us to assume there is a plethora of similar files in the corpus. It comprises $|S| = 4457$ files, with a total size of $\mu = 1,911,649,699$ bytes ($= 1.78$ GiB). The average file size is approximately 418 KiB. File type distribution is in Table 5.1. From here on filenames mentioned refer to files from this corpus.

TABLE 5.1: Filetype statistics of *t5*-corpus.

jpg	gif	doc	xls	ppt	pdf	text	html
362	69	533	250	368	1073	711	1093

5.6.2 Runtime metrics and size

`SimFind` exposed relatively quick timing – the entire *t5*-corpus (5,748,958 features) was inserted on average (3 runs) in 33.0 min[6] (this included both feature creation and database upload).

A basic experiment averaged 3 runs of the *t5*-corpus querying itself via Option 6, using a threshold of 0.5 (for sample output of the different queries see Sec. 5.12.10 in the

[6]Timings were done on an Ubuntu 16.04 LTS Desktop virtual machine running within VMWare on a Windows 10 computer (VMware allocated 40 GB storage, 8 GB RAM, and 1 processor). Nothing else was running on the host machine at the time. The size of the database after creation was 0.685 GB.

Appendix). This 1.78 GiB query took 30.99 minutes to complete. When scaled up for comparison with a 200 GiB dataset, estimated query time would be 58.03 hours (2.41 days). Be aware that this cannot be juxtaposed to the speed of the `MySQL` prototype estimations as the `MySQL` implementation skipped blocks in bytestreams to speed up the feature creation process, resulting in a different number of features per file in the database.

Database creation time was fast for `sdhash`, creating its flat file database in 60.02 seconds on average with default settings and only input/output specified (an `.sdbf` file is created, storing hashes used for paired comparisons during "querying"). Tweaking the options failed to make the process significantly faster; but this is not troublesome since uploading only happens once and could be done by other distributed machines. When it came to querying the average time of 3 trials for querying was 7.82 minutes (excluding the creation of a duplicate `.sdbf` file used to emulate querying the database with itself). This test did not specify a threshold and thus was an all-against-all comparison. The equivalent query (Option 3/4) and more output `SimFind` finished faster. If extrapolated like before `sdhash` would take 14.64 hours on average, a fourth of that of `SimFind`.

Not extremely important on this occasion, the `sdhash` flat file (64.2 MB) was much smaller than the `MongoDB` database (685 MB). This also may have been effected by the extra 3 metadata fields used (*Length*, *NumFeatures*, and *Count*). This may matter in extremely large datasets but even then is presumably secondary priority to the functionality of this system.

5.6.3 Assessing similarity detection

To demonstrate the enhancements of `SimFind` over traditional approximate matching approaches, we decided to contrast to `sdhash`. The procedure: we performed an all-against-all comparison of the *t5*-corpus and analyzed the 1497 matches outputted by `sdhash` having a score greater-or-equal to 10. This was in contrast to (Roussev, 2011) where the author recommended 21 as a threshold to distinguish between true and false positives. Ignoring that topic, this section discusses classifying the quality of the `sdhash` matches.

5.6.3.1 Existing on the edge

Our first test scenario analyzed the edge cases – non-matches and exact-matches – leading to the following questions:

1. If `sdhash` outputs a match, does `SimFind` also identify it?
2. Does the number of exact-matches coincide between `sdhash` and `SimFind`?

Regarding Question 1: `sdhash` identified 310 matches where `SimFind` returned a 0-score. These matches consisted of 148 different files having scores between 10 and 56. Intriguingly, almost all matches (except 3) were DOC files. We manually investigated 15 randomly chosen pairs and could not find any visible similarity. Hence, we concluded they correlate due to structural information. To validate our assumption, we reduced the

blocksize of `SimFind` to $bs = 64$ (instead of the default 256) to obtain finer granularity. As a result, we determined all 310 matches were based on very short (averaged 415 bytes) and low-entropy (about 1.57) byte sequences. We argue that these sequences are most likely less relevant as they are often structural or filler information.

Regarding Question 2: Out of 14 pairs of completely identical files found in the *t5*-corpus `sdhash`'s similarity score was only 67 for 11 pairs. Ordinarily a score of 100 is posit for full-matches. This can be considered a minor flaw and circumvented easily by comparing cryptographic hashes of files first.

5.6.3.2 Keeping things together

`sdhash` permits only two ways to cluster files. One might cluster matches by the absolute value of their similarity score (which does not make any sense from our viewpoint), or one might cluster similar files together. For instance, assuming A and B have a score of 40 and B and C have a score of 30, we might put all files in the same bucket. The caveat is A and C may have a score of 0.

On the flip side, `SimFind` gives certainty of what features overlap and hence can generate buckets based on features. Thus, once we identify a file that contains itemized bucket-features, we can categorize them without further consideration. More details are provided in the next section along with a follow-up to the structural correlations ascertained above.

5.7 Template/header carving

The template / header carving capability of our implementation was further explored after the Sec. 5.6.3.1 findings. Review of the literature presented in the Background and related work was done post-assessment for this very reason.

As mentioned, having more metadata available allows more detailed analysis of objects. The similarity decisions will depend on user needs and may require ad hoc modification. For testing (using the first prototype; these are not present in the second and the database shell was used for finding *More-than-ten*), we generated interesting heuristic values for every match found (these existed for each file queried against the database), namely:

- *Feat-count* is the total number of overlapping features.
- *Feat-total* is the number of all features making up the smaller object.
- *Avg-entropy* is the average entropy of all overlapping features.
- *Byte-length* is the total byte length of the overlapping features.
- *Total-byte-length* is the total byte length of the smaller object.
- *Features-e > 64* is the number of features having an entropy greater than 64 (which equals 4.0).
- *More-than-ten* is the number of features that have a database counter over ten ($featInstances > 10$). We also call them 'common features'.

- *Top-feat-count* closely parallels More-than-ten and is the absolute value for the top feature, e.g., 30 means that both objects contain a feature that appears in 30 different files (Counter in the FeatureTable is 30).

5.7.1 Categorizing / prioritizing similarity

One problem mentioned in the introduction was that currently there is no scalability / prioritizing of results except for the similarity score. In spite of that, if two files only share structural information, e.g., the same navigation menu in an HTML document, this might be less relevant than actual content similarity.

Therefore, we came up with the following plausible assumption: having a large data set should allow us to identify common patterns especially if the data set comes from the same source. Here, patterns are features that appear frequently (across files). For instance, if we process the file server of a midsize company, it might be possible to identify a 'word template' that is commonly used or other commonalities between files like header / footer information (e.g., all JPG headers include the same camera information).

From an investigative perspective it is definitely corroborative to be able to distinguish between template and content similarity. The following addresses this question and presents our preliminary results. Recollect from Sec 5.4, although we use the term 'template,' this includes structural information, headers, and footers.

A sample output that we used is shown in Table 5.2, illustrating an excerpt from files that matched `001316.html`. The first column is the filename of the matched HTML file

followed by the `sdhash` similarity score. The remaining four columns coincide with the parameters explained in Sec. 5.7. Knowing this, row one states that `001316.html` and `001410.html` obtained a `sdhash` score of 30, the smaller file had 131 features in total, and there was an overlap of 49 features that comprised the score. These 49 features were common features where the top one existed in 24 files. Naturally, our final assessment focused on all matches and not only the onces for `001316.html`.

TABLE 5.2: Additional heuristics created during querying 001316.html. Files in the database that had a feature match are shown with the heuristic (four right-most columns) beside a similarity score output from `sdhash` (second column).

Second file	sdhash-score	Feat-total	Feat-count	More-than ten	Top-feat-count
001410	030	131	049	049	24
002901	030	075	048	048	24
004003	039	068	049	049	24
000428	052	158	110	110	36
000608	067	157	115	110	36
001797	054	158	115	111	36
002773	057	146	115	111	36
003474	060	151	115	111	36
003998	061	152	115	111	36
002360	066	148	115	111	36
003483	060	158	120	111	36
003252	061	158	120	111	36
001317	086	152	147	111	36

5.7.1.1 Template identification

Again, the idea of template identification is based on the assumption that if two files match mostly due to common features then there is a high chance that this is not content similarity, i.e., *Feat-count* $\approx$ *More-than-ten*. Selection of the *More-than-ten* standard was acceptable because only 2126 *t5*-corpus features, 0.0003%, satisfied it (if it was too low getting content in template's stead would be likely).

To verify our assumption we analyzed matches that had *More-than-ten* > 30 (at least 30 common features) and where (*Feat-count* $-$ *More-than-ten*) ≤ 5, e.g., rows 1-10 from the table (these numbers were picked randomly and more research is needed to find optimal thresholds; a plot of a larger dataset would establish this). `SimFind` returned 206 matches based on 48 unique files (30 HTML, 12 PPT, 4 JPG, and 2 DOC). The highest ratio were HTML to HTML matches with 182 in total.

Next, all matches were clustered by *Top-feat-count* and *More-than-ten*. All files that contained equal common features were grouped together. This resulted in 4 groups – 3 contained HTML only and 1 was mixed. To verify the correctness of this grouping, we manually analyzed a few samples from each group.

All selected samples in each of the three HTML groups were correct, e.g., files contained the same navigation menu that contained `Javascript` or `CSS` characteristics. The remaining cross-matched group was PPT, JPG, and DOC. Having a closer look revealed that all matches originated from JPGs (the PPT and DOC files contained JPGs). Delving deeper showed that all JPG files include similar header / structural information. In this case it was something concerning Apple's print manager (see Fig. 5.3; for details see file `003369.ppt` at offset `0x00314b0`).

Manual analysis also spoke to detection in TEXT files. By filling the database with only the *t5*-corpus TEXT documents (743640 features) a `MongoDB` query was performed to identify the most common feature (*featInstances* $= 13902$). This feature was present in

```
00314b0: 0000 ... 0000  ......8BIM......
...
0031500: 222d ... 7574  "-//Apple Comput
0031510: 6572 ... 312e  er//DTD PLIST 1.
0031520: 302f ... 2f77  0//EN" "http://w
0031530: 7777 ... 5444  ww.apple.com/DTD
0031540: 732f ... 2d31  s/PropertyList-1
...
0033260: 6469 ... 0a00  dict>.</plist>..
0033270: 3842 ... 0000  8BIM.......x....
```

FIGURE 5.3: Hex dump of the similarity between JPG files.

six objects[7] much more frequently than the (108) others. Opening these files in a text editor revealed tables (varied entry values without any non-tabular data). Eight other TEXT files that did not have an occurrence of the feature were randomly[8] opened, having no tabular structure. Objects that had only a few counts of the feature had both tabular and non-tabular formatting, and had noticeably fewer columns/rows.

Repeating this with PDF files (1935080 features), a more expected trend was observed. Using *Sublime text* editor to open the 6 objects[9] containing the most common feature revealed long lengths of padding / structural information at the end. The `004987.pdf` was a PDF slide presentation, which had 19 instances of this most common feature, clearly differing from the other files semantically.

[7]Feature hash: 65aea98c57dcd2a1ffb0d35ca20603caaf 7d9f03; 6 files: `001824.text`, `001830.text`, `002050.text`, `003557.text`, `003559.text`, `004255.text`.

[8]Bash *shuf* command was used for picking.

[9]Feature hash: cc350885f73cef0c91b36becb1330680ed bac4e2; `000632.pdf`, `001390.pdf` , `002717.pdf`, `003204.pdf` , `004987.pdf`, `004123.pdf`

5.7.1.2 Content identification

Following the previous section, it should be possible to subtract content similarity by analyzing the difference between *Feat-count* and *More-than-ten*. Therefore, we decided to examine the 484 matches that were (*Feat-count* − *More-than-ten*) > 15 (again, this threshold needs refinement).

Unfortunately a manual inspection of several random matches showed that content identification is more complex. While we received very good results for 'plain' file types like HTML and TEXT, these rates decreased for container file types like DOC, XLS, and PPT, and were even worse for PDF files. We would like to research these further:

- We identified multiple matches that were based on low entropy sequences which is usually due to structural similarity (e.g., `001093.xls` and `004697.xls`). As a countermeasure, we ignored matches that were based on low entropy sequences, i.e., set *Avg-entropy* > 4.0, which dropped 97 pairs – including all investigated samples.

- There are still matches that can be categorized as template and are not content related (e.g., `000277.doc` and `001873.doc`). This is clearly due to the small data set and there are no automated strategies to overcome this downside. However, one may flag features manually in the database with an ignore flag.

5.7.1.3 Combination of template and content identification

To cull the files that have the same template but also share content we looked for high instances of *More-than-ten* features and (*Feat-count* − *More-than-ten*) > 15. With this configuration `SimFind` only returned `001316.html` and `001317.html`. As indicated by the last row of Table 5.2, both files matched by 111 *More-than-ten* features, meaning they shared the same template. However, they also coincided by $147 - 111 = 36$ features that were non-common. Comparing both files manually revealed that they contain an identical paragraph.

5.8 Limitations

We want to make the following limitations clear to the reader:

- `MongoDB` documents do have a maximum size of 16 MB.
- Our implementation only allows querying objects via features and therefore obtaining the full feature list per object cannot be as efficiently done. This could easily be added by creating another set of object documents with embedded feature documents/lists.
- It is not known if ten is the optimal number to use for the the *More-than-ten* metric. Using 10 instead of 21 set the bar higher, but this may have resulted in less accuracy.

5.9 Conclusions

In the end a database approach should be deemed practical for approximate matching. By being able to define similarity at a granular level by using combinations of attributes, thresholds, and queries one may perform more powerful analysis on their data. Amidst a complete solution and traditional methods, this helps to alleviate the similarity score issue listed in the Introduction. Likewise, speed increases could be observed depending on the feature creation algorithm used and the query performed. If speed is more important than the granularity of criteria, however, flat files like those of `sdhash` should be used. Forgiving the imperfection of the template / header capability it could still be useful on certain types of files, helping mitigate the feature priority ambiguity, especially in conjunction with extra fields.

When we look to databases there are things that could be improved, but overall the recall times should be acceptable for most use cases. Plus, a database will maintain the data persistently in a manner that will allow easy additions and manipulations in the future as it grows, unlike flat files. We desire to promote this as a standard, since integrating alongside varying approaches could reap unforeseen benefits. Elasticity of our setup enables this, even for Bloom filter-dependent algorithms.

5.10 Future

As mentioned throughout the paper, there are a few areas needing investment before being able to call this work complete.

While in this paper we focused on the `MongoDB` setup, optimization can probably taken further by a database specialist. Firstly, since there were two variables changed between the prototypes (programming language and database design) it would be useful to confirm whether a direct mapping of the `MySQL` schema to `MongoDB` would be faster or slower than the embedded-document method we used[10]; we hypothesize slower. Second, other tactics like increasing shards (distribution) and enabling threading could optimize the `MongoDB` version as well[11]. If the dataset is large enough Hadoop may be worth examination. Third, it still could be argued that `MongoDB` is not the best for all circumstances. Thoroughly testing / performing theoretical analysis on `InnoDB`, `MyISAM`, `Postgres`, `hashdb`, and others would help make our crux stronger.

Following optimization, experiments would be repeated on a more comprehensive data set. This would also involve elucidating extra attributes and trying to confirm our findings on template by identifying features semantically / manually to confirm that in fact what we label as template is so.

[10] https://code.tutsplus.com/articles/mapping-relational-databases-and-sql-to-mongodb--net-35650 (last accessed 2017-05-02).

[11] http://stackoverflow.com/questions/17752597/strategies-for-fast-searches-of-billions-of-small-documents-in-mongodb (last accessed 2017-05-02).

Some of the various related topics mentioned in the Background and related work rely on databases. An exploration of clustering, machine-learning, and layout detectors could be used in tandem with our database. Applying them to the metadata in the database or in parallel with another database could increase efficiency or better target malware.

5.11 Acknowledgments

We would like to thank Kyle Anthony for some debug help during development of the `C++` version and Rob Schmicker for setting us up with a way to perform remote testing.

5.12 Appendix

5.12.1 Feature extraction and processing

The exact procedure for feature extraction is as follows: each object is split into blocks / chunks utilizing a pseudo random function (PRF) called a rolling hash. PRF uses a window W of a fixed size (7) that rolls through the whole input, byte for byte, and generates a pseudo random number at each step. Let

$$W_p = B_{p-6}B_{p-5}\dots B_p \tag{5.1}$$

denote the window at position p that contains 7 bytes B where $PRF(W_p)$ is the corresponding rolling hash value. Then, let $bs \in \mathbb{N}^+$ represent the block size. If $PRF(W_p) \equiv -1 \bmod bs$, then a trigger point has been located and the end of a block / chunk defines a feature. The next feature starts at byte B_{p+1} and ends at the following trigger point or EOF. Note, if PRF is a valid pseudo random function, then it should trigger every bs bytes, i.e., each feature will have approximately the size bs bytes.

To enhance the runtime efficiency, we skip $\lfloor {}^{bs}/_{3} \rfloor$ after we identify a trigger point which also ensures a minimum length for the next block. Lastly, since features should be stored in a space efficient / compressed manner, all features are hashed using *SHA-1* and then stored. Please note that because of the approach used feature sets may be easily effected

by changes in the bytestream, e.g., deleting 7 bytes from `002717.pdf` at byte 002017 resulted in a feature reduction from 3288 originally to 80.

5.12.2 `MySQL` database structure

In order to identify similarities, we need to know what features belong to which objects (files), i.e., we have to store objects, features, and their correlations. Because one file consists of multiple features and a feature can be found in different objects (many-to-many relation) an associative entity is required. A visualization of the database schema is shown in Fig. 5.1. However, there are possibilities to improve this database design, discussed in Sec. 5.12.5.

Besides the mandatory hashes each table can contain additional attributes. Those attributes can be used later to weight the level of similarity between two objects. In the following subsections we name and motivate some attributes that can be helpful for different scenarios[12]. It's important to point out this is not a complete list and attributes may be added / removed depending on use case.

[12]Remark: some of the annotated attributes can also be received by querying the database (use of 'join', 'group', or 'count' of ObjectFeatureTable or FeatureTable). However, due to runtime efficiency, it is faster to store these values.

5.12.2.1 ObjectTable

This table can store information about the original object itself and will be the smallest table. The only mandatory column is the object-hash which acts as the primary key. For the remainder of this paper we will use *SHA-1* as the default hash function, giving the primary key 160 bits or 20 bytes. The following are optional additional attributes:

- *Object-type* can be a specific file type, the sequence number of a network packet, or the identifier for a complete hard drive. If assumed that we process files only and the file type is described by the file extension, 4 characters should be sufficient (4 bytes).
- *Size* stores the overall size of the object in bytes. Again assuming that we handle files only, and the maximum file size is 10 GB, we need a column that can store ~ 34 bits which is ~ 5 bytes.
- *Feature-count* contains the total features in this object. The amount of features depends on the object size and the blocksize bs. If the maximum object size is 10 GB and $bs = 256 = 2^8$ approximately 4 bytes are needed[13].
- *Metadata* can contain additional data that may be necessary. Examples for files are timestamps or file location on a device. Since the actual size highly depends on the data, it is difficult to make assumptions about this attribute.

[13] This can be estimated by $\log_2((10 \cdot 2^{30})/2^8) = \log_2(10 \cdot 2^{22}) = \log_2(10) + \log_2(2^{22}) \approx 4 + 22 = 26$ bits $/8 \approx 4$ bytes.

5.12.2.2 FeatureTable

The FeatureTable contains the feature hashes as the primary key – the only mandatory column – and requires 20 bytes. This table contains more records than the ObjectTable. Optional attributes include:

- *Counter* tallies frequency of a specific feature. Note, as this is part of the FeatureTable we know how often specific features appear in a given set (database). If a feature has a high count this can be an indicator that it is a common feature like header information, color, table, or structural information and probably should be ignored (defined as template). It is hard to make assumptions how often a specific feature appears in a set. If we assume a set of 120 million files (current size of the National Software Reference Library) and that there is a feature appearing in all files (very unlikely), we reserve 3-4 bytes.

- *Byte-length* can be used to calculate the exact similarity between two files. For instance, it is possible to add up the byte length of all existing features and return the exact overlap between two objects. The byte length of a feature relies on the blocksize bs. Usually a rather small bs is preferred in order to have finer granularity to detect similarity, but if long runs of zeros exist where PRF will not trigger this may not be so. Analyzing our test set showed that the longest feature has

1,851,840 bytes in file `001029.pdf`[14]. This is equal to $\log_2(1,851,840) \approx 21$ bits (with $bs = 256$). But we will only use only 2 bytes.

- *Shannon-entropy* reflects the relative probability of information content of a specific feature. This attribute can be used to ignore low-entropy features when they may be less relevant, e.g., a long run of zeros. We made one minor change to this: typically the entropy e of a byte sequence is a float number with $0.0 \leq e \leq 8.0$, but `SimFind` does not require exact entropy, allowing us to map it between 0 and 127 by $e' = \lceil {}^{e}/_{8} \cdot 127 \rceil$. Entropy e' hence can be stored in a single byte.

- *Randomness* can supplement entropy since there can be sequences that have high entropy but low randomness due to a pattern / trend (e.g., 1234567890). This might be done by pseudorandom sequence testing. Similar to entropy, our approach only needs a rough estimation of the randomness and therefore one byte is sufficient.

- *Ignore-flag* can indicate if a feature should be processed during queries or not, based on an a-priori measurement; e.g., insufficient entropy. By default this flag should be set to zero and modified in a second round after the database is populated. Since this is a boolean value, one bit is sufficient for this attribute.

[14]This sequence has very low entropy ($e = 0.13$), consisting of almost only zeros. Subsequently, the pseudo random function does not trigger.

5.12.2.3 ObjectFeatureTable

This associative entity links the ObjectTable to the FeatureTable by using their primary keys. As a consequence, this is the table with the most rows and therefore should not contain many attributes. Possible attributes besides the two primary keys would be:

- *Counter* tallies the frequency of a single feature in an object, since the same feature may be contained multiple times in the same object. This should be very unlikely and therefore we recommend one or two bytes only.
- *Offset* is the address where to find a specific feature in a file (for future / additional analysis). Assuming a maximum of 10 GB, we would need 4-5 bytes.

Since we do not have a unique column and do not set a primary key, most databases set a default primary key. For instance, `MySQL` creates a "6-byte field that increases monotonically as new rows are inserted. Thus, the rows ordered by the row ID follow insertion order"[15].

This table is mainly used to find the objects that contain a specific feature. More precisely, once a feature is found in the FeatureTable, this table is queried with the featureHash. In order to find the featureHash efficiently this table requires indexing this column.

[15] http://dev.mysql.com/doc/refman/5.0/en/innodb-index-types.html (last accessed 2017-05-02).

5.12.2.4 Impact of indexing

Along with the actual payload discussed in the previous sections each index requires additional storage. Since table queries are based on hash values the primary key of both ObjectTable and FeatureTable needs an index[16]. With respect to the associative entity, we recommend creating an index over the featureHash column since some queries might require finding the object of a feature. In short, this database schema requires 3 indexes.

According to different blog entries and our own experiments, it is almost impossible to predict the actual size of an index. Therefore, we cannot estimate the needed storage without running various experiments.

5.12.3 `MySQL` database queries

Queries for the `MySQL` prototype are summarized below (ordered from least complex to most complex). These were tweaked in the second version to grant more informative output options.

General match: A general match query only answers whether there is any similar object in the database set. This is a yes-or-no query and thus one will not know which is/are the similar object(s). In this specific case only the FeatureTable needs to be queried. Note, if this fulfills the usage scenario requirements, the overall size of the database will be way smaller as the other two tables can be dropped.

[16]Some database management systems like `MySQL` index their primary key by default.

Any match: This query will return one similar object only; it will not return all matches (if there are multiple matches) neither will it return the best match. Although the FeatureTable will be the most queried, there needs to be a few queries to both other tables in order to identify one related file.

All matches: The most complex query is 'all matches' which will return all objects that are similar. In this case, for each feature, both other tables are needed.

By having different object comparison options, we can focus on the actual definition of similarity. Having all feature hashes and additional attributes allows multiple ways to calculate / define similarity. The most trivial approach might be to consider only the amount of features found in the dataset (general match). A more precise approach would be to calculate the overall byte length of similarity (based on the length attribute of features). Of course, one needs to consider that features might belong to different files. The focal point may only be one file but we can define even more complex strategies. If desired features that contain long strings of zeroes can be neglected, as they may represent irrelevant data. Thus, we can focus on features that overcome a certain entropy and/or randomness value. A possible query might be 'there needs to be at least 5 matching features each having an entropy higher than 40 (≈ 2.5) and a total minimum length of 700 bytes'.

5.12.4 MySQL implementation details

To run initial tests, we created a prototype application which is separated into two parts: processing-engine and database. The goal of this prototype was to demonstrate the overall feasibility of our approach; simplicity was our aim in lieu of optimization. We chose `Python` for programming the processing engine (details about runtime efficiency are discussed in Sec. 5.12.9).

With respect to the database, we utilized a `MySQL` 5.6.21 database server that comes with XAMPP[17]. We neglected optimizing the database environment and adopted the default settings for caching or page size.

In order to use `SimFind`, one has to install a `MySQL` server on their system. We recommend XAMPP since it comes with phpMyAdmin which allows intuitive graphical access to the database. Next, one may download the zip file (https://github.com/vikhari/sim-find) and create the tables as described in `mysql-commands.sql`. A user would also have to change the `MySQL` credentials in `db_connector.py`. To process a specific folder, change the folder (hard-coded) `SimFind.py` and run the tool.

For comparing objects against this database, use `hash-file-creator.py` and `comparer.py`. The first tool outputs all feature hashes to the screen (you can pipe them into a file), and the second script can be used to compare these feature hashes. Note, this was only an early stage prototype and was used for feasibility testing.

[17] https://www.apachefriends.org/index.html (last accessed 2017-05-02).

5.12.5 Possibilities to enhance the MySQL database schema

The aforementioned sections touched on obvious database design choices. Depending on the amount / type of data, there are possibilities for ameliorating the database schema.

5.12.5.1 Using IDs as primary keys

The ObjectFeatureTable links both other tables and will be the largest table row-wise. If our schema uses hash values as primary keys (20 bytes) the ObjectFeatureTable needs to contain those too. Depending on the amount of data, it may be more space savvy to auto increment IDs rather than linking both tables. For instance, let's assume 1 TB of data and a $bs = 256$. This will result in about $2^{40}/2^{8} = 2^{32}$ features, exactly the length of an integer. Likewise, utilizing auto increment INT IDs for ObjectTable and FeatureTable our associative entity demands only two 4 byte keys, instead of two 20 byte keys. Still, we have to remember that we need additional indexes, and both other tables also contain an additional attribute. So INT IDs should decrease the size per record of ObjectFeatureTable from 40 bytes to 8 bytes but increases both other tables by 4 bytes each.

5.12.5.2 Extending the FeatureTable by objectID

A second property we have to consider is the actual amount of overlapping features. For instance, it is possible that there will be no overlapping features in a set. In this case,

we can drop ObjectFeatureTable and store the primary keys of the ObjectTable directly with the features.

This brings us to a hybrid approach. Adopting the axiom that only a small number of all features coincide, we can reduce storage space by saving the object primary key directly with the feature and only create an entry in the ObjectFeatureTable for additional overlapping features. For instance, let us assume obj_1 and obj_2 contain the same feature f where obj_1 is processed first. Thence we store the ID / primary key of obj_1 directly with the f. For obj_2 we create a record in ObjectFeatureTable linking it to the f. This invariably reduces the size of the associative entity. The `MongoDB` version can be likened to this (5.5.2).

5.12.6 Initial `MySQL` theoretical calculations

Before testing the theoretical limitations based on the database design was studied. The bytes of each relation's attribute are not restated here (see 5.12.2 for details). Also, recall that in addition to the actual "payload," pages are not completely filled (typically a relational database management system (RDBMS) only fills 50 % to 80 % of a page) and that indexes require additional storage. Consequently, it is challenging to predict the exact storage space and it may differ between RDBMSs. Albeit, we can estimate the amount of records per each table (underneath), which may give users the ability to develop their own scaled estimations for their practice.

ObjectTable: The object table contains one entry for each object and thus it has $|S|$ records.

FeatureTable: The total number of features is independent of the amount of objects but based on the total set size μ. Assuming a valid pseudo random function and a blocksize $bs = 256$, the amount of features should be approximately $\delta \approx \mu/2^8$. Thus, this table will have at most δ records. We say 'at most' by virtue of neglecting duplicates. In a real-world scenario identical features would be observed, making the total amount less than δ.

ObjectFeatureTable: As already mentioned, this table will have one record for each feature which is denoted exactly by δ.

5.12.7 Impact of `MySQL` database schema based on real-world data

The main reason for this evaluation was to identify the optimal database schema for real-world data which heavily depends on the amount of coinciding features. The results for the different schemas described in Sec. 5.12.5 in the Appendix are given in Table 5.3, where

- Schema 1 is equal to the schema described in Fig. 5.1 (hash values as primary keys),
- Schema 2 shows the results for the schema presented in Sec. 5.12.5 paragraph one (IDs as primary keys), and

- Schema 3 describes the outcome when using the schema discussed in Sec. 5.12.5 paragraph two (hybrid approach).

Prior expounding we want to elucidate the `MySQL` default behaviour of adding the size / length of the primary key index to 'data length' – it does not list 'index length' separately.

Turning to the database schema decision, it becomes obvious that 'rows' for ObjectTable and FeatureTable are independent from the database schema since they rely on the set size and algorithm, respectively. We hypothesized to have approximately $\mu/bs \approx 7,467,381$ features with our *t5*-corpus setup, which would mirror the number of rows in the FeatureTable and the ObjectFeatureTable. Contrary to this, the actual amount of rows is smaller for all tables.

- It was discovered the *t5*-corpus contains duplicates and thus the total number is less (objects are only stored once).
- The amount of features in the database is about 23% smaller than the calculated upper limit. Querying the database returned an average feature length of 383 instead of 256, a product of many low entropy features – nearly 30% of all features have an entropy less than or equal to 4.0. Furthermore, we did not consider identical features in the same object.

Comparing the recorded *Avg row length* (Table 5.3) to our calculated row length from Sec. 5.12.6 shows that the actual length is larger. This results from the primary key index structure as well as from the fill ratio of the pages.

TABLE 5.3: Results obtained from testing different database schemas. Significance in Sec. 5.12.7, full logistics in Sec. 5.12.5 of Appendix.

		Schema 1	Schema 2	Schema 3
FeatureTable	Data length (MiB)	370.0	248.8	262.8
	Index length (MiB)	0	274.0	237.0
	Rows	5,005,807	5,005,807	5,005,807
	Avg. row length	77	52	58
ObjectTable	Data length (MiB)	0.3	0.2	0.2
	Index length (MiB)	0	0.2	0.2
	Rows	4,449	4,449	4,449
	Avg. row length	65	49	50
ObjectFeatureTable	Data length (MiB)	334.8	168.7	5.5
	Index length (MiB)	278.0	76.6	2.5
	Rows	5,146,018	5,146,018	140,211
	Avg. row length	68	34	36
	Total size (MiB)	983.1	741.5	508.2
	Compression (%)	53.9	40.7	27.9

As indicated by the minor difference in *Rows* for FeatureTable and ObjectFeatureTable (Schema 1 and 2), the amount of overlapping features is low. More precisely, analyzing Schema 3 shows that the *t5*-corpus has around $^{140,211}/_{5,146,018} = 0.0272 = 2.72\,\%$ overlapping features. Thus, the hybrid schema resulted in the smallest database. Most noteworthy were the differences in storage usage as indicated by the *Compression* row. Here, compression correlates the total database size (indexes plus payload) and the total size, μ. The compression was obtained with Schema 3, at approximately 30%, but we only tested a small set. To avoid taking a stab in the dark we have provided further theory in Sec. 5.12.8.

5.12.8 More theoretical calculations – larger datasets

In the following we perform a theoretical evaluation of the chosen schema based on a larger dataset. Let us assume a set containing $|S| = 2^{20}$ objects having a total size of $\mu = 2^{38}$

bytes. The blocksize remaining $bs = 256$, we assume a set of over one million objects with a total size of 256 GiB. This results in an average object size $\mu/|S| = 2^{18}$ bytes = 256 KiB.

According to the findings in Sec. 5.12.7 the ObjectTable would have 2^{20} records and both other tables (depending on the schema) would contain roughly $2^{38}/2^{8} = 2^{30}$ records. Despite these large numbers modern database systems can handle these amounts of records. Conversely, storage space will not be an issue. Taking on the worst case where the compression is only 52.4 %, a workstation would require a hard disk of approximately $\mu/2 = 128$ GiB.

5.12.9 Estimated `MySQL` runtime efficiency

Broadly speaking, `SimFind` can be separated into two parts: the processing engine and the database lookup. The former is almost identical to the `ssdeep` approach but `SimFind` utilizes *SHA-1* instead of *FNV* and stores the hashes in a database rather than a Bloom filter. Time for database generation should be very similar to `mrsh-v2` (fast), not that it should be the highest concern since database generation usually only happens once and is thereafter updated sporadically; it does not continuously affect runtime.

The more critical part is database lookup. In section 5.3.1 of (Breitinger, Baier, & White, 2014) the authors estimated the runtime for a large dataset. They measured the lookup time for the *t5*-corpus which was about 20 min, inspecting a 200 GiB file set against a 1500 GiB file set in a database. They concluded that it will take over 3.5 years. Abiding by those numbers, `SimFind` will approximately output $200 \cdot 2^{30}/2^{8} \approx 2^{29.6}$ features. By

assuming we can perform 15,000 ($\approx 2^{13.9}$) queries per second[18] `SimFind` should need $2^{29.6}/2^{13.9} \approx 2^{15.7}$ seconds or 14 h and 48 min.

5.12.10 Sample output for `C++-MongoDB` prototype

The output of the `run_sf_create` script for creating the database (`C++-MongoDB` version) is shown below in Fig. 5.4, followed by the outputs for the menu/queries of `SimFind`.

[18] An online search showed various numbers starting from a few thousand up to 1 million https://www.flamingspork.com/blog/2014/06/03/1-million-sql-queries-per-second-mysql-5-7-on-power8 (last accessed 2017-05-02). However, we decided to use a small number.

```
vik@ubuntu:~/Downloads/simfind$ ./run_sf_create
START  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Finding files...
16 files found.

Entering objects & features into database...
53      features inserted       -       "004303.gif"

22      features inserted       -       "004775.text"

34      features inserted       -       "004784.text"

138     features inserted       -       "004580.doc"

358     features inserted       -       "004297.gif"

127     features inserted       -       "004441.pdf"

123     features inserted       -       "000300.ppt"

104     features inserted       -       "002934.html"

5321    features inserted       -       "004427.doc"

172     features inserted       -       "001076.xls"

61      features inserted       -       "001083.xls"

147     features inserted       -       "004858.jpg"

981     features inserted       -       "004838.jpg"

27028   features inserted       -       "000134.ppt"

20      features inserted       -       "002952.html"

1897    features inserted       -       "004141.pdf"

Creation time: 12.7933 seconds

DONE  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

FIGURE 5.4: Output of creation script.

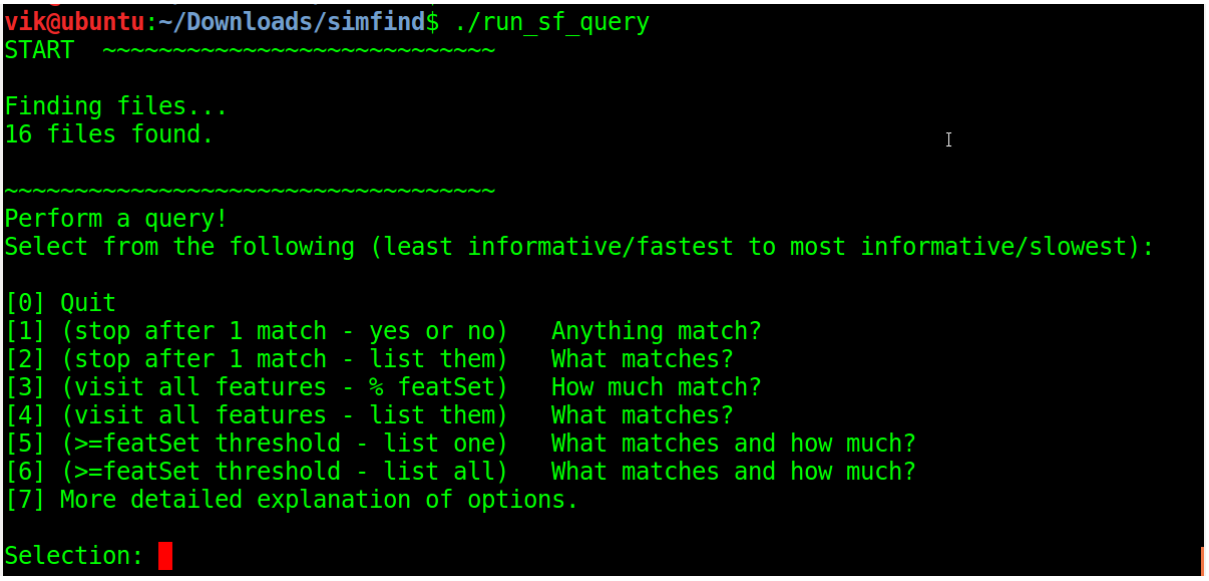

FIGURE 5.5: Output of main menu.

```
Selection: 1

Querying objects & features in database...

Yes!    -       "004303.gif"

Yes!    -       "004775.text"

Yes!    -       "004784.text"

Yes!    -       "004580.doc"

Yes!    -       "004297.gif"

Yes!    -       "004441.pdf"

Yes!    -       "000300.ppt"

Yes!    -       "002934.html"

Yes!    -       "004427.doc"

Yes!    -       "001076.xls"

Yes!    -       "001083.xls"

Yes!    -       "004858.jpg"

Yes!    -       "004838.jpg"

Yes!    -       "000134.ppt"

Yes!    -       "002952.html"

Yes!    -       "004141.pdf"

Query time: 10.4742 seconds
```

FIGURE 5.6: Output of first query.

```
5ce73b19386d8891878e01dacaff4adbf0e87e25
004775.text


508f2c72acb152195331e59bfafcbb9522ff9ed2
004784.text


5b3cadee0bd677cf8dd26d53a83e5d9636699a1a
004580.doc


a36a6718f54524d846894fb04b5b885b4e43e63b
003979.pdf
004303.gif
003576.gif
004161.pdf
001227.gif
003409.pdf
002380.doc
002439.ppt
000696.doc
003813.gif
```

FIGURE 5.7: Output of second query.

```
Selection: 3

Querying objects & features in database...

"004303.gif"    -       Percent of features found: 100%

"004775.text"   -       Percent of features found: 100%

"004784.text"   -       Percent of features found: 100%

"004580.doc"    -       Percent of features found: 100%

"004297.gif"    -       Percent of features found: 100%

"004441.pdf"    -       Percent of features found: 100%

"000300.ppt"    -       Percent of features found: 100%

"002934.html"   -       Percent of features found: 100%

"004427.doc"    -       Percent of features found: 100%

"001076.xls"    -       Percent of features found: 100%
```

FIGURE 5.8: Output of third query.

```
d6743df283b6916b070ffea2e9ebfa26127806e5  -  1
004141.pdf

1f0accceff6353442df695519de7f7d7c2939fe1  -  1
004141.pdf

7681715b1be4923b558d51c3095e982b0d434008  -  3
001934.pdf
004141.pdf
000975.doc

a1399e459c0a9ec27fcabb449fcf86336eba86bd  -  1
004141.pdf

b3de7aa0c9adf750154ddf563a850d8d35069c9c  -  1
004141.pdf

c467973750092abf50665062470375580a72dd81  -  1
004141.pdf

"004141.pdf"    -       Percent of features found: 100%
Number of features with more than 10 instances: 22
```

FIGURE 5.9: Output of fourth query.

```
Selection: 5
Threshold (0-1): 0.28


Querying objects & features in database...

"004303.gif"    -       First similar object above threshold:
004303.gif

"004775.text"   -       First similar object above threshold:
004775.text

"004784.text"   -       First similar object above threshold:
004784.text

"004580.doc"    -       First similar object above threshold:
004580.doc

"004297.gif"    -       First similar object above threshold:
004297.gif

"004441.pdf"    -       First similar object above threshold:
004441.pdf
```

FIGURE 5.10: Output of fifth query.

```
Selection: 6
Threshold (0-1): .00001


Querying objects & features in database...

"004775.text"   -       All similar objects above threshold:
004775.text             (22 features)

"004784.text"   -       All similar objects above threshold:
004784.text             (34 features)

"004580.doc"    -       All similar objects above threshold:
004580.doc              (213 features)

"004297.gif"    -       All similar objects above threshold:
000055.xls              (1 features)
000144.pdf              (1 features)
000146.pdf              (1 features)
000233.gif              (2 features)
000241.gif              (1 features)
000291.ppt              (1 features)
000292.ppt              (1 features)
```

FIGURE 5.11: Output of sixth query.

```
[7] More detailed explanation of options.

Selection: 7


Keep in mind that each of the options is performed for each file in the query_files f
older.

[1] Iterate through the query feature set and determine if a feature is in the databa
se. Stop after the first match. Return 'no' if no features in the query set are found
.

[2] Same as [1] but if there is a match list the objects that contain the feature.

[3] Iterate through the query feature set and determine if a feature is in the databa
se. Don't stop until the entire feature set has been iterated. Report the percentage
of query features found.

[4] Same as [3] but in addition list all objects that contain each feature (including
 the total number of instances of the feature in the database). At the end also repor
t number of features with instances over ten.

[5] Set a threshold percentage (between 0 and 1). Iterate through the query feature s
et and retrieve all objects that contained each feature. Then iterate through the obj
ects for each feature tallying the number of times an object occurs. Stop and report
the filename of the first object that has {threshold x total number of features in qu
ery feature set} occurrences. In other words, list the first similar object.

[6] Same as [5] but iterate through all features and all objects. List all similar ob
jects.


Time (T) for each query (for each object):
q = total time needed to query database
f = time needed to query for single feature
t = user-specified threshold percentage (decimal)
n = number features in query object's feature set
i = time needed to iterate objects of each feature in memory
[1] T = ( f*1  <= q <= f*n)
[2] T = ( f*1  <= q <= f*n)
[3] T = (         q  = f*n)
[4] T = (         q  = f*n)
[5] T = (f*n*t <= q <= f*n) + i
[6] T = (f*n*t <= q <= f*n) + i


~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

FIGURE 5.12: Output of help selection.

Chapter 6

Conclusions

As mentioned in the prior chapters the transmogrification of cybercrime has created a multitude of proverbial monsters that appear too large to handle. However, as proven by the results of these works we realize there are bold trends we can follow to help us better track criminal actions, and technical tactics we can use to thwart them. To reiterate, a database of CuFAs applying approximate matching (potentially more than one approach at a time) is the logical choice when exploring where humans have trodden. It could assimilate cloud forensic tools and bring unforeseen links between cases to light. Bytewise measures are extremely effective in saving the most sought after resource in the cybersecurity and digital forensics domain – time.

To avoid belaboring conclusions already written in each chapter, simply think about how scientific progress will go in the coming years. Artificial intelligence will allow crimes to

be automated at a level never experienced. This dynamic environment will require applications like approximate matching to push our current limits of automatic extraction of incriminating evidence. After all, with all the information that a person now accumulates in a lifetime on storage devices, pieces of evidence are just a small drop in the bucket.

References

Aleman-Meza, B., Halaschek-Wiener, C., Sahoo, S. S., Sheth, A., & Arpinar, I. B. (2005). Template based semantic similarity for security applications. In *International conference on intelligence and security informatics* (pp. 621–622).

Al Fahdi, M., Clarke, N. L., & Furnell, S. M. (2013). Challenges to digital forensics: A survey of researchers & practitioners attitudes and opinions. In *Information security for south africa, 2013* (pp. 1–8).

Allen, B. (2015). Hashdb. *https://github.com/simsong/hashdb*.

Aminnezhad, A., Dehghantanha, A., & Abdullah, M. T. (2012). A survey on privacy issues in digital forensics. *International Journal of Cyber-Security and Digital Forensics (IJCSDF)*, *1*(4), 311–323.

Åstebøl, K. P. (2012). *mvhash - a new approach for fuzzy hashing* (Unpublished master's thesis). Gjøvik University College.

Azfar, A., Choo, K.-K. R., & Liu, L. (2015). *Forensic taxonomy of popular android mhealth apps* (Tech. Rep.). University of South Australia. Retrieved from http://arxiv.org/ftp/arxiv/papers/1505/1505.02905.pdf

Baggili, I., BaAbdallah, A., Al-Safi, D., & Marrington, A. (2013). Research trends in digital forensic science: An empirical analysis of published research. In *Digital forensics and cyber crime* (pp. 144–157). Springer.

Baggili, I., Marrington, A., & Jafar, Y. (2014). Performance of a logical, five-phase,

multithreaded, bootable triage tool. In *Advances in digital forensics x* (pp. 279–295). Springer.

Baier, H. (2015). Towards automated preprocessing of bulk data in digital forensic investigations using hash functions. *it-Information Technology*, *57*(6), 347–356.

Baier, H., & Breitinger, F. (2011, May). Security Aspects of Piecewise Hashing in Computer Forensics. *IT Security Incident Management & IT Forensics (IMF)*, 21–36. doi: 10.1109/IMF.2011.16

Bär, D., Biemann, C., Gurevych, I., & Zesch, T. (2012). Ukp: Computing semantic textual similarity by combining multiple content similarity measures. In *Proceedings of the first joint conference on lexical and computational semantics-volume 1: Proceedings of the main conference and the shared task, and volume 2: Proceedings of the sixth international workshop on semantic evaluation* (pp. 435–440).

Bariki, H., Hashmi, M., & Baggili, I. (2011a). Defining a standard for reporting digital evidence items in computer forensic tools. In *Digital forensics and cyber crime* (pp. 78–95). Springer.

Bariki, H., Hashmi, M., & Baggili, I. (2011b). Defining a standard for reporting digital evidence items in computer forensic tools. In *Digital forensics and cyber crime* (pp. 78–95). Springer. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-19513-6_7#page-1

Barwick, H. (2014, May). *Data sovereignty laws hamper international crime investigations: Afp.* Retrieved from http://www.computerworld.com.au/article/544591/data_sovereignty_laws_hamper_international_crime_investigations

_afp

Beebe, N. (2009). Digital forensic research: The good, the bad and the unaddressed. In *Advances in digital forensics v* (pp. 17–36). Springer.

Bertoni, G., Daemen, J., Peeters, M., & Assche, G. V. (2008, October). *Keccak specifications* (Tech. Rep.). STMicroelectronics and NXP Semiconductors.

Birajdar, G. K., & Mankar, V. H. (2013, April). Digital image forgery detection using passive techniques: A survey. *Digital Investigation*, *10*(3), 226–245.

Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, *13*, 422–426.

Brady, O., Overill, R., & Keppens, J. (2014, Sept). Addressing the increasing volume and variety of digital evidence using an ontology. In *Intelligence and security informatics conference (jisic), 2014 ieee joint* (p. 176-183). doi: 10.1109/JISIC.2014.34

Breitinger, F. (2014). *On the utility of bytewise approximate matching in computer science with a special focus on digital forensics investigations* (Doctoral dissertation, Technical University Darmstadt). Retrieved from http://tuprints.ulb.tu-darmstadt.de/4055/

Breitinger, F., Åstebøl, K. P., Baier, H., & Busch, C. (2013, March). mvhash-b - a new approach for similarity preserving hashing. In *It security incident management and it forensics (imf), 2013 seventh international conference on* (p. 33-44). doi: 10.1109/IMF.2013.18

Breitinger, F., & Baggili, I. (2014, September). File detection on network traffic using approximate matching. *Journal of Digital Forensics, Security and Law (JDFSL)*,

9(2), 23–36. Retrieved from http://ojs.jdfsl.org/index.php/jdfsl/article/view/261

Breitinger, F., & Baier, H. (2012a, May). A Fuzzy Hashing Approach based on Random Sequences and Hamming Distance. In *7th annual Conference on Digital Forensics, Security and Law (ADFSL)* (pp. 89–100).

Breitinger, F., & Baier, H. (2012b). Performance issues about context-triggered piecewise hashing. In P. Gladyshev & M. Rogers (Eds.), *Digital forensics and cyber crime* (Vol. 88, p. 141-155). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-35515-8_12 doi: 10.1007/978-3-642-35515-8_12

Breitinger, F., & Baier, H. (2012c). Properties of a similarity preserving hash function and their realization in sdhash. In *Information security for south africa (issa), 2012* (p. 1-8). doi: 10.1109/ISSA.2012.6320445

Breitinger, F., & Baier, H. (2013). Similarity preserving hashing: Eligible properties and a new algorithm mrsh-v2. In M. Rogers & K. Seigfried-Spellar (Eds.), *Digital forensics and cyber crime* (Vol. 114, pp. 167–182). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-39891-9_11 doi: 10.1007/978-3-642-39891-9_11

Breitinger, F., Baier, H., & White, D. (2014, May). On the database lookup problem of approximate matching. *Digital Investigation*, *11, Supplement 1*(0), S1 - S9. Retrieved from http://www.sciencedirect.com/science/article/pii/S1742287614000061 (Proceedings of the First Annual DFRWS Europe) doi: http://dx.doi.org/10.1016/j.diin.2014.03.001

Breitinger, F., Guttman, B., McCarrin, M., Roussev, V., & White, D. (2014, May). *Approximate matching: Definition and terminology* (Special Publication 800-168). National Institute of Standards and Technologies. Retrieved from http://dx.doi.org/10.6028/NIST.SP.800-168

Breitinger, F., Rathgeb, C., & Baier, H. (2014, September). An efficient similarity digests database lookup - A logarithmic divide & conquer approach. *Journal of Digital Forensics, Security and Law (JDFSL)*, *9*(2), 155–166. Retrieved from http://ojs.jdfsl.org/index.php/jdfsl/article/view/276

Breitinger, F., & Roussev, V. (2014, May). Automated evaluation of approximate matching algorithms on real data. *Digital Investigation*, *11, Supplement 1*(0), S10 - S17. Retrieved from http://www.sciencedirect.com/science/article/pii/S1742287614000073 (Proceedings of the First Annual DFRWS Europe) doi: http://dx.doi.org/10.1016/j.diin.2014.03.002

Breitinger, F., Stivaktakis, G., & Baier, H. (2013, August). FRASH: A framework to test algorithms of similarity hashing. In *13th Digital Forensics Research Conference (DFRWS'13)*. Monterey.

Breitinger, F., Stivaktakis, G., & Roussev, V. (2013, Sept). Evaluating detection error trade-offs for bytewise approximate matching algorithms. *5th ICST Conference on Digital Forensics & Cyber Crime (ICDF2C)*.

Breitinger, F., Ziroff, G., Lange, S., & Baier, H. (2014, January). sahash: Similarity hashing based on levenshtein distance. In *Tenth annual ifip wg 11.9 international conference on digital forensics (ifip wg11.9)*.

Brinson, A., Robinson, A., & Rogers, M. (2006). A cyber forensics ontology: Creating a new approach to studying cyber forensics. *digital investigation*, *3*, 37–43. Retrieved from http://www.dfrws.org/2006/proceedings/5-Brinson.pdf

Broder, A. Z. (1997). On the resemblance and containment of documents. In *Compression and complexity of sequences (sequences'97)* (pp. 21–29). IEEE Computer Society.

Broder, A. Z., Charikar, M., Frieze, A. M., & Mitzenmacher, M. (1998). Min-wise Independent Permutations. *Journal of Computer and System Sciences*, *60*, 327-336.

Bureau of Labor Statistics, U.S. Department of Labor. (2014, January). *Occupational outlook handbook, 2014-2-15 edition* (Tech. Rep.). Retrieved from http://www.bls.gov/ooh/computer-and-information-technology/information-security-analysts.htm

Casey, E., Back, G., & Barnum, S. (2015). Leveraging cybox to standardize representation and exchange of digital forensic information. *Digital Investigation*, *12*, S102–S110. Retrieved from http://www.sciencedirect.com/science/article/pii/S1742287615000158

Castle, G. (2014a). *Black hat usa 2014 - forensics: Grr find all the badness, collect all the things.* Retrieved from https://www.youtube.com/watch?v=DudGrSv26NY

Castle, G. (2014b). Grr artifacts. In *Blackhat.* Retrieved from https://www.blackhat.com/docs/us-14/materials/us-14-Castle-GRR-Find-All-The-Badness-Collect-All-The-Things-WP.pdf

Castle, G., & Metz, J. (2015). *Forensic artifact labels.* Retrieved

from https://github.com/ForensicArtifacts/artifacts/blob/master/docs/Artifacts%20definition%20format%20and%20style%20guide.asciidoc

Cesarini, F., Marinai, S., & Soda, G. (2002). Retrieval by layout similarity of documents represented with mxy trees. In *International workshop on document analysis systems* (pp. 353–364).

Chao, H., & Fan, J. (2004). Layout and content extraction for pdf documents. In *International workshop on document analysis systems* (pp. 213–224).

Charikar, M. S. (2002). Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual acm symposium on theory of computing* (pp. 380–388).

Chen, L., & Wang, G. (2008). An efficient piecewise hashing method for computer forensics. In *Knowledge discovery and data mining, 2008. wkdd 2008. first international workshop on* (pp. 635–638). doi: 10.1109/WKDD.2008.80

Chivers, H. (2014). Private browsing: A window of forensic opportunity. *Digital Investigation*, *11*(1), 20 - 29. Retrieved from http://www.sciencedirect.com/science/article/pii/S1742287613001229 doi: http://dx.doi.org/10.1016/j.diin.2013.11.002

Choo, K.-K. R. (2011, February). Cyber threat landscape faced by financial and insurance industry. *Trends & issues in crime and criminal justice*(408).

Collange, S., Dandass, Y. S., Daumas, M., & Defour, D. (2009). Using graphics processors for parallelizing hash-based data carving. In *System sciences, 2009. hicss'09. 42nd hawaii international conference on* (pp. 1–10).

Collie, J. (2013). The windows iconcache.db: A resource for forensic artifacts from {USB} connectable devices. *Digital Investigation*, *9*(34), 200 - 210. Retrieved from http://www.sciencedirect.com/science/article/pii/S1742287613000078 doi: http://dx.doi.org/10.1016/j.diin.2013.01.006

Comodo Group Inc. (2013). *PDM (Partial Document Matching)*. https://www.mydlp.com/partial-document-matching-how-it-works/.

Corbin, G. (2014). *The google chrome operating system forensic artifacts* (Doctoral dissertation, Utica College). Retrieved from http://pqdtopen.proquest.com/pubnum/1571599.html?FMT=AI

Corpora, D. (2013, April). *NPS Corpus*. http://digitalcorpora.org/corpora/files.

Dardick, G. S. (2010). Cyber forensics assurance. In *8th australian digital forensics conference* (pp. 57–64). School of Computer and Information Science, Edith Cowan University, Perth, Western Australia.

Darnell, J. (2012). *Reply: Rdt&e iwg letter to swgde.* private communication (publicly documented).

Dekel, T., Oron, S., Rubinstein, M., Avidan, S., & Freeman, W. T. (2015). Best-buddies similarity for robust template matching. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2021–2029).

Deselaers, T., & Ferrari, V. (2011). Visual and semantic similarity in imagenet. In *Computer vision and pattern recognition (cvpr), 2011 ieee conference on* (pp. 1777–1784).

Dickson, J. B. (2015, March). *6 ways the sony hack changes everything.* Retrieved from http://www.darkreading.com/risk/6-ways-the-sony-hack-changes-everything-/a/d-id/1319415

Dictionary.com. (2015). *Definition of artifact.* Retrieved from http://dictionary.reference.com/browse/artifact

Dietrich, D., & Adelstein, F. (2015). Archival science, digital forensics, and new media art. *Digital Investigation*, *14*, S137–S145. Retrieved from http://www.sciencedirect.com/science/article/pii/S1742287615000493 (The Proceedings of the 15th Annual {DFRWS} Conference)

Durose, M. R., Walsh, K. A., & Burch, A. M. (2012, August). *Census of publicly funded forensic crime laboratories, 2009* (Tech Report). Bureau of Justice Statistics, U.S. Department of Justice. Retrieved from http://www.bjs.gov/content/pub/pdf/cpffcl09.pdf

European Cybercrime Center. (2014, February). *Police ransomware threat assessment* (Tech Report). Europol. Retrieved from https://www.europol.europa.eu/content/police-ransomware

Faruki, P., Laxmi, V., Bharmal, A., Gaur, M., & Ganmoor, V. (2015). Androsimilar: Robust signature for detecting variants of android malware. *Journal of Information Security and Applications*, *22*, 66–80.

FIPS, P. (1995). 180-1. secure hash standard. *National Institute of Standards and Technology*, *17*, 45.

Fisher, G. E. (2002, November). *Data Formats of the NSRL Reference Data Set (RDS)*

Distribution (Tech. Rep.). National Institute of Standards and Technology.

Fodeh, S., Punch, B., & Tan, P.-N. (2011). On ontology-driven document clustering using core semantic features. *Knowledge and information systems*, *28*(2), 395–421.

Forsyth, D. A., & Fleck, M. M. (1999). Automatic detection of human nudes. *International Journal of Computer Vision*, *32*(1), 63–77.

Fowler, G., Noll, L. C., & Vo, P. (1994–2012). *Fnv hash.* http://www.isthe.com/chongo/tech/comp/fnv/index.html.

Gallagher, P., & Director, A. (1995). *Secure Hash Standard (SHS)* (Tech. Rep.). National Institute of Standards and Technologies, Federal Information Processing Standards Publication 180-1.

Garfinkel, S. (2012). Digital forensics xml and the dfxml toolset. *Digital Investigation*, *8*(3), 161–174. Retrieved from http://calhoun.nps.edu/bitstream/handle/10945/44286/Garfinkel_2012.DI.dfxml.pdf?sequence=1

Garfinkel, S., Farrell, P., Roussev, V., & Dinolt, G. (2009). Bringing science to digital forensics with standardized forensic corpora. *digital investigation*, *6*, S2–S11.

Garfinkel, S., & McCarrin, M. (2014). Hash-based carving: Searching media for complete files and file fragments with sector hashing and hashdb. *digital investigation*, *12*.

Garfinkel, S., Nelson, A., White, D., & Roussev, V. (2010). Using purpose-built functions and block hashes to enable small block and sub-file forensics. *digital investigation*, *7*, S13–S23.

Garfinkel, S. L. (2009, March). *Announcing frag_find: finding file fragments in disk images using sector hashing.* Retrieved from http://tech.groups.yahoo.com/

group/linux_forensics/message/3063

Garfinkel, S. L. (2010). Digital forensics research: The next 10 years. *digital investigation*, *7*, S64–S73.

Garfinkel, S. L. (2013). Digital media triage with bulk data analysis and bulk_extractor. *Computers & Security*, *32*, 56–72.

Garfinkel, S. L., & McCarrin, M. (2015). Hash-based carving: Searching media for complete files and file fragments with sector hashing and hashdb. *Digital Investigation*, *14*, S95–S105.

Goh, T. (2014). *Challenges in windows 8 operating system for digital forensic investigations* (Doctoral dissertation, Challenges in Windows 8 operating system for digital forensic investigations). Retrieved from http://aut.researchgateway.ac.nz/bitstream/handle/10292/7224/GohTT.pdf?sequence=3&isAllowed=y

Goodison, S. E., Davis, R. C., & Jackson, B. A. (2015). Digital evidence and the us criminal justice system.

Greek, A. N. (2013). *Proposal to establish an open source validation and testing center* (Unpublished doctoral dissertation). UTICA COLLEGE.

Grispos, G., Storer, T., & Glisson, W. B. (2013). Calm before the storm: the challenges of cloud. *Emerging Digital Forensics Applications for Crime Detection, Prevention, and Security*, *4*, 28–48.

Gupta, A., Verma, R., & Gupta, G. (2013). Client side forensics investigation of google services. In *Ieee symposium on security and privacy*. Retrieved from https://precog.iiitd.edu.in/events/spsymposium13/

SPSymposium_files/SPsymposium-papers/SPsymposium-paper39.pdf

Gupta, V. (2013). *File fragment detection on network traffic using similarity hashing* (Unpublished master's thesis). Denmark Technical University.

Hack In The Box Security Conference. (2012, December). *Hitb2012kul d2t3 - mikko hypponen - behind enemy lines.* YouTube. Retrieved from https://www.youtube.com/watch?v=0TMFRO66Wv4

Harblson, C. (2015). *Hacking with pictures; new stegosploit tool hides malware inside internet images for instant drive-by pwning.* Retrieved from http://www.idigitaltimes.com/hacking-pictures-new-stegosploit-tool-hides-malware-inside-internet-images-instant-444768

Hibshi, H., Vidas, T., & Cranor, L. F. (2011). Usability of forensics tools: a user study. In *It security incident management and it forensics (imf), 2011 sixth international conference on* (pp. 81–91).

Hu, J., Kashi, R., & Wilfong, G. (2000). Comparison and classification of documents based on layout similarity. *Information Retrieval*, *2*(2), 227–243.

Jaccard, P. (1901). Distribution de la flore alpine dans le bassin des drouces et dans quelques regions voisines. *Bulletin de la Société Vaudoise des Sciences Naturelles*, *37*, 241–272.

Jaccard, P. (1912, February). The distribution of the flora in the alpine zone. *New Phytologist*, *11*, 37–50.

Kessler, G. C. (2010). *Judges awareness, understanding, and application of digital evidence* (Unpublished doctoral dissertation). Nova Southeastern University.

Key, S. (2013). *File block hash map analysis.* Retrieved from https://www.guidancesoftware.com/appcentral/

Kornblum, J. (2006, September). Identifying almost identical files using context triggered piecewise hashing. *Digital Investigation*, *3*, 91–97. Retrieved from http://dx.doi.org/10.1016/j.diin.2006.06.015 doi: 10.1016/j.diin.2006.06.015

Kuhn, T. S. (1970). Book and film reviews: Revolutionary view of the history of science: The structure of scientific revolutions. *The Physics Teacher*, *8*(2), 96–98.

Lallie, H. S., & Briggs, P. J. (2011). Windows 7 registry forensic evidence created by three popular bittorrent clients. *Digital Investigation*, *7*(3 - 4), 127 - 134. Retrieved from http://www.sciencedirect.com/science/article/pii/S1742287610000770 doi: http://dx.doi.org/10.1016/j.diin.2010.10.002

Lee, R. (2015). *Sans digital forensics and incident response poster.* Retrieved from https://digital-forensics.sans.org/blog/2012/06/18/sans-digital-forensics-and-incident-response-poster-released

Leskovec, J., Rajaraman, A., & Ullman, J. D. (2014). *Mining of massive datasets.* Cambridge University Press.

Levy, J. (2011). *Time is on my side.* Retrieved from https://drive.google.com/file/d/0B7mg0ZBnpGuOZjVlYjJmMWMtYTgyYy00OGVlLTkxNmYtZWM2YmJjNzc1Zjc0/view?ddrp=1&hl=en_US#

Li, Y., Sundaramurthy, S. C., Bardas, A. G., Ou, X., Caragea, D., Hu, X., & Jang, J. (2015). Experimental study of fuzzy hashing in malware clustering analysis. In *8th workshop on cyber security experimentation and test (cset 15).*

Liles, S., Rogers, M., & Hoebich, M. (2009). A survey of the legal issues facing digital forensic experts. In *Advances in digital forensics v* (pp. 267–276). Springer.

Lim, S., Park, J., Lim, K.-s., Lee, C., & Lee, S. (2010, Aug). Forensic artifacts left by virtual disk encryption tools. In *Human-centric computing (humancom), 2010 3rd international conference on* (p. 1-6). doi: 10.1109/HUMANCOM.2010.5563320

MAGNET. (2015). *Artifact lists.* Retrieved from https://www.magnetforensics.com/magnet-ief/

Malafsky, G. P., & Newman, B. D. (2009). *Organizing knowledge with ontologies and taxonomies* (Tech. Rep.). TECHi2. Retrieved from http://www.techi2.com/download/Malafsky%20KM%20taxonomy_ontology.pdf

Manber, U. (1994). Finding similar files in a large file system. In *Proceedings of the usenix winter 1994 technical conference on usenix winter 1994 technical conference* (pp. 1–10).

Martínez, V. G., Álvarez, F. H., & Encinas, L. H. (2014). State of the art in similarity preserving hashing functions. *worldcomp-proceedings.com*.

Martínez, V. G., Álvarez, F. H., Encinas, L. H., & Ávila, C. S. (2015). A new edit distance for fuzzy hashing applications. In *Proceedings of the international conference on security and management (sam)* (p. 326).

Mee, V., Tryfonas, T., & Sutherland, I. (2006). The windows registry as a forensic artefact: Illustrating evidence collection for internet usage. *Digital Investigation*, *3*(3), 166 - 173. Retrieved from http://www.sciencedirect.com/science/article/pii/S1742287606000946 doi: http://dx.doi.org/10.1016/j.diin.2006.07.001

Merriam-Webster Dictionary. (2015). *Definition of artifact.* Retrieved from http://www.merriam-webster.com/dictionary/artifact

Mickelberg, K., Schive, L., & Pollard, N. (2014, June). *Us cybercrime: Rising risks, reduced readiness (key findings from the 2014 us state of cybercrime survey)* (Survey). PricewaterhouseCoopers LLP. Retrieved from http://www.pwc.com/en_US/us/increasing-it-effectiveness/publications/assets/2014-us-state-of-cybercrime.pdf

MITRE Corporation. (2015, Sept). *Cyber observable expression (cybox).* Github. Retrieved from https://cyboxproject.github.io/

Moore, J., Baggili, I., Marrington, A., & Rodrigues, A. (2014). Preliminary forensic analysis of the xbox one. *Digital Investigation*, *11, Supplement 2*, S57 - S65. Retrieved from http://www.sciencedirect.com/science/article/pii/S1742287614000577 (Fourteenth Annual {DFRWS} Conference) doi: http://dx.doi.org/10.1016/j.diin.2014.05.014

Mutawa, N. A., Baggili, I., & Marrington, A. (2012). Forensic analysis of social networking applications on mobile devices. *Digital Investigation*, *9, Supplement*, S24 - S33. Retrieved from http://www.sciencedirect.com/science/article/pii/S1742287612000321 (The Proceedings of the Twelfth Annual {DFRWS} Conference12th Annual Digital Forensics Research Conference) doi: http://dx.doi.org/10.1016/j.diin.2012.05.007

Nance, K., Hay, B., & Bishop, M. (2009). Digital forensics: defining a research agenda. In *System sciences, 2009. hicss'09. 42nd hawaii international conference on* (pp.

1–6).

Neuner, S., Mulazzani, M., Schrittwieser, S., & Weippl, E. (2015). Gradually improving the forensic process. In *Availability, reliability and security (ares), 2015 10th international conference on* (pp. 404–410).

NIST Information Technology Laboratory. (2013). *National Software Reference Library.* (http://www.nsrl.nist.gov)

Oliver, J., Cheng, C., & Chen, Y. (2013). Tlsh–a locality sensitive hash. In *4th cybercrime and trustworthy computing workshop (ctc)* (pp. 7–13).

Oliver, J., Forman, S., & Cheng, C. (2014). Using randomization to attack similarity digests. In *Applications and techniques in information security* (pp. 199–210). Springer.

Oxford Dictionaries. (2015). *Definition of artifact.* Retrieved from http://www.oxforddictionaries.com/us/definition/american_english/artifact

Payer, M., Crane, S., Larsen, P., Brunthaler, S., Wartell, R., & Franz, M. (2014). Similarity-based matching meets malware diversity. *arXiv preprint arXiv:1409.7760*.

Poli, R., Healy, M., & Kameas, A. (2010). *Theory and applications of ontology: Computer applications.* Springer. Retrieved from https://books.google.com/books?id=lQS6Abf9wzwC&pg=PA1&lpg=PA1&dq=big+o+little+o+ontology&source=bl&ots=n6DzTFqKaP&sig=Ue_rYSrJWGEI3XBRsSyawv1PvPU&hl=en&sa=X&ved=0CB0Q6AEwAGoVChMIzr7s69SuyAIVwz4-Ch2Y1wLw#v=onepage&q=big%20o%20little%20o%20ontology&f=false

Potthast, M., Barrón-Cedeño, A., Stein, B., & Rosso, P. (2011). Cross-language plagiarism detection. *Language Resources and Evaluation*, *45*(1), 45–62.

Psaroudakis, I., Katos, V., Saragiotis, P., & Mitrou, L. (2014, November). A method for forensic artefact collection, analysis and incident response in environments running session initiation protocol and session description protocol. *Int. J. Electron. Secur. Digit. Forensic*, *6*(4), 241–267. Retrieved from http://dx.doi.org/10.1504/IJESDF.2014.065737 doi: 10.1504/IJESDF.2014.065737

Rabin, M. O. (1981). *Fingerprinting by random polynomials* (Tech. Rep. No. TR1581). Cambridge, Massachusetts: Center for Research in Computing Technology, Harvard University.

Rajaraman, A., & Ullman, J. D. (2012). *Mining of massive datasets*. Cambridge: Cambridge University Press.

Rathgeb, C., Breitinger, F., & Busch, C. (2013, June). Alignment-free cancelable iris biometric templates based on adaptive bloom filters. In *Biometrics (icb), international conference on* (p. 1-8). doi: 10.1109/ICB.2013.6612976

Rathgeb, C., Breitinger, F., Busch, C., & Baier, H. (2013). On the application of bloom filters to iris biometrics. *IET Biometrics*.

Regional Computer Forensics Laboratory, U.S. Department of Justice. (2012). *Regional computer forensics laboratory program annual report* (Tech Report). Retrieved from https://www.rcfl.gov/downloads/documents/2012-rcfl-national-report

Regional Computer Forensics Laboratory, U.S. Department of Justice. (2013). *Regional*

computer forensics laboratory annual report for fiscal year 2013 (Tech Report). Retrieved from https://www.rcfl.gov/downloads/documents/fiscal-year-2013

Reith, M., Carr, C., & Gunsch, G. (2002). An examination of digital forensic models. *International Journal of Digital Evidence*, *1*(3), 1–12.

Ren, L., & Cheng, R. (2015, August). *Bytewise approximate matching, searching and clustering.* DFRWS USA.

Rivest, R. (1992). The MD5 Message-Digest Algorithm.

Roberts, S. J. (2013, Oct). *Osx auditor.* Github. Retrieved from https://github.com/jipegit/OSXAuditor

Rogers, M. K., & Seigfried, K. (2004). The future of computer forensics: a needs analysis survey. *Computers & Security*, *23*(1), 12–16.

Rosiello, A. P., Kirda, E., Ferrandi, F., et al. (2007). A layout-similarity-based approach for detecting phishing pages. In *Security and privacy in communications networks and the workshops, 2007. securecomm 2007. third international conference on* (pp. 454–463).

Roussev, V. (2009). Building a better similarity trap with statistically improbable features. In *System sciences, 2009. hicss '09. 42nd hawaii international conference on* (pp. 1–10). doi: 10.1109/HICSS.2009.97

Roussev, V. (2010). Data fingerprinting with similarity digests. In K.-P. Chow & S. Shenoi (Eds.), *Advances in digital forensics vi* (Vol. 337, pp. 207–226). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-15506-2_15 doi: 10.1007/978-3-642-15506-2_15

Roussev, V. (2011, August). An evaluation of forensic similarity hashes. *Digital Investigation*, *8*, 34–41. Retrieved from http://dx.doi.org/10.1016/j.diin.2011.05.005 doi: 10.1016/j.diin.2011.05.005

Roussev, V. (2012). Managing terabyte-scale investigations with similarity digests. In G. Peterson & S. Shenoi (Eds.), *Advances in digital forensics viii* (Vol. 383, pp. 19–34). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-33962-2_2 doi: 10.1007/978-3-642-33962-2_2

Roussev, V., III, G. G. R., & Marziale, L. (2007, September). Multi-resolution similarity hashing. *Digital Investigation*, *4*, 105–113. doi: 10.1016/j.diin.2007.06.011

Roussev, V., Richard, I., Golden, & Marziale, L. (2008). Class-aware similarity hashing for data classification. In I. Ray & S. Shenoi (Eds.), *Advances in digital forensics iv* (Vol. 285, pp. 101–113). Springer US. Retrieved from http://dx.doi.org/10.1007/978-0-387-84927-0_9 doi: 10.1007/978-0-387-84927-0_9

Rowe, N. C. (2012). Testing the national software reference library. *Digital Investigation*, *9*, S131–S138.

Ruan, K., Carthy, J., Kechadi, T., & Baggili, I. (2013). Cloud forensics definitions and critical criteria for cloud forensic capability: An overview of survey results. *Digital Investigation*, *10*(1), 34–43.

Sadowski, C., & Levin, G. (2007, December). *Simhash: Hash-based similarity detection.* http://simhash.googlecode.com/svn/ trunk/paper/SimHashWithBib.pdf.

Saleem, S., Baggili, I., & Popov, O. (2014). Quantifying relevance of mobile digital evidence as they relate to case types: A survey and a guide for best practices.

Journal of Digital Forensics, Security and Law, *9*(3), 19–50.

Sandvik, R. (2013). *Forensic analysis of the tor browser bundle on os x, linux, and windows* (Tech. Rep.). Tor Tech Report. Retrieved from https://research.torproject.org/techreports/tbb-forensic-analysis-2013-06-28.pdf

Security, H. N. (2013, July). *The rise of sophisticated malware.* Retrieved from http://www.net-security.org/malware_news.php?id=2543

Seo, K., Lim, K., Choi, J., Chang, K., & Lee, S. (2009). Detecting similar files based on hash and statistical analysis for digital forensic investigation. In *Computer science and its applications, 2009. csa '09. 2nd international conference on* (p. 1-6). doi: 10.1109/CSA.2009.5404198

Srinivasan, S. (2013). Digital forensics curriculum in security education. *Journal of Information Technology Education*, *12*.

Stauffer, C., & Grimson, E. (2001). Similarity templates for detection and recognition. In *Computer vision and pattern recognition, 2001. cvpr 2001. proceedings of the 2001 ieee computer society conference on* (Vol. 1, pp. I–I).

Stein, B. (2007). Principles of hash-based text retrieval. In *Proceedings of the 30th annual international acm sigir conference on research and development in information retrieval* (pp. 527–534).

Steinebach, M. (2012). Robust hashing for efficient forensic analysis of image sets. In *Digital forensics and cyber crime* (Vol. 88, pp. 180–187). Springer.

Stirparo, P. (2015). *Mac4n6 osx and ios artifact collection.* Retrieved from https://github.com/pstirparo/mac4n6

SWGDE/SWGIT. (2015, May). (2.8 ed.) [Computer software manual]. Retrieved from https://www.swgde.org/documents/Current%20Documents/2015-05-27%20SWGDE-SWGIT%20Glossary%20v2.8

Symantec. (2010). *Machine learning sets new standard for data loss prevention: Describe, fingerprint, learn* (Tech. Rep.). Symantec Corporation.

Thomson, A. (2012). *Windows 8 forensic guide* (Tech. Rep.). The George Washington University. Retrieved from http://www.bestcourse4u.com/wp-content/uploads/2015/Microsoft%20win/Windows-8-Forensic-Guide.pdf

Tridgell, A. (2002–2009). *spamsum.* http://www.samba.org/ftp/unpacked/junkcode/spamsum/. Retrieved from http://samba.org/ftp/unpacked/junkcode/spamsum/README (last accessed 2017-05-02)

Walnycky, D., Baggili, I., Marrington, A., Moore, J., & Breitinger, F. (2015). Network and device forensic analysis of android social-messaging applications. *Digital Investigation*, *14, Supplement 1*, S77 - S84. Retrieved from http://www.sciencedirect.com/science/article/pii/S1742287615000547 (The Proceedings of the Fifteenth Annual {DFRWS} Conference) doi: http://dx.doi.org/10.1016/j.diin.2015.05.009

Walters, R. (2014, October 27). *Cyber attacks on u.s. companies in 2014.* Retrieved from http://www.heritage.org/research/reports/2014/10/cyber-attacks-on-us-companies-in-2014

Waziri, I., & Sitarz, R. (2015). *Cyber forensics: The need for an official governing body* (Tech Report). Center for Education and Research in Information Assurance and

Security. Retrieved from https://www.cerias.purdue.edu/assets/pdf/bibtex_archive/2015-4.pdf

Winter, C., Schneider, M., & Yannikos, Y. (2013). F2s2: Fast forensic similarity search through indexing piecewise hashsignatures. *Digital Investigation*, *10*(4), 361–371. doi: http://dx.doi.org/10.1016/j.diin.2013.08.003

Yasin, M., & Abulaish, M. (2013). Digla - a digsby log analysis tool to identify forensic artifacts. *Digital Investigation*, *9*(34), 222 - 234. Retrieved from http://www.sciencedirect.com/science/article/pii/S1742287612000837 doi: http://dx.doi.org/10.1016/j.diin.2012.11.003

Yasin, M., Cheema, A. R., & Kausar, F. (2010). Analysis of internet download manager for collection of digital forensic artefacts. *Digital Investigation*, *7*(12), 90 - 94. Retrieved from http://www.sciencedirect.com/science/article/pii/S1742287610000575 doi: http://dx.doi.org/10.1016/j.diin.2010.08.005

Yasin, M., Wahla, M., & Kausar, F. (2009, Sept). Analysis of download accelerator plus (dap) for forensic artefacts. In *It security incident management and it forensics, 2009. imf '09. fifth international conference on* (p. 142-152). doi: 10.1109/IMF.2009.11

Yelp. (2016, Jan). *Osx collector.* Github. Retrieved from http://yelp.github.io/osxcollector

Zauner, C. (2010). *Implementation and benchmarking of perceptual image hash functions* (Unpublished master's thesis). University of Applied Sciences Upper Austria.

Zax, R., & Adelstein, F. (2009). Faust: Forensic artifacts of uninstalled steganography tools. *Digital Investigation*, *6*(12), 25 - 38. Retrieved from http://www.sciencedirect.com/science/article/pii/S1742287609000267 doi: http://dx.doi.org/10.1016/j.diin.2009.02.002

Zhang, T., Tang, Y. Y., Fang, B., & Xiang, Y. (2012). Document clustering in correlation similarity measure space. *IEEE Transactions on Knowledge and Data Engineering*, *24*(6), 1002–1013.

Zhou, W., Zhou, Y., Jiang, X., & Ning, P. (2012). Detecting repackaged smartphone applications in third-party android marketplaces. In *Proceedings of the second acm conference on data and application security and privacy* (pp. 317–326).

Ziroff, G. (2012). *Approaches to similarity-preserving hashing, Bachelor's thesis, Hochschule Darmstadt* (Unpublished master's thesis). Hochschule Darmstadt.