A NEW PORTABLE DIGITAL FORENSICS CURRICULUM

BY

ANTHONY JAMES LANG

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Adviser:

Professor Roy Campbell

ABSTRACT

This thesis describes the design, development, and resulting curriculum materials of a new introductory course in digital forensics. This course is part of a new certificate program in digital forensics at the University of Illinois at Urbana-Champaign, along with an advanced digital forensics course and laboratory exercises that are currently under development. We are designing these courses from the ground up to reflect the multidisciplinary nature of digital forensics, by incorporating the knowledge of a curriculum development team including domain experts from the fields of computer science, law, social science, psychology, and accounting. To lower the entry barrier for institutions to adopt digital forensics programs, we are designing the curriculum with the express intent of distributing it as a self-contained curriculum package including everything needed to teach the course. At the time of writing, we have taught a pilot class for the introductory course and revised the curriculum based on our experiences and feedback from the students. In addition to outlining our program's progress and high-level goals, this thesis presents the introductory course curriculum in narrative form, for those modules for which I was the primary author, and a brief summary of those modules for which I was a coordinating assistant.

To my fiancé Holly, for her endless patience, boundless love, and unwavering support.

ACKNOWLEDGMENTS

I would like to acknowledge the attendees of the 2013 DFCS workshop for their engaging discussions and invaluable feedback. This work was supported by the National Science Foundation under Grant No. [DUE-1241773]. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

As we increasingly rely on digital devices in almost every aspect of our daily lives, these devices are becoming increasingly involved in legal investigations of all kinds. Digital forensics (DF) is the science of identifying, collecting, preserving, documenting, examining, analyzing, and presenting evidence from computers, networks, and other electronic devices. For our purposes, I interpret this to subsume the disciplines of computer forensics, network forensics, and mobile device forensics. DF is now a major part of many criminal and civil investigations; its tools are frequently used by law enforcement agencies and private labs for investigation, data recovery, and diagnostics. Although digital forensics has already assumed such an important role in our society, it is still a new and rapidly developing area of study. This presents a challenging position to the digital forensics education community, that this work proposes to assist with.

1.1 Literature Review

To begin to understand the current state of the digital forensics education community and the challenges facing it, I conducted a literature survey of current higher education programs in digital forensics. While many programs have curriculum descriptions available online, and these were considered in developing our own curriculum, such a brief listing does not reveal the important challenges and design decisions required to understand the state of the field. Consequently, I restricted my inquiry to those programs with a published description of their curriculum development in the literature.

Wassenaar et al. [2] describe the certificate program in digital forensics at Cypress College. They put a clear focus on training students in practical

Parts of this chapter appear in an article currently under review for publication [1].

skills to prepare them for professional certification. They also require their instructors to be digital forensics practitioners, and rely heavily on their personal experience to lend the program credibility, since there is no generally accepted curriculum model at the college level.

Chi et al. [3] describe their efforts to expand the existing computer forensics course at Florida A&M University, which was previously part of the Information Assurance program, into a cross disciplinary concentration shared with the department of Sociology and Criminal Justice. This paper explains their challenges in teaching computer forensics to students without a strong technical background. Their solution was to create several remedial prep courses to get the Sociology and Criminal Justice students the prerequisite knowledge they need to enter the computer forensics concentration courses; in the process, they shifted the curriculum's focus much more toward training in practical skills to prepare students for professional certification.

Srinivasan [4] describes the computer forensics course at the University of Louisville. They cover a great deal of material, but their target demographics are restricted to students in the Information Security concentration in their Computer Information Systems program.

In [5] and [6], Liu describes the development process for Metropolitan State University's baccalaureate program in digital forensics. They employed a "backwards design" or "practitioner's model" to build the topic list for their curriculum. Basically, they looked at the needs of their students' target industry, clustered them into knowledge groups, and derived topic lists from these groups. Since they implemented an entire baccalaureate program in digital forensics, they were able to build the students' required prerequisite knowledge for the digital forensics courses and also cover the theory behind what they were learning. However, one of the most striking things about this work is the author's description of all the difficulties they had to overcome, particularly finding qualified faculty, that illustrate the huge entry barrier facing institutions that want to adopt digital forensics curricula.

These accounts concur with my impression from personal conversations with digital forensics educators and from inspecting online curriculum listings that most digital forensics programs currently have either training based courses taught by practitioners that teach students how to use a tool and follow a procedure, or courses that cover theory but restrict the student demographics to Computer Science or similar majors.

1.2 Curriculum Standards

Establishment of a standardized curriculum for digital forensics is important for several reasons. Principally, it provides a means for employers to validate the qualifications of a recent graduate from a digital forensics program. If the student graduated from a program using the standard curriculum, employers can immediately assess the minimum skill set that candidate is likely to have, without the need of additional evaluations. Similarly, as digital forensics graduates may serve as expert witnesses in legal proceedings, courts would also benefit from the added assurance of their expert's credentials. From the point of view of a perspective student, standardized curriculum gives the dual benefit of simplifying the evaluation of degree options and of increasing the employability of those degrees, for the aforementioned reasons.

These observations are by no means novel, and there have been concerted efforts from the digital forensics education community to establish standardized curriculum in the past. Most recently, the American Academy of Forensic Sciences' (AAFS) Forensic Science Education Programs Accreditation Commission (FEPAC) published, and offers accreditation based on, a standard that includes digital forensics [7]. However, at the time of writing, only a single university has adopted this standard and received their accreditation for digital forensics [8].

We organized and hosted a workshop in the spring of 2013 to facilitate a dialog among leaders in the digital forensics research, education, and professional communities about goals for a curriculum standard and roadblocks to widespread adoption of such a standard. Different stakeholders presented their opinions and needs for various aspects of a curriculum standard and gave their perspectives on what is preventing development and adoption of curriculum standards in digital forensics.

The discussions at the workshop generated as many questions as answers. For example, "What prerequisites should be required?", "What department should host the program?", and "What entity should publish the standard?" This may not seem like progress, but the questions themselves are informative. The issues presented and questions asked by the attendees of the workshop indicated that the primary barriers to adoption of curriculum standards are not pedagogical, but practical. In other words, the main problem with previously proposed standards was not the topic coverage, but the fact

that they were difficult to implement at most institutions.

1.3 Challenges

Based on input from digital forensics educators at our workshop, our own experience, and a review of the literature, I have compiled a list of the principal challenges facing institutions wishing to implement digital forensics programs:

- **Balancing training and education**

  Demand for continuing professional education and certification has led to development of training based courses that teach digital forensics as a stepwise laboratory procedure, and neglect to educate students in the theoretical foundations of what they are learning [9], [10]. The same pressure is put on many applied disciplines, but it is easier to resist in more well established fields, such as computer science, because there is a tradition of higher education providing a balance of skills and theory, leaving some training to the employer. This poses a significant problem to institutions interested in providing their students with a strong theoretical background in their digital forensics program.

- **Lack of an adequate textbook on digital forensics**

  Existing books on digital forensics are mostly written as handbooks for practitioners, containing useful tips and general information about best practice based on the authors' personal experience [5]. While these contributions are valuable, they offer very little explanation of the underlying technology or discussion of the theory for the topics, so are insufficient as textbooks for a course in higher education.

- **Finding qualified faculty**

  Given the absence of a standard curriculum and adequately detailed textbook resources to teach from, digital forensics training and education must rely heavily on the personal experience of the instructor [10], [5]. This is particularly problematic given the scarcity of qualified digital forensics professionals.

- **Lab setup**

  Licenses for proprietary digital forensics software tools and specialized hardware can be prohibitively expensive; even assuming you have lab exercises planned, installing and configuring equipment for a digital forensics lab is no easy task [10], [5].

- **Selecting appropriate prerequisites**

  Since digital forensics is essentially an application area at the intersection of computer science and law, it has natural prerequisite knowledge from those fields. However, since digital forensics students are very unlikely to be double majoring in Computer Science and Law, the question of which prerequisites to require and which to include in the digital forensics curriculum becomes quite difficult. Most existing programs opt to require substantial technical prerequisites. This enables them to easily focus their curriculum on the topics they see fit, but it restricts their curriculum's target demographics significantly [5], [3]. Where to draw the line on this trade off was one of the most hotly debated issues at our workshop, and one that we found particularly challenging in our own curriculum development.

- **Lack of widely accepted curriculum standards**

  Although proposed curriculum standards exist for digital forensics, there is no generally accepted model [7], [11], [12], [13]. This directly contributes to institutions' problems adopting a digital forensics program, by increasing the uncertainty of decision makers and the difficulty of curriculum development. It also contributes indirectly by exacerbating the other difficulties as described above.

1.4 Our Program

In this section, I outline our program's high-level goals, overall direction, and motivating rationale.

To help address the needs of the digital forensics research, education, and professional communities detailed in section 1.3, and the broader social need

for qualified digital forensics practitioners, we are developing a new undergraduate certificate program in digital forensics.

To lower the entry barrier facing institutions that wish to adopt a digital forensics program, we are developing our curriculum with the express intent of distributing it as a self-contained curriculum package containing everything a Computer Science professor will need to teach the course, including an instructor handbook detailing the course content, a lab instructor handbook explaining the lab exercises, PowerPoint slide decks for all lectures, remedial resources (such as reading lists) for the benefit of students from less technical backgrounds, and question sets to be drawn from for homeworks and exams. To the best of our knowledge, this will be an unprecedented contribution to the digital forensics educational community.

To create a curriculum in line with the fundamentally interdisciplinary nature of the field of digital forensics, we assembled a curriculum development team that includes domain experts in computer security, computer networks, law, civil and criminal justice, fraud investigation, and psychology. We take a modular approach to curriculum development, with domain experts taking the lead in developing and teaching topical modules focused on their areas of expertise. These modules are combined to form a coherent narrative to expose students to the many important perspectives on digital forensics.

When complete, our program will consist of an introductory and advanced course in digital forensics with accompanying hands-on laboratory sessions. The introductory course should be accessible to a wide range of students from many disciplines and valuable as a standalone offering, if the students do not choose to take the advanced course as well. The advanced course is still intended to be accessible to students from many disciplines, but will target students intending to go into digital forensics professionally, so will be more technically intensive.

This program will not be a job-track training program intended to prepare students to directly enter the job market as digital forensic examiners and analysts. Instead, it will provide a broadly applicable education in the field of digital forensics that will be valuable for students going into many disciplines related to digital forensics, such as law, in addition to forensic examiners and analysts. It is expected that these students will receive additional education specific to their career paths and some on-the-job training specific to their eventual professional roles.

At the time of writing, we have developed the curriculum for our introductory course and taught a pilot class that was cross listed in Computer Science and Law. The curriculum for the advanced course is currently under development.

1.4.1 Program Goals

Several high-level goals have guided the development of our curriculum.

- **Lower entry barrier for new institutions to adopt digital forensics programs**

  As described above, the primary problems facing institutions interested in adopting a digital forensics curriculum are finding a qualified instructor, difficulty and expense of setting up a lab, finding and selecting an appropriate textbook, balancing training and education, selecting prerequisites, and the lack of a widely accepted standard curriculum.

  Our curriculum is designed to be easily adoptable by other universities and colleges. So, we made two key decisions to address these problems. First, the curriculum package will contain everything a Computer Science professor will need to teach the course. We believe that given sufficiently detailed background material (as provided by our handbook), the instructor will not need to have industry experience practicing digital forensics. Second, the laboratory will not require the purchase of any specialized hardware or software licenses; since the lab exercises in our curriculum require only open source, freeware tools, the difficulty and expense of setting up a lab are reduced. Since our handbook can be distributed to students in place of a textbook, and adoption of a packaged curriculum removes an institution's need to balance training and education or select prerequisites.

- **Work toward curriculum standardization**

  Developing a curriculum standard for digital forensics is a challenging problem, but we believe we can contribute by distributing our curriculum as an easily adoptable and widely applicable option. We presented a draft of the curriculum for our introductory course to the

attendees of the 2013 workshop, and received very positive feedback and approval for the course content and modular, interdisciplinary design. We intend to continue revising and improving our curriculum based on feedback from the digital forensics research and education communities, our students, and future technological developments. Thus we can simultaneously break down the barriers to adoption (by solving additional logistic issues we hear about) and improve our curriculum to more closely match the digital forensics community's requirements for a curriculum standard.

- **Provide students with an education-based introduction to the field of digital forensics**

  We believe students should understand the theory behind what they are doing, not just how to do it. University graduates are marketable professionals, not because they know how to perform standard techniques better than a candidates with a training-based educations, but because their deeper understanding of the principles and theory underlying those techniques enables them to adapt and innovate when presented with new problems. Such students will contribute to the field of digital forensics, not just by performing sound forensic examination and analysis, but by improving standard practices and finding better solutions to problems. To this end, we designed our curriculum with a focus on knowledge and deeper understanding, rather than memorization of procedures and standard practices.

- **Develop a curriculum that reflects the fundamentally interdisciplinary nature of digital forensics**

  Many different disciplines have important perspectives on digital forensics. For example, a lawyer, computer scientist, and psychologist have fundamentally different perspectives on digital forensics. The lawyer sees it as a way to strengthen his or her case; a computer scientist sees it as a way of understanding and manipulating computers; and a psychologist sees it as a way of understanding the criminal mind. Our curriculum is intended to provide students the benefit of those diverse perspectives by having instructors from relevant disciplines develop and teach interdisciplinary modules. Rather than just teach students how

to work in a crime lab examining hard drives, we discuss how digital forensics skills can be applied to diverse practices. For example, in network intrusion response, investigations are often focused not on legal recourse, but damage assessment and mitigation and improving defenses for the future. These investigations are then fundamentally different from criminal investigations, because the evidence does not have to be court admissible, only acceptable to the leaders of the victim organization, and the identity of the attacker is of secondary importance. We also introduce students to other application areas, such as fraud investigation, to demonstrate the breadth of application for digital forensics knowledge.

- **Make the curriculum accessible and useful to a broad demographic from multiple disciplines**

  We believe that a course on digital forensics would be a valuable addition to many students' education, even if they do not intend to become digital forensics practitioners. There are many practical topics in our curriculum that are important to everyone. For example, most students have little or no knowledge of the legal justice system or laws related to computer crime. Also, knowing what evidence is left behind by computer and Internet activities can inform better practices, and some of our students enrolled principally for this reason. After Computer Science students, we found that the second largest demographic in our pilot class was Law students, who were interested in digital forensics so they could better understand and utilize digital evidence in their cases.

  Design of a digital forensics curriculum that is accessible to such broad demographics is a particularly difficult problem that we have not entirely solved. There are two issues that must be addressed to this end: prerequisites and technical difficulty. The problem is to make the course accessible without reducing the value for Computer Science students. We discuss our efforts in this area in the next section.

1.5 Curriculum Development

This section describes our initial curriculum development efforts, the lessons we learned from teaching the pilot class and our students' feedback, and the revisions we have made based on this experience.

1.5.1 Pilot Class

In the fall of 2013, we taught a pilot class of our introductory course. The class consisted of two 75-minute lecture sessions and an hour-long lab session weekly for a full 16 week term.

The introductory course was designed to give students from a wide range of disciplines an introduction to the field of digital forensics, focused particularly on the sub-disciplines of computer forensics, network forensics, and mobile device forensics and providing relevant interdisciplinary perspectives.

It is difficult to give an introduction to a field as broad as digital forensics in a single class, but we were able to cover the major sub-disciplines and introduce many interdisciplinary perspectives in part because of our focus on education rather than training. Memorization of standard practices and procedures would be time consuming, so by removing it, we were able to increase the breadth of topics in the introductory course. To reduce the computer forensics module to a manageable size, we chose to focus on NTFS and Windows forensics, as these are the systems students will be most familiar with and most likely to encounter in practice.

To help us develop an initial working list of topics for our introductory course, we started by compiling a list of all the topics from all the courses and recommended curriculum lists we could find, de-duplicating them, and then organizing them into modules. Within each module, we selected what we believed were the most important key concepts that could fit into the time slots available across a semester. To develop the curriculum for these topics, we started by referencing various textbooks recommended online or used by other institutions for their digital forensics classes. However, we found (as did [5]) that most of them were basically handbooks focused only on industry practice based on the authors' personal experience. For those topics for which we were not able to find adequately detailed textbook references, we gathered the required details from research papers, technical reports, and

other sources. This had the added benefit of necessitating that we find the most up-to-date source material available. In addition to these resources, we incorporated our interdisciplinary curriculum development team's domain expertise into the relevant modules. Table 1.1 shows a brief topic list for each module in our pilot class.

The issue of prerequisites was also difficult to resolve, since, even at an introductory level, digital forensics is a technically challenging topic that a student without a strong technical background would have great difficulty understanding. Simply requiring an operating systems and a networking course would have been sufficient, but would have shut out the broad interdisciplinary demographics our program targets. Fortunately, the majority of the topics covered in such courses are not necessary background for a digital forensics course. For example, it is unnecessary to know how to design context switching and memory management modules to get started in our computer forensics module. All that is required is a high level of computer literacy and some basic familiarity with how an operating system works. Our solution was to require knowledge prerequisites rather than course prerequisites. The course required instructor permission to enroll, and this permission was only given after the instructor has a conversation with the student and was satisfied he or she had the necessary level of prerequisite knowledge.

1.5.2 Lessons Learned From Pilot Class

Teaching the pilot class for our new curriculum was a very illuminating experience for our curriculum development team. While it went quite smoothly in general, several unforeseen issues became apparent as the semester progressed.

1. Coordination between instructors/modules was a challenge. We found it difficult to maintain a good narrative flow between modules. In retrospect, it is clear that the modules could have been ordered more efficiently. Some modules had overlapping topics, and the relevance of some topics needed to be made more explicit.

2. We had differing understandings of the knowledge prerequisites among the professors. Consequently, some students enrolled in the course who

Table 1.1: Pilot Class Topic List, by Module

Forensics Concepts
Course outline and syllabus
Define digital forensics and its subfields
Evidence handling
Psychology
Psychology of cybercrime
Criminal profiling
Computer Forensics
Introduction to file systems
NTFS analysis
Deleted file recovery and file carving
Windows Registry, log files, link files, Recycle Bin
Web browser forensics, email forensics, EXIF
U.S. Legal System
Disputes, courtroom workgroup, attorneys
Judges, juries, legal process
Network Forensics
Networking fundamentals review
Network evidence acquisition
Protocol analysis, packet analysis, flow analysis
Application protocols, statistical flow analysis
Network intrusion detection and analysis
Law
Fourth Amendment: reasonable expectation of privacy
Warrant vs. subpoena, Federal Rules of Evidence
Privacy laws, computer crime laws
Fraud Examination
Introduction to fraud examination
Characteristics and skills of a forensic accountant
The nature and extent of fraud, Benford's Law
Mobile Device Forensics and Malware
Mobile device technology fundamentals
Mobile device evidence extraction and analysis
Mobile network evidence
Legal and ethical considerations of interception
Malware taxonomy, detection, and circumvention

did not have the necessary technical background to understand the material. There was a very wide range in levels of computer literacy. In fact, several of the students were unaware that they were enrolling in a course that required a high degree of computer literacy. This issue was especially visible during the hands-on lab exercises.

3. Our pilot class enrollment mainly consisted of Computer Science students and Law students. The Computer Science students, and some of the Law students, had little trouble following the technical material. However, most of the Law students had difficulties, even with tutoring. The students' wide range of computer literacy made it difficult to pace the exercises appropriately. Although the students who struggled were a small minority of our pilot class's enrollment, they represented the potential interdisciplinary demographics to which we want our curriculum to be accessible. So as the semester progressed, and these issues became apparent, we reevaluated our curriculum to devise a solution that would not shut out these students. We observed that many of our exercises naturally had both investigative/legal components and technical components. The Law students typically performed better on the investigative components (e.g. not jumping to conclusions about a piece of evidence), and the Computer Science students generally performed better on the technical components. The solution we implemented was twofold. First, we decided to put less focus on technical detail and more on investigative and evidentiary complexities. Second, we reworked some of the labs into group assignments in which Law students were required to partner with Computer Science students, We also wrote a team project where Law students partnered with Computer Science students and performed different roles; together they submitted a report on a fictitious case. Grouping them together allowed them to learn from each other, and we observed positive results, with students teaching each other their domain knowledge and putting their expertise together to better solve problems. The labs went visibly smoother, and both Computer Science and Law students commented that their experience was enriched by interacting with the other majors.

1.5.3 Student Feedback

The Science, Technology, Engineering and Mathematics Initiative (I-STEM) Office at our University was hired to conduct an evaluation of our digital forensics pilot class. The evaluators conducted surveys and focus groups to get student feedback [14]. The three surveys over the course of the semester ascertained the students' background, experience and impressions of the course, and suggestions for improvement. The focus groups involved a dialog between small groups of students and one of the evaluators, who prompted the students with topics to get the conversation started, but generally focused on providing an environment for the students to freely share their opinions of the course.

Feedback from the students in our pilot course was generally quite positive, with 80% of survey respondents agreeing or strongly agreeing that course objectives and content were thoroughly covered, and 93% agreeing or strongly agreeing that they were satisfied with what they learned, for the amount of time they invested in the course. In particular, students viewed the interdisciplinary aspect as a strength of the course, and said that having multiple instructors teach the course was useful, helpful, and exciting. This is apparent from their responses in free-form feedback and focus groups, and from the 70% of survey respondents who agreed or strongly agreed that having multiple instructors teach the course was helpful. The students also gave positive feedback for the group work and interactions with students from other disciplines. Their free-form survey responses and focus group comments show that they enjoyed the cross-disciplinary engagement, and they felt they learned from the other students. Specifically, 88% of survey respondents agreed or strongly agreed that the group assignment contributed to their learning.

In the free-form feedback, students also reported issues with several aspects of the course. Students felt that there was a lack of communication among the instructors, and that the topics felt out of place and did not fit with one another. They suggested using a single, longterm case study to connect the lectures from the beginning to the end of the semester. In addition, some Law students had difficulty with the technical terminology. They suggested that instructors provide a sort of glossary of terms for quick reference.

1.5.4 Revisions

We have revised the curriculum based on what we learned teaching the pilot class and feedback from the students, and are in the process of developing several additional items for the curriculum package.

We have changed the module ordering. Specifically, we will put the legal justice system and law modules before any of the technical material, as those modules present the wider social impact of digital forensics and how it is used in court respectively. This will make it clear to students *why* digital forensics is practiced before they learn *how* it is practiced.

We have extended the increased focus on investigative issues, analysis of evidence, and group activities that we successfully applied to the later portion of the course to the earlier portions as well. These group activities are not intended to *require* a legal background for any of the students, as we understand that it is likely that many institutions will not have students with such a background in their class. Rather, the activities will encourage students to look at the problems from a different perspective. Thus, students from diverse backgrounds will be more valuable to the group, and all the students will gain a broader understanding of the issues they are learning about.

To improve the flow of the course between modules and make the relevance of topics more explicit, we have written a fictitious case study that will run through the entire course. The story in the case study will advance as the semester progresses; new examples and assignments will be tied into the story to maintain students' interest and make the "big picture" clear. This case study will be distributed in the curriculum package, and adopting institutions may incorporate ours or their own case study and / or examples.

We are also developing three supplementary items for the curriculum package, to address the problem of varying computer literacy in students, without shutting out the interdisciplinary demographics.

First, we will compile a primer on technical fundamentals, to be made available to students before the first day of class. It will contain very brief explanations of fundamental concepts that are important as knowledge prerequisites, to refresh the memory of students who may have taken the relevant background classes some time ago, and will also contain references to more in-depth readings for students who are missing specific topics.

Second, we will provide a glossary of terminology for quick reference. This will be useful for students from diverse backgrounds who may be familiar with the concepts, but under different names.

Third, we will include a short prerequisite "quiz" that must be completed before students can receive the required instructor permission to enroll in the course. Considering the wide range of technical literacy we saw in our pilot course, a couple simple questions that would seem trivial to someone with a technical background, but very challenging to a typical layperson (e.g., "What is ASCII?") would suffice. The quiz should take no more than a couple minutes to complete, with the answers written or given verbally to the instructor, and would have two primary purposes. First, it would let students know what sort of course they are enrolling in from the start. Second, it would deny enrollment to students who simply do not have the background to understand even the primer material we will make available to them.

CHAPTER 2

INTRODUCTORY COURSE CURRICULUM

In this chapter, I present the revised course content, in narrative form, of those modules for which I was the principle author; namely Forensics Concepts, Computer Forensics, Mobile Device Forensics, and Malware Forensics; I present brief summaries of those modules for which I was only a coordinating assistant; namely Legal Justice System, Law, Psychology of Cybercrime, Network Forensics, and Fraud Investigation.

This course consists of 9 topical modules intended to give students a high level overview of the field of digital forensics, by presenting the different sub-disciplines and important interdisciplinary perspectives. The focus is on the underlying principles and theory governing digital forensics practice, rather than learning the details of standard practices as a stepwise laboratory procedure.

2.1 Module: Forensics Concepts

This section presents the material for the first lecture in the course in narrative form[1]. The purpose of this lecture is to give students a high level introduction to the basic principles of forensic science and forensic evidence in general, to put the remaining modules into context.

2.1.1 Lecture: Forensics Concepts

Before we begin our introduction to digital forensics, I want to define a few terms that do not have a universally accepted definition. For our purposes, I define forensics, digital forensics, and digital evidence as follows.

[1]This section is presented in narrative form, so the second person (e.g. "you") indicates the students and the first person (e.g. "I") indicates the instructor.

- *Forensics* is the application of science to legal problems and investigations.
- *Digital forensics* is a branch of Forensics involving the recovery and investigation of Digital Evidence.
- *Digital evidence* is data that is stored or transmitted using a digital device and is relevant to a legal investigation,

Thus, any kind of science used in support of a legal investigation is called a forensic science. For example, DNA analysis, blood spatter analysis, and ballistics are forensic sciences. When computer science is used to support legal investigation, it is called Digital forensics.

2.1.1.1 Sub-Disciplines of Digital Forensics

"Digital forensics" is an umbrella term covering several partially overlapping sub-disciplines.

- *Computer forensics* is the sub-discipline of digital forensics that deals with preservation, extraction, analysis, and investigation of digital evidence from individual computers.
- *Network forensics* is the sub-discipline of digital forensics that deals with preservation, extraction, analysis, and investigation of digital evidence from network components and servers.
- *Mobile device forensics* is the sub-discipline of digital forensics that deals with preservation, extraction, analysis, and investigation of digital evidence from mobile devices such as cell phones and GPS devices.

These branches are overlapping and distinguished mostly for convenience of explanation. For example, evidence about network events is generally considered part of network forensics, but web browser forensics is considered part of computer forensics. An actual investigation may require skills from multiple branches of digital forensics, as well as conventional forensics, such as DNA and fingerprint analysis. There is some debate as to whether multimedia forensics should be considered a distinct sub-discipline of digital forensics but in either case, we will not cover multimedia forensics in this course.

2.1.1.2 Scientific Method

We will go into more details of the investigative methodologies in the modules specific to types of investigations, but as a general principle of forensic science, investigations should follow the scientific method. It is important that forensic disciplines follow the scientific method because it helps to facilitate a more rigorous and ethical investigation, by requiring objective, logical proof rather than subjective assumptions. The overall objective is to logically deduce what happened and prove this conclusion accurate with appropriate confidence. Many of you are likely already familiar with the steps of the scientific method, so we will focus on how they are applied to a forensic investigation.

1. *Observation*

   This step basically encompasses everything that happens before you begin the digital forensics investigation. You will have a certain level of background knowledge about the case, who is suspected of doing what, the reason the investigation began etc.

2. *Form hypothesis*

   Before moving forward with the investigation, you must form, or revise, an investigative hypothesis. This is your proposed explanation of what happened, based on everything you have observed thus far. Selecting a good investigative hypothesis is one of the most important and difficult steps in a digital forensics investigation. A candidate investigative hypothesis should be evaluated according to these three criteria [15]:

   - *Falsifiability*: It must be possible to show the hypothesis to be false, by some feasible test. If a hypothesis is not falsifiable, then you cannot make rigorous scientific claims about its accuracy, and it should not be used as an investigative hypothesis.
   - *Consistency*: The hypothesis must be consistent with all previous observations.
   - *Simplicity*: Other considerations being equal, a simpler hypothesis should be preferred.

A full investigative hypothesis that explains all the observations will likely be too complex to be tested directly. Instead, you should break the hypothesis into sub-hypotheses that are easily testable and refutable and together can confirm or refute the full hypothesis.

3. *Prediction*

   After selecting or revising your investigative hypothesis, you should use it to predict observations (e.g. If H is true, I expect to find X). These are predictions of specific items you expect to find if your hypothesis is true. For example, if you hypothesis is that *Alice sent Bob confidential information over email*, you might predict that you will find emails containing confidential information from Alice in Bobs email inbox.

4. *Experimentation*

   Once you have an investigative hypothesis and specific predictions, you should conduct tests / experiments and compare the results of these experiments to your predictions to decide if they confirm or refute your hypothesis with acceptable confidence. If your experiments prove or refute the hypothesis, you move on to the next step: conclusion and reporting. If your experiments fail to prove or refute your hypothesis, you must return to the "Form hypothesis" step and revise your hypothesis based on your new observations, repeating the subsequent steps until you pass the tests.

5. *Conclusion*

   Once you have formed and proven an investigative hypothesis with an acceptable confidence range, you must draw conclusions for the wider investigation and generate your report. The type of conclusions you draw will depend on the nature of the investigation, which we will discuss in later modules as appropriate. We will learn some principles for best practice in writing digital forensics reports later in this lecture.

2.1.1.3 Testing and Experimentation Principles for DF

Designing good experimental tests for digital forensics can be difficult, but there are a few guidelines that can help.

The experiments should have a high chance of refuting or confirming the hypothesis. An experiment that is more likely to provide conclusive evidence should naturally be preferred to one with a high chance of providing evidence for which there are many possible explanations.

All tests and other elements of the experiment must be tractable, or feasible, given the resource and time constraints of the investigation. This will of course become obvious when you try to execute the experimental procedures, but it should be considered early in the design process to avoid wasted time.

The experiment should use techniques that are accepted as sound by the digital forensics academic community. This is a bit less obvious, since a technique does not become more sound just because people say it is. However, it is important to remember that your technique and results will be scrutinized by both sides in the pre-trial and courtroom procedures, and you must be careful to ensure that the credibility of your analysis is solid. This is important from the courts point of view, because they have no other way of evaluating the validity of your methods, which of course they must do.

The simplest and most common type of test would be examination of an evidentiary device with digital forensics tools and techniques looking for specific pieces of information. However, more interactive experimentation is also common. For example, configuring a dummy system to match the evidentiary system, carrying out various actions on the dummy system, and comparing the resulting state of the dummy system with the observed state of the evidentiary device.

2.1.1.4 Refutation

An important principle to ensure an objective forensic investigation is refutation. The principle of refutation is that tests to confirm a hypothesis should include attempts to refute the hypothesis. In formal mathematical systems, this concept is obvious and required for a valid proof, but it is easy to overlook in an investigation because of confirmation bias.

Confirmation bias is our natural tendency as humans to try to confirm our initial assumptions. We tend to design experiments that are more likely to confirm our initial hypothesis than to refute it, and this is very dangerous in a forensic analysis.

It is often easy to find evidence of a persons guilt, but there may be other

likely explanations for the evidence. This underlying problem with human nature makes the principle of "innocent until proven guilty" very important for forensic analysis. You should not conclude that a person is guilty unless all other viable explanations are refuted to a reasonable degree of confidence.[2]

2.1.1.5 Reporting and Testimony

Once your analysis is complete and you have drawn your investigative conclusions, you must generate your report for presentation to the lead investigator and potentially the court. The form and content of this report will differ significantly from that of a typical engineering or scientific report in a couple key respects.

First, the target must be a general lay audience. This differs from the "educated lay audience" commonly targeted by academic writing, which assumes general domain knowledge, missing only knowledge about the specific topic of the work. A general lay audience should be assumed to have no background knowledge outside a typical high school education, so the report should contain as little technical jargon as possible and have clear explanations of all technical concepts required to understand the evidence.

Second, your report should not state your actual investigative hypotheses and conclusions. Specifically, the report should not contain any explicit declaration of the guilt or innocence of any party. This may seem counterintuitive, since your investigative hypothesis and conclusion will likely clearly implicate or exonerate some parties, but it is the responsibility of judges and juries to decide guilt or innocence. As a forensic expert, your role is to provide and explain evidence. To make this intuitive, imagine your report as telling a story, but leaving the ending up to the readers imagination.

In both your report and expert testimony, it is important to avoid making any speculative statements. As a forensics expert, you should only make statements directly supported by evidence.

2.1.1.6 Circumstantial vs. Direct Evidence

A very important concept that you should be familiar with when carrying out an investigation is the distinction between circumstantial and direct evidence.

[2] I am indebted to Dr. Frank Nekrasz for this explanation

- Evidence from which you might infer that some event may have occurred, but for which there are other viable explanations, is *circumstantial evidence*.

- Evidence that directly shows that an event occurred, without the need for any inference, is *direct evidence*.

Another way of thinking about this is that circumstantial evidence implies that an event occurred, while direct evidence demonstrates that an event occurred. A single piece of circumstantial evidence is unlikely to convincingly support a fact in court, but if many pieces of circumstantial evidence can be corroborated to eliminate alternative explanations and draw the same conclusion, a court may accept the conclusion as fact.

This distinction is especially important for digital forensics because digital evidence is almost always circumstantial. Digital evidence typically directly links a computer to a specific action, but implicating a person requires corroborating evidence that the person was the one directing the computer to take that specific action. For example, systems logs, browser history, and saved files can be direct evidence that a particular account on a computer was used to access a specific website, but it is only circumstantial evidence that the owner of the computer accessed the website. However, an eye witness testifying that they saw the owner using the computer at the time the website was accessed would directly link the owner to the action of viewing the website.

2.1.1.7 Forensic Soundness

Before we continue, I would like to define a commonly used term: *forensic soundness*. The effective definition of forensic soundness is that a method is forensically sound if it is rigorously performed according to the best practices of its respective discipline. Thus, you could consider "forensically sound" as roughly equivalent to "properly done" or "best practice". This term is commonly used to indicate only a method that adheres to standard practices, but it is sometimes used to indicate a method that also adheres to the ideals of rigorous scientific method, and it is in the latter sense that I shall use it. What is considered forensically sound will of course depend on the task in

question, but there are a few general principles that should be followed to help ensure forensic soundness.

1. Every interaction with the evidence and every step in the analysis should be rigorously documented, such that another forensics expert with the same tools could exactly duplicate your analysis and achieve the exact same result. In fact, this is often done for particularly important elements of the analysis. We will talk more about this later in this lecture.

2. Every realistic precaution must be taken to ensure that the evidence is not damaged or otherwise altered from its original state during the acquisition, transportation, and analysis processes. If some alteration of the evidence is unavoidable, the exact nature of the change must be documented. We will discuss this at length in each module as we discuss specific techniques.

3. The analysis must be objective and unbiased. This requires the rigorous application of the scientific method, as we have already discussed, but should also serve as a guiding principle for every aspect of the investigation.

Note that "forensic soundness" differs subtly from "court admissible," which means "acceptable for presentation in court." Rules and laws for court admissibility will be discussed in more detail in the next two modules. It is very important to note that both forensic soundness and court admissibility are orthogonal to the distinction between circumstantial and direct evidence. It is perfectly possible, and very likely, that forensically sound analysis will result in circumstantial evidence that is court admissible. The distinction between circumstantial and direct evidence is not in whether the court will accept the evidence, but in what they do with it.

2.1.1.8 Evidence Integrity

One of the most important considerations in conducting a rigorous, forensically sound investigation is maintaining the integrity of the evidence. That is to say that damage or alteration of evidence should be avoided whenever

possible. Often some minor alterations to evidence are unavoidable during forensic analysis; in such cases, the exact nature of the changes must be documented. This is important, not only to ensure that evidence is not lost, but also to ensure that artifacts inadvertently introduced during analysis do not confuse the investigation. Forensic evidence is unlikely to be acceptable in court if its integrity cannot be attested to with comprehensive documentation.

The integrity of digital evidence is usually demonstrated using cryptographic hash functions. A *cryptographic hash function* is a function that takes arbitrary sized digital data as input and outputs a fixed size "hash value" with three required properties:

- *Collision resistance*: It is computationally infeasible to find two different inputs that produce the same output.
- *Pre-image resistance*: Given a hash value, it is computationally infeasible to find an input whose hash value would match it.
- *Second pre-image resistance*: Given an input, it is computationally infeasible to find a different input that will produce the same hash value.

Note that a cryptographic hash function is a specific type of hash function. Normal hash functions are simply functions that take a variable length input and output a fixed length output. A cryptographic hash has more rigorous requirements that make it suitable for computer security systems, and in our case, digital evidence.

A digital forensics analyst computes the cryptographic hash of the evidence as soon as it is received (hopefully immediately after acquisition). If this cryptographic hash value has been documented, then the integrity of the digital evidence can be verified at any time by recomputing the cryptographic hash function of the evidence and comparing it to the previously documented value.

Often evidence will be handled by multiple people before it makes it to the digital forensics examiner and its cryptographic hash value is computed. To ensure the integrity of evidence as it is handled by these various people, a complete *chain of custody* must be rigorously documented. The chain

of custody is the formal documentation of every individual who handled a piece of evidence. If there is a break in the chain (i.e. some transfer of the evidence between individuals was not accounted for), then the authenticity of the evidence can be disputed in court, and it may not be admissible.

2.1.1.9 Sources

The primary sources for this lecture's material were Eoghan Casey's textbooks: *Digital Evidence and Computer Crime* [16], [17], [18] and *Handbook of Digital Forensics and Investigation* [19], [20].

2.2 Module: Legal Justice System

In this section, I present a brief summary of the material from the two lectures covering the United States legal justice system. The instructor and subject matter expert for this module was Professor Anna-Maria Marshall.

The purpose of this module is to give students an introduction to the operation of the legal justice system in the United States, focusing on the different roles and procedures, to familiarize students with the context in which digital forensic evidence is used. This module presents the court's point of view on digital forensics, giving students an idea of how digital forensics practitioners interact with the system. This module will not go into specific laws related to digital forensics, which will be covered in the following module.

2.2.1 Key Concepts

- *Disputes*: The United States uses an *adversarial legal system*. Meaning that when disputes are referred to trial, the parties involved contest their points under the supervision of an impartial third party (a judge or jury). This is in contrast to the *inquisitorial legal system* used by many other countries, where the impartial third party leads the investigation directly.

  However, disputes are rarely referred to trial. Most are resolved by settlements negotiated by the *courtroom workgroup* without the need

for a costly and uncertain court proceeding.

- *Courtroom Workgroup*: The courtroom workgroup is the general term used to refer to those parties involved in the operation of the court. It principally consists of the prosecution and defense attorneys, the judge, the defendant and complainant, and various other roles that assist in the proceedings, such as clerks and bailiffs. Their combined inclusion in this "workgroup" implies their shared interest in the smooth and efficient operation of the court.

- *Civil vs. Criminal Court*: There are actually two distinct justice systems in the United States, each with separate jurisdiction over different types of disputes.

  The *civil justice system* handles disputes between individuals and/or corporate entities with the typical objective of obtaining some kind of material compensation. The injured party bringing their dispute to a civil court is called the *plaintiff*, and their opponent is called the *defendant*. Both are represented by their own attorneys, and the decision is based on a *preponderance of evidence*, which simply means whichever side is determined to be slightly more correct wins the case.

  The *criminal justice system* handles disputes between individuals and the State, where the end objective is to decide whether the individual is guilty of violating the law and, if so, to determine their punishment. It is important to note that the victim of the crime is not a party in the dispute, they are merely a witness, and the prosecuting attorneys represent the State, not the victim. In a criminal case, a conviction will only be decided if the guilt of the defendant can be demonstrated *beyond a reasonable doubt*. This constitutes a significantly higher standard of evidence than a civil case.

- *Juries*: In the United States, cases can be decided juries or judges, depending on the choice of the defendant. A jury consists of a group of ordinary citizens called at random to serve as an impartial third party to decide the final verdict of "guilty" or "not guilty" in the case.

- *Standards for Scientific Evidence (Daubert)* [21]: In order to be considered in a legal proceeding, scientific evidence must satisfy certain

standards (Sometimes called the Daubert test, after the case law that established them). The two basic requirements for scientific evidence are *relevance*, and *reliability*. Relevance is pretty self explanatory, but reliability is more subtle and difficult to demonstrate. To determine if evidence is reliable, five conditions are considered: testing, error rates, standards, acceptability, and peer reviews. The evidence must be the result of a falsifiable empirical test. That is to say, it must be possible for the test to have resulted in the opposite result. For example, if you want to test if a turkey is finished cooking, a method that always returns "finished" regardless of the state of the turkey is of little use. This method must result in quantifiable empirical error rates, it must be subject to standards governing its operation, it must have received widespread acceptance among the relevant scientific community, and it must have been subjected to peer review within that community.

2.3 Module: Law

In this section, I present a brief summary of the material from the four lectures covering legal principles and laws related to digital forensics. The instructor and subject matter expert for this module was Professor Jay Kesan.

The purpose of this module is familiarize students with the most important laws governing the seizure, examination, and presentation of digital evidence in the United States. This module presents a lawyer's point of view on digital forensics, giving students an idea of how digital evidence is used in a trial.

2.3.1 Key Concepts

- *The Fourth Amendment*: This module presents the functional interpretation of the Fourth Amendment that dictates the rules for lawful searches and seizures in the United States. The Fourth Amendment specifically protects people from *unreasonable* searches and seizures, but what exactly "unreasonable" means in this context is a matter of interpretation and case law.

  The functional interpretation of the Fourth Amendment and later case

law have resulted in the modern rules governing search warrants. To obtain a *Search warrant* an investigator must specify the specific places and things that will be seized and show *probable cause* that they will find evidence of wrongdoing.

There are several caveats to these rules however. The biggest is that the Fourth Amendment only applies if the individual has a *reasonable expectation of privacy* for the object being seized. For example, if someone leaves a notebook on a park bench, it is not reasonable for them to expect the contents of the notebook to remain private, so investigators would not need to obtain a warrant to seize the notebook, but if that same notebook was left in the owner's home, they could reasonably expect that its contents should remain private, and investigators would need to obtain a warrant to seize it.

- *Federal Rules of Evidence*: The restrictions on what evidence can and cannot be admitted to court are called the Federal Rules of Evidence. These rules are designed to prevent inappropriate evidence from being shown to a jury, that may introduce an illogical bias to their reasoning. This is important, because juries are selected at random from ordinary people, who are not expected to have had any legal training or education in logical deduction.

  A few of the more important rules:

  - *Relevance vs. Prejudice*: The probative value (value in finding the truth) of a piece of evidence must outweigh its potential prejudicial impact on the jury (the degree to which it is likely to illogically bias their reasoning).
  - *Hearsay*: Statements made outside of court that are brought up in an attempt to demonstrate the truth of their contents are considered hearsay, and are generally not admissible as evidence. There is an exception to this rule for business records that is very important to digital forensics. Records that are routinely kept by businesses are admissible to court.
  - *Rule 702*: This rule lays out the basic requirements for the opinion of an expert witness to be admitted. An expert's opinion is admissible as evidence in court if "(1) the testimony is based

upon sufficient facts or data, (2) the testimony is the product of reliable principles and methods, and (3) the witness had applied the principles and methods reliably to the case [22]." This rather vague requirement is clarified significantly by the Daubert rules for scientific evidence, already discussed in section 2.2 above.

- *Privacy Law*: This module also introduces some of the important laws that protect individual's privacy by restricting the actions investigators can take, specifically the Stored Communications Act (SCA), the Family Educational Rights and Privacy Act (FERPA), the Genetic Information Nondiscriminatory Act (GINA), the Health Insurance Portability and Accountability Act (HIPAA), and the Computer Fraud and Abuse Act (CFAA).

2.4 Module: Computer Forensics

This section presents the material for the computer forensics module in narrative form[3].

Computer forensics is the sub-discipline of digital forensics that deals with preservation, extraction, analysis, and investigation of digital evidence from individual computers. An actual investigation may require skills from multiple branches of digital forensics, as well as conventional forensics, such as DNA and fingerprint analysis. However, in this module we will focus on those digital forensics skills specifically related to individual computers.

This module will consist of 6 lectures intended to give you a high level overview of the field of computer forensics, with a focus on the principles underlying the techniques, rather than memorization of standard procedures. Consequently, we will present examples of important investigative practices, but rarely go into tool-specific detail.

The lecture topics in this module are: Introduction to Computer Forensics, Introduction to File System Forensics, NTFS Analysis, File Carving, Windows Analysis, and Windows Application Analysis.

[3]This section is presented in narrative form, so the second person (e.g. "you") indicates the students and the first person (e.g. "I") indicates the instructor.

2.4.1 Lecture: Introduction to Computer Forensics

To give you some context for the techniques you will be learning in this module, I will first give you a brief overview of the high level steps in an investigation that uses computer forensics. The remainder of the lecture will be focused on the "preservation" step of this computer forensics investigative process, beginning with a brief review of some technical fundamentals, followed by discussion of forensic duplication.

2.4.1.1 Computer Forensics Investigation [18]

Here we present computer forensics investigation as a sequence of discrete steps, for ease of understanding. In practice, investigations are not likely to follow such cut and dry steps. As the investigation progresses and new facts become available, some steps may be repeated, and elements of some steps may be interposed into other stages as well.

While computer forensics skills can be employed in a variety of contexts, this formulation specifically refers to the computer forensics process for a physical crime scene with digital evidence.

1. *Preparation*: It is important to know what to expect at the scene before you head out, so you can get the proper items approved for seizure in a warrant and bring the appropriate hardware and personnel with proper training for the subsequent steps.

2. *Survey*: Once you arrive at the scene, you should survey the evidence before touching anything; Take note of all the potential evidence sources, physical and digital, so you can make the best decisions about what evidence to preserve based on the situation. This is important, as collection of one piece of evidence often destroys others. All stages of the investigative process must be documented, both to attest to the soundness of the evidence in court and to avoid missing details. During the survey, you should document what evidence is going to be collected, how it is going to be collected, and who is going to collect it.

3. *Preservation*: We will discuss this step in more detail later in this lecture, but it basically deals with physically seizing and/or making forensic duplicates of the digital evidence sources.

4. *Examination*: This step will be the primary focus of the remainder of this module, so we will discuss it at length later. Once the devices and data are secured, you must use digital forensics techniques to find evidence therein. This is commonly referred to as "examination."

5. *Analysis*: Once the evidence has been extracted, you must interpret it to understand what it means. For example, examination might reveal that a prefetch file for program X exists, but no entry exists in the registry or program files for program X. Analysis of this evidence would result in the understanding that program X was uninstalled sometime since the last accessed time of the prefetch file. The analysis step is where you combine the knowledge gained from the digital forensics evidence with knowledge from other evidence sources and information about the case to form an overall picture of events and draw conclusions.

6. *Reporting*: After you have concluded your investigation and drawn your conclusions, you must write a report for digestion by decision makers. This should clearly summarize the relevant findings and conclusions with enough detail to support decisions and potentially be presented in court.

In a large investigation, these steps would likely be performed by different individuals. Preparation, survey, and preservation might be performed by uniformed officers or detectives with some training in digital forensics, while examination and analysis would be performed by a digital forensics expert with (hopefully) significant training and experience back at the lab.

2.4.1.2 Preservation Considerations

Depending on the type of investigation, you may only be interested in grabbing a few specific files if you know what you are looking for. In cases where digital evidence is likely to play a more important role and/or you are not sure what you might need, you should take full bit-by-bit images of the devices.

In some situations, where a suspect may be escaping or human life is in danger, such as kidnapping, a less forensically sound on-scene examination may be required. In such situations, the priority would be getting actionable intelligence as soon as possible, rather than gathering evidence that can be

presented in court. This is also the case when digital forensics is used in active military engagements.

If a running computer is encountered, the preservation step is where you must make the difficult decision to pull the plug or not. At least until the late 1990's, it was standard practice to disconnect the power from a running computer, to prevent damage to evidence in a sloppy on-scene analysis by an officer with insufficient digital forensics training. With digital forensics training becoming more widespread in law enforcement and the development of more sophisticated techniques for analyzing running systems, or "live-analysis," there has been a gradual shift away from this cut-and-dry policy. There is also a growing appreciation of the importance of the digital evidence lost when disconnecting the power. For example, active network connections, running processes and their state, unsaved application data, and certain types of malware only appear in RAM while the computer is running. Most importantly though, are decrypted files. Many applications and operating systems, have options to store the users' data in encrypted form on the hard disk, decrypting it as needed for use in memory. We will talk about techniques for dealing with encrypted files in another lecture, but suffice it to say that this is an important motivation to attempt a live analysis. However, live analysis is very difficult, even with the ideal tools for the exact circumstances, and in general it is impossible to guarantee the integrity of the device data if you attempt a live analysis. You inevitably have to accept some level of spoliation. Evaluation of whether this type of analysis is possible, warranted, and worthwhile is a difficult decision that must be made on a case by case basis. While these techniques and questions are a very exciting area of research, engineering, and policy, the details of live analysis is are beyond the scope of this course.

2.4.1.3 File Systems

Before we cover more details of proper data preservation techniques, we need to cover a few basics, starting with understanding file systems.

The purpose of a file system is to facilitate long term storage and retrieval of data by a computer. The interface must be standard for interoperability between computers. An analogy would be the Dewey Decimal System, commonly used by libraries to organize books. A person who is familiar with the

Dewey Decimal System can walk into any library that uses it and easily find books on the topic they're interested in.

It will be helpful to understand some basic terminology we will be using throughout this module.

- A *sector* is the smallest unit of data that can be read from the hard disk, typically 512 Bytes. The sector is an atomic unit, so you cannot, for example, read 4.5 sectors from the disk.
- A *volume* is a collection of addressable sectors.
- A *partition* is a collection of consecutive addressable sectors.
- A *cluster* is a standard size container for storing file contents in NTFS. The number of sectors in a cluster is defined boot sector and must be a power of 2.

The distinction between partition and volume can be confusing. Technically a partition is a volume, but a volume is not necessarily a partition. The entire hard disk is also a volume and a partition.

The first sector on a hard disk will contain a pointer to a partition table that lists the location, size, and file system type of every partition on the device. Before being used by a computer, a partition must be formatted with a file system. For example, the "C drive" on a Windows computer is actually an NTFS formatted partition.

2.4.1.4 Basic Hardware Components

There are many other components of a modern computer, some required, some not, but these are the most important components of a computer to a digital forensics analyst.

- The *CPU* is the brain of the computer. It performs arithmetic operations and issues instructions to other hardware components.
- *Memory*, or RAM, is the workspace of the computer. It is where programs store data for quick access while they are using it.
- The *hard drive* is the storage of the computer. It is where data is stored when it is not being used.

- The *motherboard* is the backbone of the computer. All other components plug into it. The firmware running on a motherboard is called the Basic Input Output System, or BIOS.

2.4.1.5 Forensic Duplication

There are many elements in the process of preserving digital evidence sources. Most of these are general investigative practices and are shared with conventional forensic disciplines like fingerprint analysis, such as photographing all objects in their original position before taking them into evidence. The preservation techniques we are interested in this course are those specifically pertaining to digital evidence.

Forensic examination should not be conducted on the original device, unless the circumstances absolutely necessitate it, since this could lead to inadvertently damaging the evidence, and the required cautionary measures would make investigation inconvenient. Instead, an evidentiary drive should be copied using techniques that do not change the contents of the original drive. Examiners can then experiment freely on the duplicate disk image without risk of damaging the original evidence.

The problem of copying all the data in a hard drive without changing any of it can be somewhat tricky, and examiners usually use specialized proprietary digital forensics tools to do it. As an interesting side note however, the old school UNIX tool "dd" can be used to make a forensically sound duplicate, if you use the right parameters. However, if there is an error in the duplication process, or the tools are used incorrectly, the duplicate may not be a true copy of the original, or worse, the original may be damaged. To avoid this, and to increase the confidence the court will have in the soundness of any resulting evidence, duplication is usually done with certified digital forensics tools. The CFTT (Computer Forensics Tools Testing) team at NIST (the National Institute of Standards and Technology) certifies computer forensics tools for forensic soundness. Whenever possible, a DF practitioner should use tools certified by NIST to not alter evidence.

One of the most common problems that can happen if forensic duplication is done improperly is an alteration of the timestamps of files on the evidentiary device. Timestamp information is critical in reconstructing crimes, and in some cases, can determine if an action was even illegal. For example, if an

examination of a disk indicates that the suspect tried to delete a suspicious file, then it could be destruction of evidence. If the suspect tried to delete the file after he or she became aware of the investigation, then it is a crime, if they attempted to delete the file before they became aware of the investigation, then it is not a crime. If the timestamps are not intact, there is no way to determine this.

Even if the forensic duplication tool is certified to not alter the source, a *write blocker* should be employed when imaging an evidentiary drive. Write blockers prevent a computer from sending write commands to a hard drive. This functions as a safeguard against damage to the original device, and it bolsters the court's confidence in the integrity of the evidence.

There are two general types of write blockers.

- *Hardware write blockers* are physical devices that sit between a computer making the duplicate and the evidentiary hard disk and intercept and block any write commands.

- *Software write blockers* are programs that run on the computer making the duplicate and attempt to intercept any system instructions that would result in a disk write command.

2.4.1.6 DCO and HPA[23]

The *Device Configuration Overlay* (DCO) and *Host Protected Area* (HPA) are areas on a SATA or ATA hard disk that are normally inaccessible to the operating system and other programs, but are sometimes used by the BIOS to store important recovery information.

A knowledgeable individual can hide files in them however, so they should be included in the disk image, if they exist. The sizes of the HPA and DCO are defined by values stored within the hard disk's internal logic board. If a SATA hard disk is configured with a DCO and or HPA, the size it will report when queried with normal read commands will be its true size minus the size of the DCO and HPA. The partition table will NOT list the HPA or DCO, so to detect these areas, you (or a tool) must send the

IDENTIFY_DEVICE,

READ_NATIVE_MAX_ADDRESS, and

DEVICE_CONFIGURATION_IDENTIFY

commands directly to the hard disk, bypassing the BIOS. By subtracting the pointers returned by these commands, you can determine the size of the HPA and DCO.

The DCO and HPA can only be accessed with commands that modify these pointers. After setting all 3 pointers to the actual size of the disk, the size of the DCO and HPA will be 0. The disk does not erase any data in doing this, it just moves the pointers indicating the HPA and DCO size. The SET_MAX_ADDRESS command is used to set the pointer that delineates the end of the user addressable sectors. This is the pointer returned by the IDENTIFY_DEVICE command. The DEVICE_CONFIGURATION_SET command is used to set the pointer delineating the end of the HPA and start of the DCO. This is the pointer returned by the READ_NATIVE_MAX_ADDRESS command. But its usage is restricted. You may only use the DEVICE_CONFIGURATION_SET command once to create the DCO. It cannot be resized, so to remove it you must issue the DEVICE_CONFIGURATION_RESTORE command to reset the READ_NATIVE_MAX_ADDRESS pointer to the actual disk size. Once this reset is done, you may create the DCO again with a different size. Also note that the DCO cannot be created if there is already an HPA. You can have both, but you must create the DCO first . After these pointers have been reset, the data that was inside the DCO or HPA will now be in user addressable sectors. However, these sectors will not be formatted with a file system. Many forensics imaging tools will do this for you, and you should verify whether a tool does this properly before using it.

Note that the DCO is rarely used, and many hard disk manufacturers don't implement it properly, so it may cause damage to the data if you use it. If you detect a DCO on an evidentiary device, It is recommended to take an image without the DCO and again with the DCO, so at least you have the user addressable sectors and the HPA.

If there is a Host Protected Area (HPA) or Device Configuration Overlay (DCO) on the hard drive, then you must send a command to the drive to change the pointers determining the size of the HPA and DCO to allow access. But this changes the state of the drive and is considered a write operation. Some write blockers will not allow these commands to pass, while others will. In practice, an examiner should know which commands his write blocker allows, and should check disks for a HPA or DCO before imaging. Write operations initiated independently by the BIOS cannot be intercepted

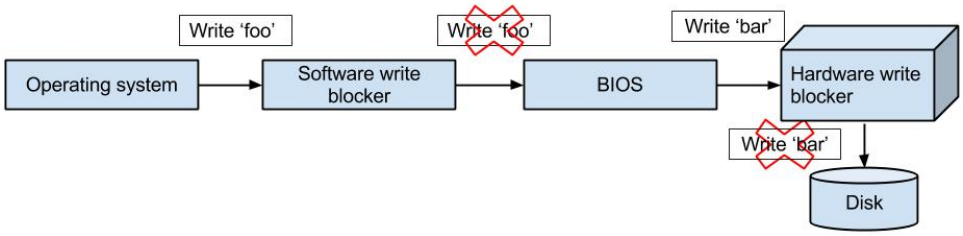

Figure 2.1: Example showing the difference between software and hardware write blockers. If there was no hardware write blocker here, the 'bar' write command would have been executed on the disk.

by software write blockers. If the BIOS were to attempt to write to the hard disk (probably to the HPA or DCO) a software write blocker would not prevent the loss of data.

2.4.1.7 Sources

The primary sources for this lecture's material were Brian Carrier's textbook: *File System Forensic Analysis* [24] and Eoghan Casey's textbook: *Digital Evidence and Computer Crime* [18].

2.4.2 Lecture: Introduction to File System Forensics

Last lecture we covered the preservation step in a computer forensics investigation. We will now move on to the examination and analysis step, which will be the main focus of the remainder of this module. While there are many related areas of computer forensics that can be applied to examination and analysis, we will start at the lowest level: file system forensics.

File system forensics deals with evidence related to file system events such as file creation, deletion, and duplication. For example, file system forensics deals with finding hidden files; recovering deleted files; determining creation, modification, and last access times; and determining file ownership. Evidence related to the content of files and the behavior of the operating system and various applications is outside the purview of file system forensics, and will be discussed later in this module.

2.4.2.1 NTFS Basics

Due to time constraints, we have decided to focus on only one file system in this course. We will discuss NTFS, the file system used by Microsoft Windows, because it is the most common and most likely to be encountered in practice.

The *Master File Table* (MFT) is the key data structure in NTFS. Everything in an NTFS partition, except the boot code, is located in the MFT. The MFT is actually a list of *file records*. Every file and directory in an NTFS formatted partition has a corresponding file record in the MFT. The MFT allocates a small number of records to start, but it can expand as needed as more files are added. As more file records are added to the MFT, it may not be able to fit in a single contiguous region of the disk. In this case, it would be fragmented into multiple locations on the disk. The MFT itself is considered a file, and the first file record in the MFT is for itself. This file record lists the size and location of all sections of the MFT fragmented on the disk. The file records are always 1024 bytes long, but technically this size is set in the boot sector, so later versions of NTFS may include a different size file record. File records contain a standard header and several attributes depending on the type of file.

The first cluster in a bootable partition is called the *boot sector*. The operating system can just look at the first cluster in the NTFS partition, and knowing it is NTFS, read the location of the first entry in the MFT from cluster 0. The first entry in the MFT contains the size and location of the rest of the MFT, which contains all the files in the partition. Note that the $MFT file record lists the location of the rest of the MFT as cluster addresses.

2.4.2.2 File record attributes

Attributes are small data structures within a file record that define the characteristics of that file. Each attribute has a header specifying its name, size, and type-id number. Note that *name* is not the same as *type*. There can be multiple attributes of the same type in a single file record, but they must have different names. The first instance of each attribute type within a file record will have a blank name, so only duplicate attributes will have names.

The attribute header is followed by the attribute content. Since the attributes must be contained in a file record with a fixed length of 1024 bytes, attribute content that is too large to fit must be *non-resident*. The header of a non-resident attribute contains a pointer to a disk location outside the MFT where the actual attribute content is stored. Strangely, non-resident attributes are still considered part of the file record, so the "size" of the MFT is much larger than the end address minus the start address.

NTFS allows for custom attribute definitions, but most file record attributes are of predefined standard types. We will briefly mention a few of these types here to give you a general context, before we do more in depth analysis in the next lecture. This is not an exhaustive list, just a sample of the most important ones for our purposes. Note the naming convention: Attribute names begin with $ and are all capital letters with underscores to separate words.

- The $STANDARD_INFORMATION attribute stores timestamps, ownership, and security information.
- The $DATA attribute contains the actual contents of the file.
- $INDEX_ROOT and $INDEX_ALLOCATION attributes contain a list of the files and subdirectories contained in a directory.
- The $FILE_NAME attribute stores the name of the file and timestamp information.
- The $SECURITY_DESCRIPTOR attribute stores Access Control List (ACL) and security properties of the file.
- The $BITMAP attribute stores the allocation status of the MFT.

While both $STANDARD_INFORMATION and $FILE_NAME attributes store created, last accessed, and last modified timestamps, they are not necessarily the same values for each. We will discuss the different ways these timestamps are updated in the next lecture, along with how they can be clues for an investigation.

Each bit in the $BITMAP's content corresponds to a file record in the MFT. If it is set to 1, the file record is allocated (i.e. in use), if the bit is set to 0, the file record is unallocated (i.e. the file was deleted) and that space

in the MFT is available for new files. Large directory files will also have a $BITMAP attribute to store the allocation status of its index.

An *Access Control List* (ACL) is a list of permissions for how a file can be accessed. For example, an ACL entry might grant the user "Bob" permission to read a file's contents. The $SECURITY_DESCRIPTOR attribute does not actually contain the full specification of the ACL. It contains a reference to a file that contains the full ACL for every file in the file system. This was done to eliminate redundant storage to save space, since there are likely to be few unique ACL configurations shared between many files. For example, most of user Bob's files will have identical ACL entries, granting him and the system administrator access and prohibiting access to other users.

Every file has a $DATA attribute, although it may have a size of 0. Directories do not normally have a $DATA attribute, but they could technically. So you could have something like a document file that has sub directories. This is technically allowed by the NTFS specification, but it highly unlikely that Windows would create such a file. The first instance of a $DATA attribute in a file record does not have a name, but some forensics tools, like the one we will be using in this course: "The Sleuth Kit", will display "$Data" as the name for the first $DATA attribute in a file. If there is more than one $DATA attribute, it is referred to as an *Alternate Data Stream* (ADS).

ADS were originally included to allow compatibility with Mac file systems. However, Windows now uses ADS to store various file properties that the user isn't intended to directly access, such as origin information of downloaded files or file authorship information for documents. The contents of an ADS are not displayed to users by default, but can be accessed easily with digital forensics tools or even a command prompt if you know the proper syntax. ADS are sometimes used by knowledgeable individuals and malicious programs to hide data, since a cursory inspection might miss them.

2.4.2.3 Non-base file records

File records have a fixed length of 1024 bytes, but can have any number of attributes. These attributes can have non-resident content, but the headers at least must be stored in the file record. If the number of attribute headers grows too large to fit in a single file record, then a non-base file record is allocated. This actually happens more often than you might think, be-

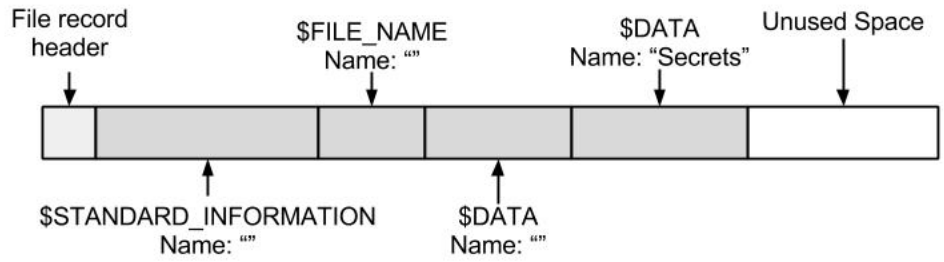

Figure 2.2: File record with ADS. Note the second attribute of type $DATA named “Secrets” Secrets is an ADS. A forensics tool would show the ADS, and it could be accessed from the command prompt using the syntax “filename:Secrets”. For example, if the file was named “temp.txt”, you could display the content of secrets in notepad by typing “notepad temp.txt:Secrets” at the command prompt in the directory containing “temp.txt”.

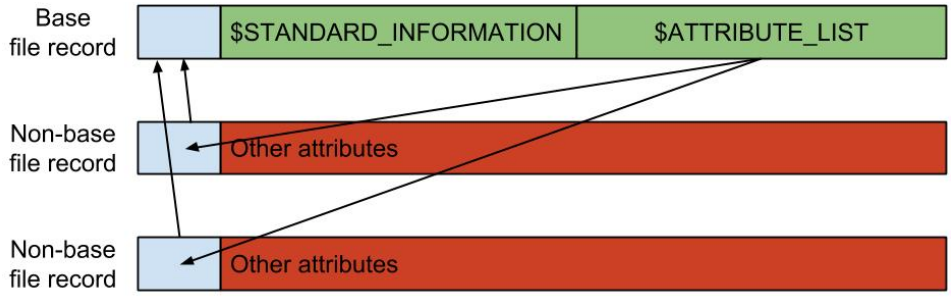

Figure 2.3: Non-base file records. Note the $ATTERIBUTE_LIST attribute indicates the location of the non-base file records.

cause non-resident content can become fragmented, especially for large files. The location of all the fragments must be tracked in the $DATA attribute's header. The header of a non-base file records will have a pointer to the base file record, and the base file record will have an $ATTRIBUTE_LIST attribute that contains a list of each of the file's attributes and pointers to the non-base file records that contain them.

2.4.2.4 Encryption

Encryption is a widely used method of preventing unauthorized persons from reading private data. It has many legitimate uses and is essential to make tasks like online banking secure. However, it is also often used by criminals to hide incriminating digital evidence, and it is in this context that it is

important to our current discussion.

Encryption is performed by supplying an *encryption function* with a *secret key* and applying it to some data you wish to hide. The encryption function uses the secret key to scramble the data in such a way that it is almost indistinguishable from random data. The resulting scrambled data is often referred to as *ciphertext* or simply encrypted data. In order to read the data, you must use the appropriate *decryption function* with the same secret key that was used to encrypt the data. This will return the original data. Thus, the secret key is like a password for accessing the encrypted data.

If you encounter encrypted files during an investigation, you have two basic options. You could attempt to brute force the encryption, which basically means have a computer try and guess many encryption keys until it finds the correct one. However, the difficulty of brute forcing an encryption key depends on the strength of the key. If the secret key is very poorly chosen, or the encryption function is flawed in some significant way, then a decryption tool can find the secret key quickly. However, brute forcing data that was encrypted using a modern encryption function, like AES256, with a well chosen secret key would likely take millions of years, even using a supercomputing cluster.

The other option is to find the secret key by other means. Sometimes people will write their passwords and encryption keys in notebooks, and these should be located in the physical investigation. Another common mistake people make is to use their login password for their computer or other accounts as their encryption key. Digital forensics tools can often recover login credentials stored by web browsers.

There is another problem with encrypted data in an investigation. There is some debate as to whether encrypted files require a separate search warrant, so make sure your warrant has provisions for decrypting files, or it could constitute illegal search and compromise the resulting evidence.

2.4.2.5 Sources

The primary source for this lecture's material was Brian Carrier's textbook: *File System Forensic Analysis* [24].

2.4.3 Lecture: NTFS Analysis

In this lecture we are going to go more in depth with the way NTFS works and some of the potential evidence sources in NTFS analysis. We will also begin our discussion of deleted file recovery.

2.4.3.1 File System Metadata Files

In addition to the files created by the user and operating system, NTFS requires its own internal metadata files. These files are used to operate and organize the file system, and a solid understanding of their purpose and usage is required to conduct sound forensic analysis of a Windows file system. Since they are hidden from the user, they can be some of the best sources of digital evidence. Even if you are using a highly automated tool to assist your analysis, you need to know what the tool is doing in order to interpret and report your findings accurately and explain them in court if you are called as an expert witness.

- $MFT

  As we covered in the last lecture, the first file record in the MFT is for the $MFT file itself. All other file records are non-resident content of the $DATA attribute of the $MFT file record. Thus by processing the header of the $DATA attribute of $MFT, you can find the cluster addresses (disk locations) of all the other file records. The $MFT file's $BITMAP attribute is used to manage the allocation status of the MFT records (i.e. whether they are deleted or not). The $STANDARD_INFORMATION attribute of the $MFT file stores the date and time the file system was created, and it not updated in normal operation.

- $MFTMirr

  Since so much of NTFS operation depends on the MFT, it can be a single point of failure if it is corrupted. The $MFTMirr file stores a backup of some of the most important entries in the MFT. The $DATA attribute of the $MFTMirr file contains the file records for the first 4 metadata files in the MFT: $MFT, $MFTMirr, $LogFile, and $Volume. It is non-resident, and its content is always stored at the middle of the

volume, far away from the rest of the MFT. Thus if the start of the MFT is corrupted or accidentally overwritten somehow, a recovery tool can look in the middle of the volume to find the file records for $MFT, $MFTMirr, $LogFile, and $Volume. The file record for the $MFT will contain pointers to all the remaining file records, the $LogFile file which can be used to restore the file system to a safe state, and the $Volume file contains important version and status information. A digital forensics examiner or a recovery tool may be able to use the $MFTMirr to restore a partially corrupted volume, such as one that has been partially overwritten, or somehow damaged during acquisition.

- $Boot

  The $Boot file contains the boot sector of the file system. The non-resident content of the $DATA attribute of the $Boot file must be located at sector 0 of the volume, and contains the information required to load the file system. It will contain the location of the start of the MFT, the size of clusters and file records, the file system's serial number, and the boot code. The boot code comprises instructions to locate and load the code to initialize the operating system, if this is a bootable volume. A backup copy of the boot sector is sometimes stored in the unused space between the end of the file system and the end of the volume.

- $BitMap

  Not to be confused with the $BITMAP *attribute*. the $BitMap *file* defines the allocation status of all the clusters in the file system. The $DATA attribute of the $BitMap file contains the actual bitmap, with a 0 indicating an unallocated cluster and a 1 indicating an allocated cluster.

2.4.3.2 NTFS Indexes

Several important mechanisms in NTFS depend on *indexes*. While there are various types of indexes, they are all basically collections of index entries. Each *index entry* contains a header and an attribute of some kind. The type of attribute in the index entries depends on what the index is used for. The

details of the allocation algorithms and data structures used to store these index entries is beyond the scope of this course, and not important for our discussion here.

The most common indexes are directory indexes, where each file in the directory has a corresponding directory index entry containing a $FILE_NAME attributes and a pointer to the file record for that file. Directory indexes are stored in the $INDEX_ROOT and $INDEX_ALLOCATION attributes. If there are only a few entries, only the $INDEX_ROOT is needed, but for large directories, the $INDEX_ALLOCATION is used as well. Note that these directory indexes are not the same as the directory tree you see in Windows Explorer (e.g. "C:\Documents\foo"); every directory has its own index.

2.4.3.3 More Attribute Details

The $STANDARD_INFORMATION attribute exists for all files and directories. It contains time stamp information for "Created", "File Modified" (last time the content of $DATA or the $INDEX_* attributes were modified), "MFT Modified" (last time the metadata of this file was modified, which is not shown to the user in Windows), and "Accessed" time (time the $DATA was last read from or written to). Note that this is not the only attribute that contains these timestamps, but the $STANDARD_INFORMATION timestamps are the primary ones. These timestamps are critical for an investigation to correlate digital evidence and reconstruct a timeline of events.

This attribute also contains ownership and security information that can help link a file to a specific user account. For example, if only Bob's account can access the file, then someone with Bob's password must have put it there. Note that this assertion makes a critical (and sometimes false) assumption: that the file was added using default Windows file management mechanisms. Enforcement of ownership and security requirements is voluntary unless encryption is used, which is not the default, so if the file system is mounted by a non-Windows operating system then files can be placed anywhere, and the ownership information can be forged. What you can really say is, "someone with access to this computer put these files here." The $STANDARD_INFORMATION attribute also contains a flag for general properties of the file, such as read-only, compressed, sparse, or encrypted.

For every file and directory there is at least one $FILE_NAME attribute in its file record and at least one in its parent directory's index. This includes directory files. This attribute contains a pointer to the file record of its parent directory, and its parent directory's $FILE_NAME attribute will have a reference to *its* parent directory, and so on. This is very important, because when files are deleted their parent directories have their indexes sorted, often overwriting the index entry for the deleted file. However, the file record for the deleted file will not likely be overwritten right away. With the parent directory pointer in the $FILE_NAME attribute of a file, you can often reconstruct the full path to a deleted file.

The $FILE_NAME attribute also contains "Created," "File Modified," "MFT Modified," and "Accessed" timestamps, like the $STANDARD_INFORMATION attribute, but the timestamps on the $FILE_NAME attribute are not usually updated by Windows, so they often correspond to the time the file was created. This can be important in an investigation, because if a suspect intentionally modified the timestamps on some files to mislead investigators, he or she most likely modified the $STANDARD_INFORMATION timestamps, and the true file creation date can often be found in the $FILE_NAME attribute.

2.4.3.4 File Deletion Example[24]

To tie all this discussion together, we will go through an example of what happens under the hood in NTFS when a file is deleted. Note that you can find the file record for the root directory, because it is always file record number 5, and you can traverse the directories by processing their $INDEX_ROOT and $INDEX_ALLOCATION attributes to find pointers to the file records of the files and directories they contain. The general steps for deleting the file "C:\Examples\file.dat" are as follows.

1. Process the $INDEX_ROOT and $INDEX_ALLOCATION attributes of the root directory's file record to find the file record for the "Examples" directory. Update the last accessed time of the root directory.

2. Process the $INDEX_ROOT and $INDEX_ALLOCATION attributes of the "Examples" director's file record to find the file record for "file.dat".

3. Remove the index entry for "file.dat" from the "Examples" directory index and resort the index if needed. Update last written, modified, and accessed times for "Examples" directory.

4. Deallocate the file record for "file.dat" by clearing the "in-use" flag in its file record header and clearing the corresponding bit in the $BITMAP attribute of the $MFT file record. Recall that the $BITMAP attribute of the $MFT file record indicates the allocation status of the file records themselves.

5. Process the headers of the non-resident attributes in the file record for "file.dat" to find the clusters storing the non-resident content. Set the corresponding bits to 0 in the $Bitmap metadata file.

Note that the non-resident content clusters were NOT overwritten and the file record the "file.dat" was NOT overwritten, they were just marked as unallocated. So until the file system creates another file in that same memory location, the metadata and pointers are still there. This will be important for our next topic: deleted file recovery.

Recall that the $Bitmap metadata file stores the allocation status of all the clusters in the file system. Not to be confused with the $BITMAP attribute of the $MFT file record, which stores the allocation status of the file records. A file record marked as unallocated in the $MFT file record's $BITMAP attribute is only available for reuse as a new file record in the MFT. The clusters it is stored in are NOT marked as unallocated in the $Bitmap metadata file. Clusters marked as unallocated in the $Bitmap metadata file are available for reuse in any way, such as storing non-resident attribute content.

2.4.3.5 Deleted File Recovery

Probably the most important application of file system forensics is deleted file recovery. Deleted files can be important to an investigation, not just for their contents, but also the fact that the user attempted to delete them. Attempting to delete files can constitute destruction of evidence, if the user is aware that they are under investigation at the time of deletion. The techniques we will cover in the remainder of this lecture and the next lecture

are also widely used in private practice to recover accidentally deleted or damaged data.

If the file record for the deleted files remains intact, the process of recovery is very easy. You can simply look for file records marked as not in use. If the attribute content is resident, you can just read it from the file record. If the attribute is non-resident, like the $DATA attribute of a large file, you can follow the pointers in the non-resident attribute's header to find the clusters on the disk that have the file's data. Tools will do this automatically by looking through the MFT for file records marked as not in use and looking at their $FILE_NAME attributes to reconstruct the files' paths. Thus when you browse a file system in an investigative tool like Autopsy, it shows deleted files in the directories they were deleted from. This process is sometimes called "undelete" when software advertises it as a feature. Note that this only works if both the file record and the file's non-resident content have not been overwritten.

2.4.3.6 Sources

The primary source for this lecture's material was Brian Carrier's textbook: *File System Forensic Analysis* [24].

2.4.4 Lecture: File Carving

In the last lecture, we learned about how to recover deleted files when the file record and file content remains intact, but if the file record has been overwritten for use by another file, then you cannot follow the pointers in the attribute header to find the attribute's content. However, as long as the non-resident attribute content has not been overwritten, you can still recover it. Recovering data without using the file system metadata is called *file carving*. You may note in the following discussion that all file carving techniques require you to know the type of file you are looking for.

Modern file systems tend to overwrite the metadata for files when they are deleted, but also manage their data better, so there are lower levels of fragmentation, making file carving more necessary and viable. Note that this behavior is actually implemented in the operating system level in Windows,

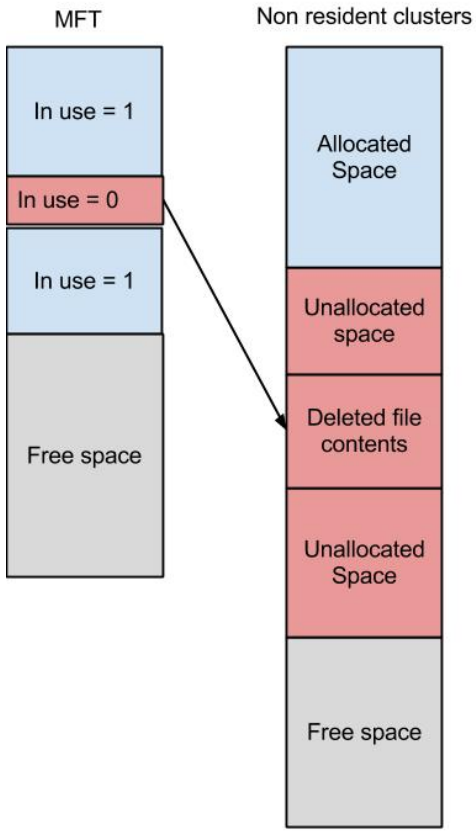

Figure 2.4: Deleted file recovery with metadata. As you can see, if the file record has not been reallocated and the content clusters have not been overwritten, then you have all the information required to access the file as if it had never been deleted. You just need a forensics tool to view it, because the OS will not display these unallocated files.

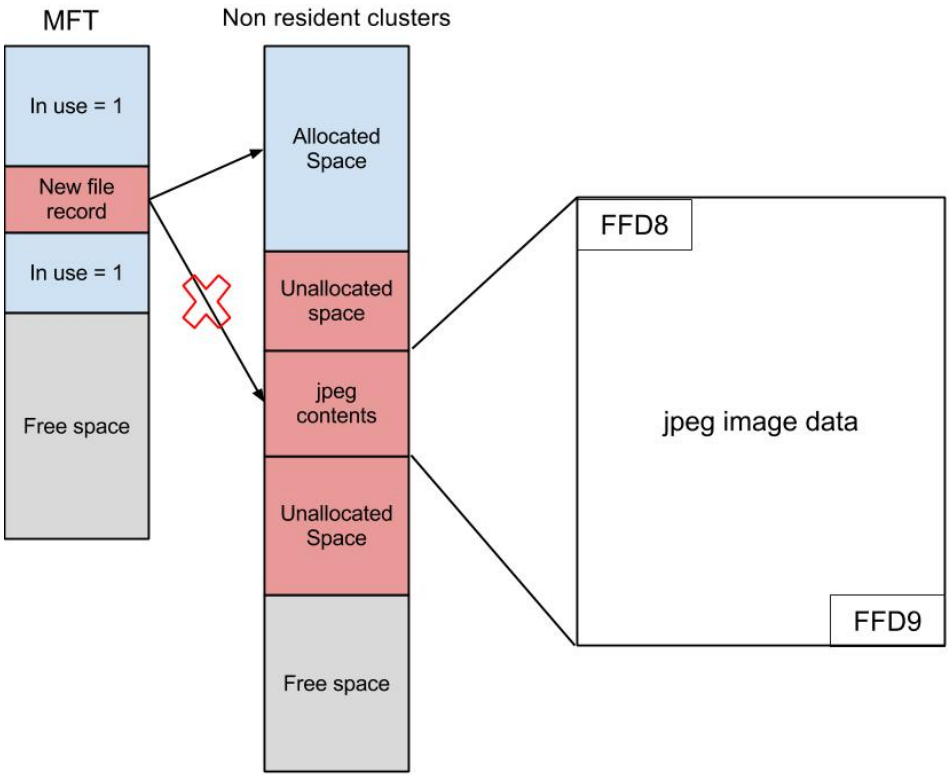

Figure 2.5: Basic header-footer file carving. Although the file record metadata was overwritten with a new file record, the jpeg can be recovered by searching for "FFD8" and "FFD9" and carving the data in between.

the NTFS version has not been updated since XP, but Windows 7 nonetheless overwrites the file records of deleted files much more quickly than XP.

2.4.4.1 Basic File Carving Techniques

In the simplest case, a file's non-resident clusters are all intact and not fragmented. That is to say, they all exist in a contiguous section of the disk. In this basic case, there are several techniques for file carving.

- *Header-footer carving*: Most file types have a unique sequence of values at the start and end of the file. For example, the header of jpeg files always begins with the string "FFD8" and the footer ends in "FFD9". If you search the disk until you find a header value, then search the following data until you find the footer value, then the data in between is most likely a jpeg. In this basic case there is not much difficulty

in recovering known file types, and there are many tools that do this reliably, but some difficulties can arise. For example, if the file type does not have a known footer signature, you have to try to figure out the length of the file. This is often in the header, but if the file format is proprietary, it may be difficult to locate reliably. Sometimes files are embedded in other files, such as jpegs in word documents. These can disrupt file carving tools and lead to incorrect results.

- *File structure based carving* uses the internal layout of a file. If there is standard information in the file other than the header and footer, such as identifier strings and size information, this can be used to carve the file.

- *Content-based carving*: Some tools are able to leverage patterns in the content of files to find them on disk. Some things they look for include structured data, such as HTML and XML files, character count, language recognition, statistical attributes, and information entropy.

2.4.4.2 Carving with Fragmented Clusters

You will recall that I have mentioned fragmentation several times, but I never really defined it. For the following discussion it is important to know a little bit more about fragmentation. *Fragmentation* is when the non-resident content of an attribute is stored in multiple locations on the disk, rather than one contiguous block. The non-resident attribute's header will contain multiple pointers, one to each block of clusters the file is stored in.

Fragmentation happens for various reasons. For example, when the partition is close to full, and no large unallocated blocks are available, or a file's size grows significantly after it is initially allocated, so it no longer fits in the small unallocated block it was originally placed in. Fragmentation is becoming less common as modern operating systems implement better allocation techniques. Another contributing factor to the decrease in fragmentation is that most of the data volume on modern hard drives often consists of media files, which are copied and read, but not modified in size, so they don't fragment as often.

The real difficulty with file carving comes in when the file is fragmented on disk. There are many theoretical mechanisms for reconstructing fragmented

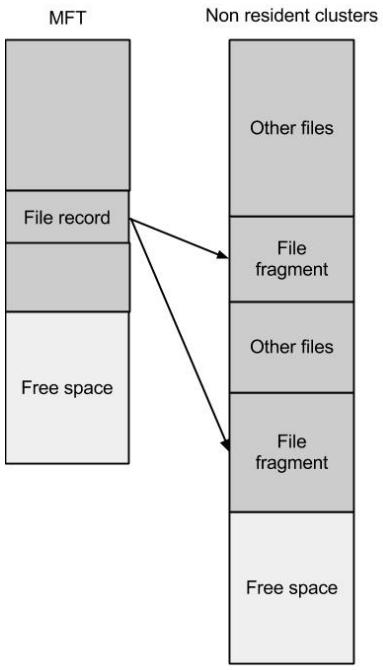

Figure 2.6: File fragmentation. Note the file record contains pointers to the different fragments.

files, but only a few have been successfully used on real data sets. Two techniques for carving fragmented files that work in practice are Bifragmented Gap Carving and SmartCarving.

Bifragmented Gap Carving [25] only works if the file is in exactly two parts. Basically, the first one has the header, the second the footer. So you take all the headers and try to match them with the footers. The difficulty is that the two fragments are both more than one cluster long, so you must determine where the fragments begin and end. You can identify the header and footer cluster, but you only know that some number of clusters after the header and some number of clusters before the footer are part of the file.

Let the gap size g denote the number of clusters between the header and footer that do not belong to the file. For each g starting at 1 up to the distance between the header and footer clusters, remove every possible set of g contiguous clusters between the header and footer and attempt validation on each attempt.

This is guaranteed to work if the file is in two fragments, all its clusters are intact, the header fragment is before the footer fragment, and you have a reliable validation mechanism, but it does not scale well for large gap sizes.

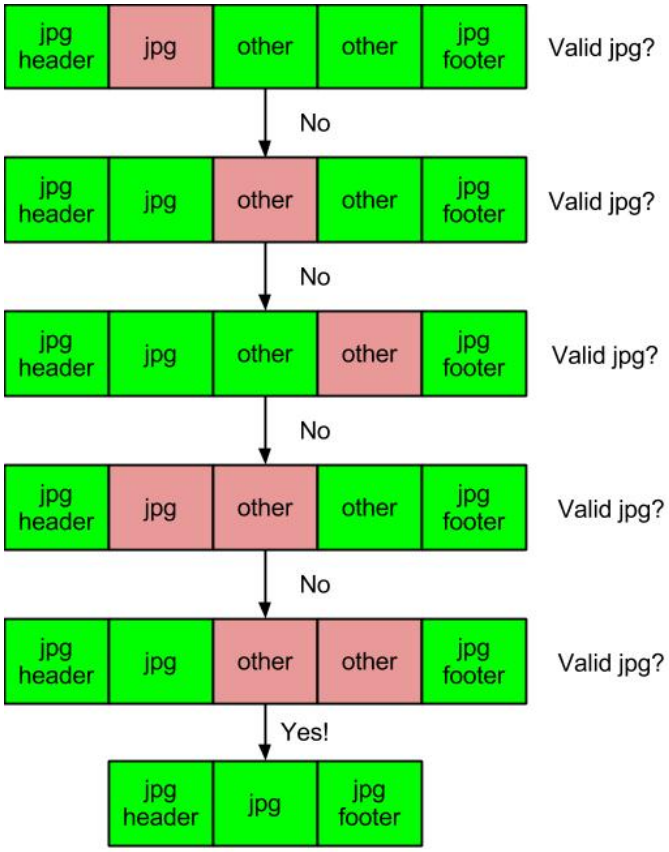

Figure 2.7: Bifragmented Gap Carving example. The "gap" (red squares) at each iteration is varied until the correct configuration is reached, where the "gap" is aligned with all the non-jpg clusters between the jpg fragments.

The worst case runtime is $O(nd^2)$ where d is the number of clusters between the header and footer and n is the number of files being carved. Missing or corrupt clusters will result in the worst case runtime frequently.

SmartCarving is a technique based on research published in 2009 by Anandabrata Pal and Nasir Memon that works well for carving jpeg files fragmented into more than two parts [26].

SmartCarving consists of three high level steps:

1. *Pre-processing*: in this stage any compressed or encrypted clusters are decompressed and decrypted, and all allocated clusters are removed. Reducing the number of candidate clusters is critical to improving the performance of the subsequent reassembly.

2. *Collation*: in this stage, the clusters are classified by file type. Various heuristics are used to identify file type, such as keyword searches, ASCII detection, entropy, and file signatures. Successfully classifying the clusters breaks the problem into parts and again reduces the number of inputs to the subsequent step.

3. *Reassembly*: in this stage, the filtered and classified clusters are taken as the inputs to a fairly complicated algorithm called sequential hypothesis parallel unique path (SHT-PUP), the details of which are far beyond the scope of this course, but it basically starts with the header for a file and iteratively finds subsequent fragments in the file.

The end result is a very effective reconstruction of fragmented files, even those fragmented into as many as four parts.

2.4.4.3 Slack Space

So far we have been talking about recovering files that have been deleted, had their file records overwritten, and even files whose content is fragmented across multiple areas of the disk, but what about a file whose non-resident clusters have been reallocated to a new file?

It turns out that even in this case, some data recovery may be possible. The trick is *slack space*. Disk space in a file system is allocated in fixed chunks, called clusters in NTFS, and the size of a file's content is rarely an

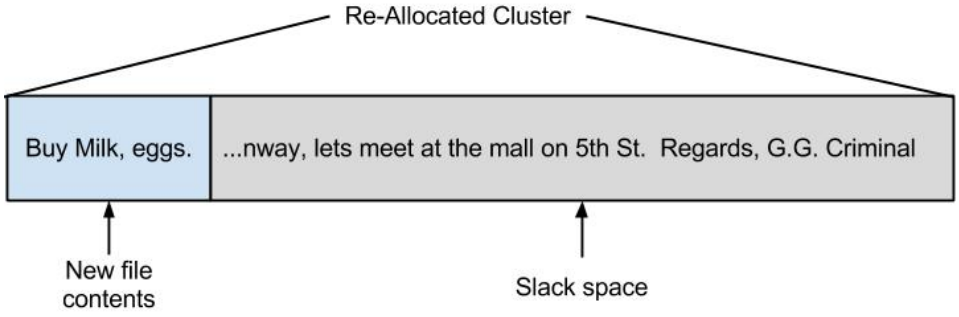

Figure 2.8: Slack space. Here you can see a file that was overwritten by a smaller file. The original file is an email from G.G. Criminal, that we definitely want everything we can get about, and the new file is a shopping list. If we did a keyword search for "G.G. Criminal" this cluster would come up.

exact multiple of the cluster size. So you often have some extra space left over at the end of a cluster. This extra space is called slack space, and in some cases it can still contain useful information. For example, if you are looking for a fairly large text file, and it's clusters were reallocated to a small file, you can just read the remainder of the text file from slack space. You can find data like this if you are looking for the non-resident content of a file whose file record is not overwritten, but its content has been, or if you are looking for a file with some unique characteristics, like a keyword or specific structural semantics, like a C source file.

2.4.4.4 SSD Forensics [27]

Solid State Drives (SSDs) are large scale storage devices for computers that offer much faster data retrieval than hard disks. They are still several times more expensive per unit of storage than hard disk, but the price is going down, and they are becoming more popular, so it is important to understand the implications they have for digital forensics.

SSDs use a completely different method for storing data than a hard disk. A hard disk stores data as magnetic signals on spinning platters. SSDs store data in flash memory, like a USB thumb drive, but redesigned to allow much more data density. So far so good. How does this effect forensics then? The difficulty stems with two facts:

1. Flash memory degrades every time you write to it, so frequently writing

to the same location in flash memory will damage the device.

2. Flash memory can be written to and read from very quickly, but you cannot write over saved data. The data must be erased in chunks called blocks before it can be written to.

You can think of flash memory like an "Etch-A-Sketch." You can write black lines as much as you want to make a picture, but to clear the screen for a new picture, you must shake it and erase all the black lines at once. Likewise in flash memory, you can set bits from 1 to 0 as much as you want, but to set them back to 1 you have to erase the whole block they are stored in, and this process is much slower than reading data and writing 0s.

To solve these two problems, SSD developers designed internal smart controllers for SSDs that do wear leveling and garbage collection. In *wear leveling*, the controller writes data to different locations on the drive based on which parts of the drive have been used least recently. To allow data to be retrieved while doing this, the controller maintains an internal mapping from the virtual sector addresses that the operating system sees and the actual physical location of the data.

The obvious consequence of this is that the device will quickly fill up with 1s and 0s and the controller will no longer be able to write. The solution is *garbage collection*. When physical blocks of flash memory are no longer needed, they are marked for garbage collection. There is a garbage collector process that constantly runs in parallel with the normal reads and writes and goes through the SSD and erases blocks marked for garbage collection.

So far, this still would not cause a problem for forensics, because the SSD controller would not know which files were marked as "unallocated" by the file system, so it would not mark them for garbage collection. However, the SSD developers introduced another optimization: the "TRIM" command. Operating systems (on behalf of their file systems) can now send a "TRIM" command to the SSD controller when they delete a file to indicate that it can be reused for new files. This greatly improves the efficiency of the garbage collection and the performance of the SSDs. However, it also means that files in unallocated space (the very files we are trying to recover as forensic analysts) will only remain on the disk until the garbage collector gets around to erasing their blocks. This usually only takes a few minutes.

It is still possible to recover files that were deleted from an SSD if you pull the power within a couple minutes of the file deletion, then physically disassemble the device and disconnect the controller, to disable garbage collection, and attach custom hardware to the flash chips to read the data directly. But this process is obviously impractical in almost all situations. So as it stands, there is not really a reliable method of recovering deleted data from an SSD.

Figure 2.9 shows a demonstration of how the wear leveling and garbage collection work. In practice files do not neatly fit into SSD memory blocks like this, but its easier to visualize this way.

In step 1, on the top left, has three ready blocks, a block marked for garbage collection, a text file, and a jpeg file.

In step 2, you see the state after the operating system writes to the jpeg file. Since the SSD cannot write to the block the jpeg is in, it writes to a ready block and marks the old version of the jpeg for garbage collection. Note the garbage collector is always running, and it erased the previously marked block.

In step 3 you see what happens after the OS edits the txt file and creates a doc file. As before, the new version of the txt file and the new doc file are written to a new location, and the old version of the txt file is marked for garbage collection. The garbage collector erases the old jpeg so that block is ready again.

In step 4, you see the state after the OS writes to the doc file. The new version of the doc file is written to a ready block and the old one is marked for garbage collection. The garbage collector erases the old text block.

2.4.4.5 Sources

The primary sources for this lecture's material were articles by Garfinkel [25], Pal [26], and Gobanov [27].

2.4.5 Lecture: Windows Analysis

So far in this module, we have discussed file system forensics with a focus on NTFS, and deleted file recovery. We have been treating the contents of files as a black-box, and ignored evidence related to operating system behavior. We will now give an introduction to the larger part of computer forensics: that

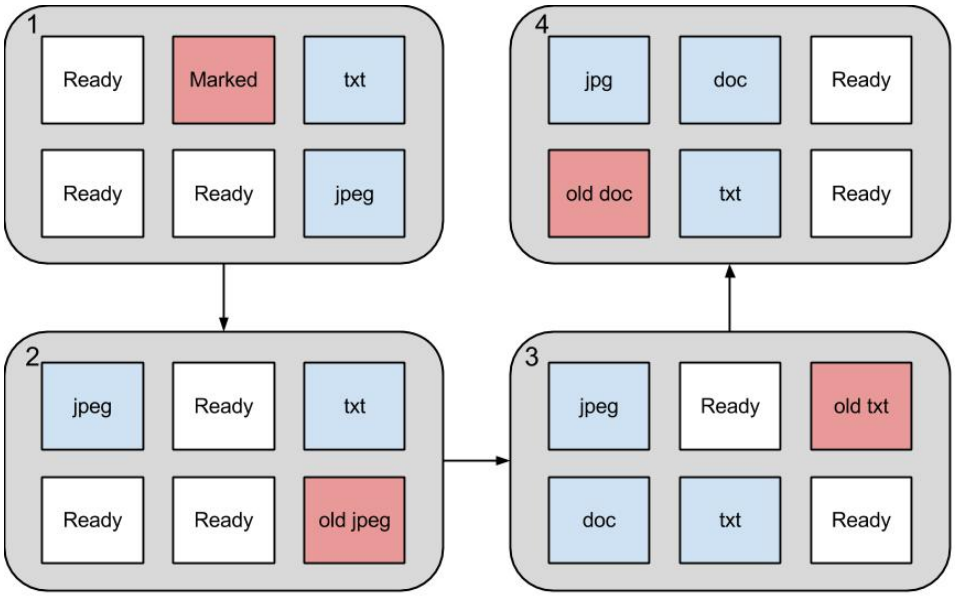

Figure 2.9: SSD garbage collection. In this figure, Blue is a valid file, Red is a file marked for garbage collection, and White is an erased block, ready for writing.

dealing with evidence related to operating system and application specific behavior. These subjects comprise the majority of the field of computer forensics, and we will only have time to cover a small portion in the two remaining lectures of this module.

The discussion in this lectures will give you an idea of what kind of evidence can be extracted from the operating system using computer forensics, by presenting specific examples and techniques that are particularly common and useful in computer forensics investigations. We will limit our topics to Microsoft Windows systems, since they are the most common and most likely to be familiar to you.

2.4.5.1 Registry

Windows stores persistent settings and usage information, such as autocomplete values, in the Registry. Basically, if Windows remembers it after a reboot, it's probably stored in the registry. A program running on Windows can use the registry to store its configuration, but this is optional. Some programs use their own files for configuration. Registry files are called registry hives and are located in "C:\Windows\system32\config" on Windows

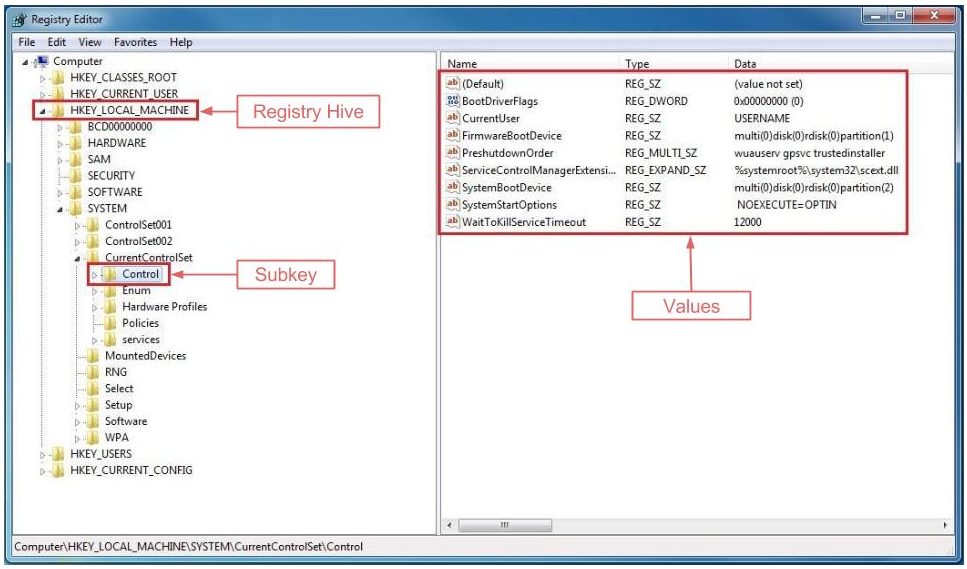

Figure 2.10: Registry keys and values. Here is an example view of the registry of a Windows 7 machine, viewed using the default Windows registry editor.

7. There is also a dedicated registry hive for each account on the system, stored in a file named "ntuser.dat" in the user's directory.

From the file system's point of view (and thus yours if you open the "system32\config" folder in Windows Explorer), these registry hives are just single files, but if you open them in a registry editor like "regedit.exe" you can view their internal structure. The internal structure of a registry hive is much like the directory structure of a file system. Instead of directories, registry hives have *keys* and instead of files they have *values*. Like directories and files, keys can contain values and/or subkeys. Like files, these keys can be stored in various formats, some are binary, some are plain text, some are hexadecimal. Registry keys have a "last written" timestamp that is not displayed in the default "regedit.exe" registry editor in Windows. You can view these timestamps in a tool like "Registry Commander," and they will be shown if you export the registry key to a plaintext file.

The importance of the Registry to computer forensics can hardly be overstated. Information about almost every Windows component can be found there. There are many many registry keys that are of potential interest to a computer forensics investigator, but we only have time to cover a few, so I will point out a couple interesting ones.

- Using these registry keys, you can see which user account was last

logged into the system, when they logged out, and which account was last used to shut the computer down. Particularly the first two are important, because they allow you to attribute events, modifications, etc. to a specific user account, rather than just a computer. Since computers often have multiple users, this step is one of the most important elements of the investigative process. To see who logged in last, you can view the

"Microsoft\Windows \CurrentVersion\Authentication\LogonUI"

key in the

"HKEY_LOCAL_MACHINE\SOFTWARE"

registry hive. The "LastLoggedOnUser" value of this key shows the username of the last user account used to log into the system. The "LastLoggedOnSAMUser" value of this key shows the domain and user id of the last user account used to log into the system. To see who last shut the computer down, you can view the

"CurrentControlSet\Control\Windows"

key in the

"HKEY_LOCAL_MACHINE\SYSTEM"

registry hive. The "ShutdownTime" value gives the time the system was last shut down, in binary, and the last write time on a user's "ntuser.dat" file indicates when that user logged off. These last two combined can give an indication of which user was last to shut the system off.

- Another key piece of information you can get from the registry is a list of recently attached USB devices. This information can be useful if you have not found an important USB device, you may need to look for it, or if you have the device, you can use the registry to establish a time that it was used by a specific account. The

  "CurrentControlSet \Enum\USBSTOR"

  subkey stores recently attached USB devices' information. The names of the first level subkeys under USBSTOR are device class identifiers taken from the device descriptions that identify the specific kind of USB device attached. The second level subkeys are the serial numbers that

uniquely identify the particular instance of the device (or if Windows cannot read the serial number of the device, a pseudorandom identifier. If the second character is an "&" it is indicative that the device does not have a serial number). So, two devices of the same type will have different unique ids but be under the same first level subkey.

To determine a device's installation time, you can search for the device's serial number in the "c:\Windows\setupapi.dev.log" file. This will show the time the device was first connected to the system.

If the system is running Windows Vista or 7, you can view the history of USB devices attached in the

"Microsoft\Windows Portable Devices\Devices"

subkey of the "HKEY_LOCAL_MACHINE\Software" hive along with their display name, the name that is shown in Windows Explorer when they are attached.

- The list of files last played in the Windows Media Player can be found in the

  "<sid>\Software\Microsoft\MediaPlayer\Player\RecentURLList"

  subkey of the "HKEY_USERS" hive, where sid is the security id of the user. These keys will display the full path to the last played file, showing if it was on one of the partitions of an internal drive in the computer or an external drive, and could give you a lead. If you don't find a file in that location, you can try to do deleted file recovery or file carving, and if it is from an external drive, you can look for external hard drive, which may contain additional evidence.

- The "Microsoft\Internet Explorer\TypedURLs"

  subkey of the "HKEY_USERS\SOFTWARE" hive lists the last 25 URLs typed into the address bar in Internet Explorer. This is slightly different than the internet history, which shows the recently visited website URLs. By showing the exact words typed into the address bar, you can clearly show that the user intended to navigate to the website, or search for those terms, rather than accidentally clicked on a link.

- Values under the "HKEY_LOCAL_MACHINE\SOFTWARE" key are

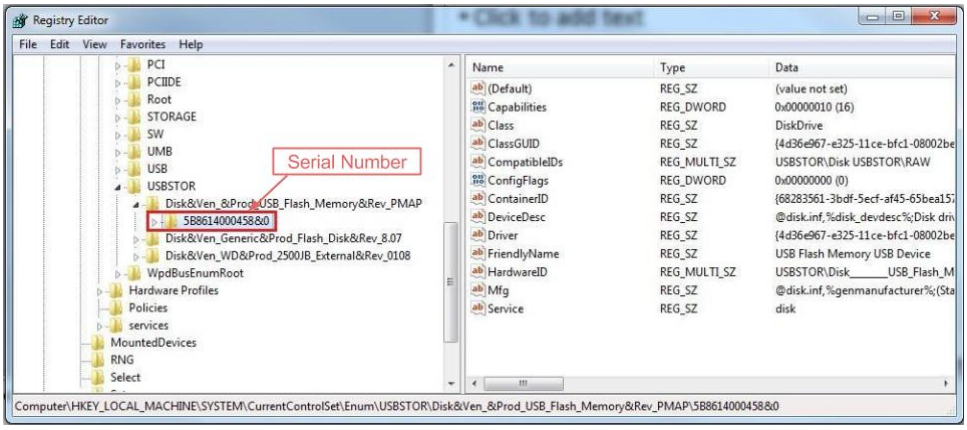

Figure 2.11: USBSTOR. Here you see an image of the USBSTOR registry key. Immediately below USBSTOR are 3 keys named for device types, and under those keys there are subkeys for the specific instances of those devices, named with their serial number.

usually created when programs are installed, but not always deleted when the program is uninstalled. The values under the

"Microsoft\Windows \CurrentVersion\App Paths"

subkey of the "HKEY_LOCAL_MACHINE\SOFTWARE" hive can provide clues to potentially incriminating applications that had been intentionally removed to evade detection.

The values under the

"Microsoft\Windows \CurrentVersion\Uninstall"

subkey of the " HKEY_LOCAL_MACHINE\SOFTWARE" hive show recently uninstalled applications.

Timestamps on these values can be used to corroborate timestamp evidence from file system forensics, or detect tampering if there are unexplained discrepancies.

2.4.5.2 Events Logs

In addition to the Registry, another very important source of computer forensics evidence on Windows systems is the Event Log. Windows offers a logging service to applications where they can register an event to be retained

by Windows in the Event Logs. Applications generally log events for troubleshooting and auditing purposes. The information in these events could be important, but is irrelevant to the user under normal circumstances. The more interesting event logs for a digital forensics investigator are usually those created by the system itself. Much of the important evidence contained in the event logs can also be found in the registry. Thus the event logs can be used to corroborate evidence found in the registry, but perhaps more importantly, discrepancy between the event logs and registry is an indication of evidence tampering.

Depending on the local security policy of the machine under investigation, there may be more or less events tracked in the logs. By default, home versions of Windows have most of their security auditing turned off, but Windows servers have most of it turned on. So Windows event logs often play an important role in investigating server-side incidents.

Before Windows Vista, the event logs were stored in a proprietary binary format. Now they are stored in an XML format with reasonably detailed public documentation. The event logs are now stored in files ending in the ".evtx" extension. The Registry links the .evtx files to DLLs (Dynamic Linked Libraries) containing message text, which together create the complete event log presented by the event viewer. In Windows Vista and 7 there are hundreds of event logs, each containing events listing date and time, user account and computer, an event ID and a description of the event. The description is constructed from registry entries and related DLL files, so when viewing an event log on a different system the descriptions might not be the same. For example, if login event logging is enabled in the security audit policy of the specific system in question, these events can be used to directly tell which account was logged in at what time, and should be validated with the registry items.

Since the event logs are not stored in simple plain text files, their forensic examination can be a little tricky in some cases. There are third party tools that can parse the .evtx files offline (e.g. on Linux), but they won't get the complete picture without the corresponding registry entries and DLLs. Security events are usually consistent within the same version of Windows (i.e. the message DLLs are the same), but the application logs are likely to have incompatible or missing message information because they depend on the application DLLs.

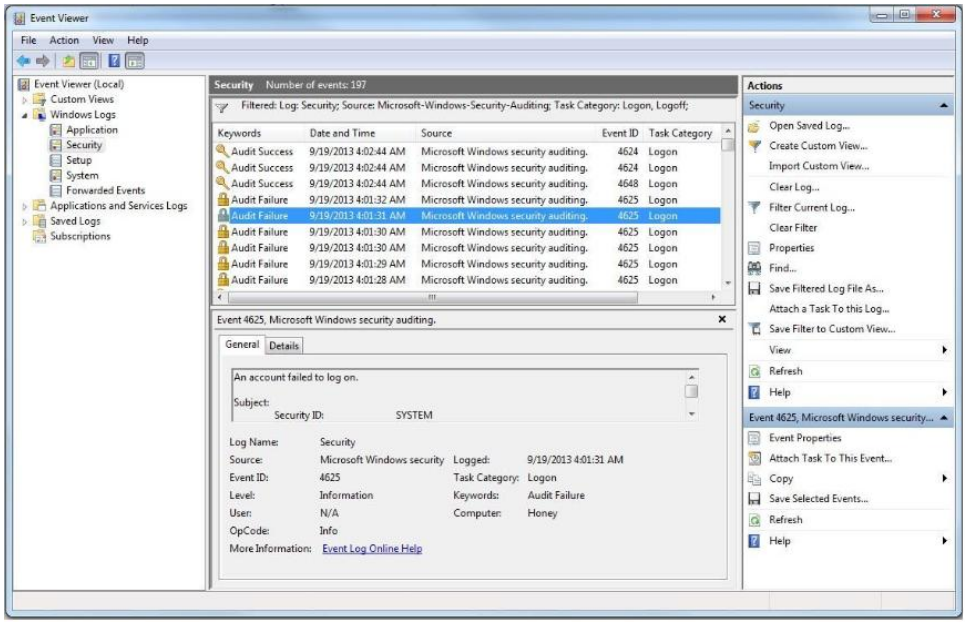

Figure 2.12: Login attempts. A classic example of an investigative use of the Windows Event Logs is tracking login attempts. Here you can see the security event log of a Windows 7 machine, with the log filtered to only show logon/off events. You can see a string of several failed log on attempts, which can be important information for an investigation, particularly when correlated with other events.

If you are analyzing a live system, you can collect the logs before shutting it down to guarantee the correct messages. If you are analyzing a "dead" (powered down) system, then you can copy the event logs to another Windows system of the same version to view the system events. However, if there is an important event log for an application you do not have access to, other than on the evidentiary machine, you can boot the forensic duplicate in a virtual machine. The details of virtual machines are beyond the scope of this course, but you can think of them as programs that pretend to be a computer to allow a disk image to be loaded into a controlled environment. If the evidentiary drive is loaded in a virtual environment, the application DLLs will be available in this environment to view the full event logs.

2.4.5.3 Link files

Another potentially interesting source of evidence in a computer forensics investigation is link files. Link files, or shortcuts, are files that point to other

files. They are created by Windows automatically in various circumstances, such as installing programs, but also track recently accessed folders. Link files can be important for a computer forensics investigation because they contain the full path to the file location and last accessed timestamps. The file path obviously helps find the actual file on disk, but it's especially important if the file is on an external device or remote drive. The link file will contain the full path with device serial number and volume label for the storage device containing the file. The timestamps on the shortcut are important too. A person might argue that a folder of illegal pictures was put on their computer by a virus, but if there is a link on their desktop to the folder that was clicked a few days ago, when the registry and log files indicate that they were logged into the computer, they will have a harder time denying knowledge of the files in court.

2.4.5.4 Recycle bin

Another interesting source of evidence on a Windows computer is the Recycle Bin. The Recycle Bin is just another folder from the point of view of the file system, but it is forensically interesting because of the misconceptions people have of it. What most people consider "deleting" a file is really just moving it to the Recycle Bin. What really happens when you click delete in Windows is the file record's file name and parent directory are updated.

There is a subdirectory in the Recycle Bin folder for each user account, named with the account's full security identifier. When a file is moved to the recycle bin, it is given a new file name starting with "$R" followed by a pseudorandom sequence of characters and the original file extension. There is also a file created with filename starting with "$I" followed by the same pseudorandom sequence of characters. This "$I" file contains original filename, the full path to the file's original location, and it's time of deletion. This renaming solves the problem of multiple files with the same name from different folders being sent to the recycle bin together, and the "$I" file allows the files to be restored if desired.

The presence of a file in the recycle bin indicates that the user attempted to delete the file. This can be significant legally if the file was relevant to the investigation, because it could constitute attempted destruction of evidence.

2.4.5.5 Sources

The primary sources for this lecture's material were Eoghan Casey's textbooks *Digital Evidence and Computer Crime* [28] and *Handbook of Digital Forensics and Investigation* [29]. Other sources consulted were [30] and [31]

2.4.6 Lecture: Windows Application Analysis

In the last lecture we discussed some potential sources of evidence related to the Microsoft Windows operating system. In this lecture, we will finish our discussion of computer forensics by discussing some evidence sources related to the applications running on Windows systems. It would be impossible to completely cover this topic in a single lecture, but hopefully this discussion will give you an idea of the wealth of information and potential evidence that can be gained by forensic examination of applications, by presenting specific examples and techniques that are particularly common and useful in computer forensics investigations.

2.4.6.1 Application Metadata

To understand our remaining topics, it is important to know the difference between application metadata and file system metadata. Recall from our previous discussion that file system metadata is stored by NTFS in the file's file record. It includes timestamp and ownership information, but treats the file's contents as a black-box. For example, an image file would be treated the same as a text document file. Unlike file system metadata, application metadata is stored with the file's actual contents, NOT in the file record. Since the application metadata is defined by each application itself, it is much more varied and often more detailed, and thus can contain a wealth of information for a DF investigator.

Application timestamps can provide a more accurate view in some cases. For example, if a file is copied to another device, it's file system metadata timestamps will likely be updated, showing the copy date as its creation date and last write date. However the application metadata timestamps will remain unchanged, thus giving the creation date of the file itself and last write date of its content. Note that both of these timestamps provide accurate,

useful information, but they describe different events. When analyzing application metadata, careful consideration must be paid to the circumstances and rules for updating the values.

While application metadata for most files can be easily altered by widely available tools, but it can be an excellent source of further leads in an investigation, and can still be useful as circumstantial evidence (to put pressure on a suspect to confess, for example).

A ubiquitous and illustrative example of application metadata is that of the Microsoft Office application suite. Office files have very rich application metadata that can be a treasure trove for a digital forensics investigator. Depending on the version and configuration, they can contain change records (if track changes is used), the last ten authors to edit the document (Note an "author" in Office is the name given to the application at time of installation), hidden annotations and comments, and of course, created, last written, and last accessed timestamps.

Another often important source of evidence from application metadata is the EXIF tags stored in digital image file formats such as TFF and JPEG. JPEG files actually contain an embedded TIFF file in their metadata, which in turn contains the EXIF metadata. EXIF metadata contains creation date and time, make and model of camera, camera settings (e.g. aperture, shutter speed etc.), a preview thumbnail, descriptions, copyright. and sometimes GPS coordinates. The GPS coordinates are only recorded if the camera is GPS enabled and configured to "Geotag" its photos. This would be a very restricting criteria, except that many iPhones and Android phones Geotag by default.

2.4.6.2 Web Browser Forensics

One of the most important sources of evidence in a computer forensics investigation is the web browser. A *web browser* is an application for displaying and navigating web pages. The browser sends a request to a web server for a particular web page and the web server sends a collection of files that the browser uses to construct the local version of the web page.

Web pages can have many different appearances, behaviors, and implementations, but typically they consist of a hypertext file and other supporting resource files and scripts. A *hypertext* file is a normal plain text document,

but with markup tags for formatting, dynamic behavior (e.g. html5), and famously: hyperlinks. *Hyperlinks* are elements in hypertext that contain a URL (universal resource locator). They can point to other hypertext files, images, or other websites. Most web pages contain dynamic content, built by live interaction with the web server. This is important to a forensic investigation because the dynamic content cannot be reconstructed from the local browser cache (at least not as easily as the static content). In general, you can recover the files, but they will not look exactly the same as when they were originally viewed by the user.

Browsers automatically store browsing history and cache viewed web pages. In Internet Explorer, browser history is organized in binary "index.dat" files that must interpreted by a forensics tool to be useful. These index.dat files also list the file location of the web page files downloaded by the browser in the "Temporary Internet Files" folder. You can reconstruct the browsing session from the cache, but it will not contain any of the dynamic content or server side scripts. This will enable you to partially reconstruct the users browsing activity. In Windows 7, the

"C:\Users\<username>\AppData\Local\Microsoft\Windows\
Temporary Internet Files\Low\Content\IE5\index.dat"

file contains browsing history with every file in cache listed along with browsing event that generated it. The

"C:\Documents and Settings\<username>\AppData\Local\Microsoft\
Windows\History\History.IE5\index.dat"

file lists the browsing history without all the cached files.

2.4.6.3 Email

Another potentially important source of evidence in a computer forensics investigation is email. Email messages are exchanged and stored by mail servers. Users interact with mail servers through mail clients or their web browser. Email works just like regular mail. The email server is analogous to the post office and your email client is analogous to your mailbox. To send mail you just write it and put it in the mailbox (click send) and the protocol takes care of the rest.

There are several different protocols used to make email work. There are different protocols used by mail servers to communicate with each other,

but for our discussion, we are concerned with the protocols clients use to communicate with the mail server.

- The simplest, and probably closest to physical mail, is the POP3 protocol. In POP3, your email client retrieves the email and it is deleted from the mail server.

- The IMAP protocol is different. The client reads the mail from the server, but does not delete it, until explicitly instructed to by the user. This would be like you going to the post office to get your mail, and bringing back copies, but leaving the originals.

- The third option is most likely what most of you use for your personal email: *webmail*. In webmail, there is no email client, just a web browser. The browser connects to a web page (e.g. mail.google.com) and communicates as it would with a normal web page.

For computer forensics, we are concerned with finding the email archives. The *email archive* is the local repository for the client's email messages. Depending on the type of client used, the location of the mail archives will be different, and they may not be stored on the computer at all. If they are stored on the local machine, they are usually found under the

"C:\Users\<username>\AppData\Roaming\"

folder in Windows. For example, the email archive for the "Thunderbird" email client on Windows 7 is located at

"C:\Users\<username>\AppData\Roaming\Thunderbird\Profiles
\<profilename>\Mail\Inbox"

The format of email archive files is usually plain text, and they can be viewed in a text editor. However, there are more advanced techniques and tools available for email forensics. Viewing even a small number of emails in a text editor is awkward and slow. There exist tools that take advantage of the index and table of contents files (depending on the client) to display the email in a more user friendly, easily navigable way. Most commercial forensics suits, like EnCase, can process email inboxes from every major client. Generally, email attachments are stored in-line with the rest of the email data, so can be recovered by a tool, or manually with a little manipulation.

2.4.6.4 Installed programs

Currently installed and recently removed programs can be an important clue during an investigation. Aside from the obvious case where you find pirated software or other illegal programs, traces of legal software that can be used for anti-forensics and evidence tampering is a good indication that you should be on the lookout for tampering, double checking your timestamps and evidence sources whenever possible. Often the primary benefit of knowing what programs are or were installed on a machine is not to directly give evidence of a crime, but determine what the machine is used for. For example, if you find only typical office applications like Microsoft Word and Excel, then you are probably dealing with a work computer; if you find applications related to hobbies, such as video games, then you are probably dealing with a personal computer. Perhaps none of these applications are illegal or used for illegal activities, but knowing what the computer is used for in general can guide your investigation.

As mentioned briefly in the previous lecture, installed programs are listed under the registry key

"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows \CurrentVersion\App Paths",

and recently uninstalled programs are located under the registry key

"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows \CurrentVersion\Uninstall".

You may also find evidence by searching allocated and/or unallocated space for the executable's file name, if known. If no attempt has been made to conceal the application, it will simply be listed under program files. Examining the creation and access times of the executable files can give installation and last execution times, respectively. These should be compared to the timestamps of the corresponding registry keys, if available, to detect possible tampering. You can sometimes also find traces of deleted applications configuration and temporary data under

"C:\Documents and Settings\<username>\Application Data",

"C:\Documents and Settings\<username>\local Settings \Application Data",

"C:\Users\<username>\AppData",

and "C:\ProgramData".

Often uninstall/cleaning utilities are sloppy in cleaning up these auxiliary folders, so evidence can remain long after the files in unallocated space have been overwritten and the registry entries removed. Note that there is an additional difficulty in implicating a specific person with this method. Programs that were installed by another user, but configured to be accessible to everyone will show up as installed under all users' directories.

There is another source of information on installed programs, and the fact that it is little known is sometimes forensically important. *Prefetch files* are used by Windows to expedite the start up of recently executed files. They are stored in the "C:/Windows/Prefetch" directory, and all have the extension ".pf". These files contain the path to the executable, the last run time, the number of times run from that location, and the names of the DLLs (external libraries) the executable requires. At most 128 prefetch files are stored at any one time, and they are retained based on most recently used. Thus, prefetch files give you an indication of what programs the user frequently used, in addition to their last executed times. Since they are less well known, and therefore less likely to be modified by a user attempting to hide evidence, they can be valuable to corroborate findings from other sources.

2.4.6.5 Sources

The primary sources for this lecture's material were Eoghan Casey's textbooks *Digital Evidence and Computer Crime* [28] and *Handbook of Digital Forensics and Investigation* [29]. Other sources consulted were [30], [32], [33], [34], and [35].

2.5 Module: Forensic Psychology

In this section I present a brief summary of the material from the two lectures covering forensic psychology. The instructor and subject matter expert for this module was Professor Masooda Bashir.

The overall purpose of this module is to give students a basic familiarity with the field of criminal psychology, particularly focusing on cyber crime and the psychological profiling of cyber criminals.

2.5.1 Key Concepts

- *Forensic psychology* is broadly defined as the application of the science of psychology to assist legal investigations or proceedings. A more concrete understanding of the field can be gained by considering some common roles and responsibilities of forensic psychologists:
  - Psychological Disorders and Offender Assessment
  - Punishment, Rehabilitation, and Assignment
  - Interviewing Suspects and Detecting Deception
  - Witness Evidence
  - Police Psychology
  - Decision-making Strategies of Juries
  - Crime Prevention and Cyber-Crime Victims
  - Research
  - Psychological Profiling
- *Criminal profiling* is the art and science of predicting the likely characteristics and future behavior of a suspect based on known case evidence and previous related cases. This lecture presents and compares both the *inductive* (attempt to match case evidence to pre-existing profile or template) and *deductive* (develop profile based on case evidence only) processes for building psychological profiles.

2.6 Module: Network Forensics

In this section, I present a brief summary of the material from the six lectures covering network forensics. The instructor and subject matter expert for this module was Dr. Faisal Syed.

The purpose of this module is to give students a high level overview of the field of network forensics, focusing on the principles underlying practice, rather than memorization of standard procedures.

2.6.1 Key Concepts

- *Networking Fundamentals*: This module begins with a brief review of some basic computer networking fundamentals, principally including basic network components, the OSI abstraction model, and basic TCI/IP protocol operation.

- *Network Evidence Acquisition*: This module also introduces some basic techniques for acquiring evidence from network devices. The volatility of many types of network based evidence makes their acquisition a strategic as well as technical challenge. Based on his or her current information, a network forensics investigator must prioritize their search for evidence based on both its expected importance and lifespan. Additionally, since network forensics investigation is often carried out on live networks, the investigators must take care to minimize disruption in the operation of the systems.

- *Packet Analysis*: This module also introduces some basic techniques for extracting evidence from captured packet data. Specifically, techniques for analyzing the protocol header fields, decoding and analyzing the protocols within packets, aggregating packets into streams, and reconstructing higher layer protocols' data from these streams.

- *Statistical Flow Analysis*: This module also presents an overview of statistical flow analysis, which gathers evidence about network events by considering aggregate statistics of flow records. Flow records generally contain the source, destination, start time, stop time, and data volume of each flow.

- *Network Intrusion Detection and Analysis*: This module also includes a brief introduction to Network Intrusion Detection Systems, including their basic operation and how they can be used in a network forensics investigation.

2.7 Module: Fraud Investigation

In this section, I present a brief summary of the material from the two lectures covering fraud examination. The instructor and subject matter expert for

this module was Professor Frank Nekrasz.

This module introduces the basic concepts of fraud investigation, to show students a different type of investigation in which digital evidence often plays a critical role.

2.7.1 Key Concepts

- *Fraud*: For the purposes of this work, let fraud be defined as follows "Fraud is any intentional act or omission designed to deceive others, resulting in the victim suffering a loss and/or the perpetrator achieving a gain [36]."

- *Introduction to Fraud Examination*: Fraud examination is the methodology of investigating an allegation of fraud. It involves all stages of the investigation, including collection and analysis of evidence, interviewing witnesses and potential suspects, writing investigating reports, and providing expert testimony. A fraud examiner is unlikely to be a digital forensics expert, but a digital forensics expert is often an important member of the fraud examination team.

- *Benford's Law*: One of the most important investigative techniques covered in this module uses Benford's Law: the principle that the distribution of digits in naturally occurring numbers follows a predicable, non-uniform distribution; where the first digits of numbers are much more likely to be lower valued (e.g. 1's and 2's) than higher valued (e.g. 8's and 9's). These distributions are found to hold quite accurately for real financial data, but when fraudsters alter numbers in financial documents, they rarely do so carefully enough to avoid upsetting the distribution of digits. Thus, a fraud examiner can often detect fraudulent financial records by computing the expected distribution of digits for a given financial document, and comparing it to the distribution present in the document.

2.8 Module: Mobile Device Forensics

This section presents the material for the mobile device forensics module in narrative form[4].

Mobile device forensics is the sub-discipline of digital forensics that deals with preservation, extraction, analysis, and investigation of digital evidence from mobile devices. Since they are by far the most common application of mobile device forensics skills, our discussion will focus on cell phones, but many of the same concepts and techniques can be applied to other mobile devices such as GPS navigation units.

While it has historically received very little focus in digital forensics education and training, mobile device forensics skills are becoming more and more important to a well rounded digital forensics professional for several key reasons.

- Almost everyone has a cell phone these days, so almost every case has a potential use for mobile device forensics.
- These devices are more strongly bound to a person's identity than a computer, because they are almost never shared between multiple users.
- It is more difficult to hide mobile evidence, since users do not typically have low level access to the device, and the underlying systems are esoteric and difficult to understand.
- Evidence from mobile devices can be corroborated with evidence from the service provider and computers the device was tethered to, increasing confidence in both.
- Mobile device evidence has rich location information, made all the more useful because most people always have their cell phone with them.

2.8.1 Lecture: Mobile Device Forensics I

In this lecture, we are going to begin our discussion of mobile device forensics, focusing on the evidence that can be acquired from the physical device, rather

[4]This section is presented in narrative form, so the second person (e.g. "you") indicates the students and the first person (e.g. "I") indicates the instructor.

than the network. We will briefly review some technology fundamentals, discuss flash memory and how it effects mobile device forensics investigations, and give a brief overview of the types of evidence that can be acquired from mobile phones.

2.8.1.1 Mobile Device Technology Fundamentals

Mobile devices are basically just miniature computers, with a CPU, RAM, and persistent storage (flash memory instead of hard disk). But to allow self-sufficient, mobile operation, they also typically have a battery, keypad, screen, and radio communication chip.

Mobile phones are often divided into two categories based on their general capabilities: Baseline phones, and "smartphones." The defining difference between a *smartphone* and a *baseline phone* is the ability to install third party applications. Thus, while baselines phones can only be used to a few standard functions such as text messaging and voice communication, smartphones can do almost anything a conventional computer can. Commonly the only limitation on a smartphone's capabilities (other than physical limitations like processing power, RAM, storage, and battery life), is the operation system enforced requirement that the applications be "signed" (i.e. cryptographically endorsed) by the vendor of the operating system (e.g. Android Play Store, or Apple App Store). This restriction can be, and often is, bypassed if the user uses a security exploit to "Jailbreak" or "Root" the device.

We will cover the potential investigative benefits of this ubiquitous network connectivity in next week's lecture on mobile network forensics, but today we will consider the consequences of this connectivity for a local examination of a device. Phones are often referred to as *GSM devices* or *CDMA devices* depending on the type of network communication chip they have. This difference is separate from the categorization of mobile network communication standards into so called "generations" according to access speed. Unlike general computers, mobile devices usually have a globally unique identification number. GSM devices each have a unique IMEI number (International Mobile Equipment Identity), and CDMA devices have a unique ESN number. Most devices will also have a manufacturer specified serial number. These identifiers can be important for a digital forensics investigation, because they serve to reliably identify a device, even if all the device data has been lost.

To access a GSM network, a device must also have a SIM (Subscriber Identity Module) card. SIM cards are actually tiny computers that have their own processor, RAM, and ROM (Read Only Memory, for persistent storage). This allows them to do relatively secure cryptographic authentication with the GSM network, without ever transferring the device's secret keys outside the SIM card. This is an important property, without which the SIM card could be easily copied by criminals who could then impersonate legitimate users for free network communication or concealing other criminal activities. SIM cards also contain a unique ICC-ID and subscriber identifier (IMSI) that can be used to distinguish them regardless of what phone they happen to be plugged into.

For context in our following discussion, it may be helpful to introduce some common mobile device operating systems. The most common smartphone operating systems are

- *Android*: an open source mobile operating system based on the Linux kernel.
- *iOS*: a proprietary mobile operating system for the Apple iPhone and other Apple mobile devices.
- *Windows Phone*: a lightweight, mobile version of the proprietary Microsoft Windows operating system.
- *BlackBerry*: a proprietary mobile operating system for BlackBerry devices.

By contrast, baseline phones often have simple proprietary operating systems and file systems developed in-house by the manufacturer. Many of these have been reverse engineered by forensics experts and can be processed by specialized mobile device forensics tools, but are difficult to interpret without such tools.

2.8.1.2 NAND Flash Memory

Unlike traditional computers which generally use fragile magnetic disks for persistent storage, mobile phones usually use some kind of more durable NAND flash memory for their non-volatile storage. NAND flash memory

uses the same physical mechanisms for storing data as Solid State Drives, and suffers from the same restrictions: data can only be erased in blocks and repeated writes damage memory cells.

For our purposes, the most important difference between NAND flash and SSDs is in the implementation of the Flash Translation Layer (FTL). The *Flash Translation Layer* (FTL) is the mechanism that handles garbage collection, wear leveling, and mapping from logical file structure to data layout in physical memory. SSDs are a type of *managed NAND* flash, because their FTL is handled by a smart controller on the flash device itself. The NAND flash chips on most mobile devices are *raw NAND* flash, because they only have a Program/Erase/Read (P/E/R) controller on the NAND flash device itself, and the FTL must be implemented in the file system and executed by the host device. This distinction has two important consequences for mobile device forensics:

1. The mapping from the logical location of the data (analogous to the "Cluster Address" from NTFS), to the actual data location on the memory chip is implemented in the mobile device, not the chip itself.

2. Garbage collection is much less likely to destroy deleted files. Depending on the implementation, the garbage collection process (erasing blocks marked as not in use) may only occur when data is written to the chip, or at least only when the operating system is loaded and the file system has been mounted.

The result of these two facts is that if you take what is called a "physical" image of the flash memory (to be covered in more detail later), bypassing the file system and directly reading the flash memory contents, the data returned will contain deleted file contents and will also be very difficult to interpret, because parts of files may be scattered depending on how the wear leveling was carried out.

2.8.1.3 Example Flash File System: YAFFS2 [37]

To solidify your understanding of these points, I will describe the basic operation of YAFFS2 (Yet Another Flash File System version 2). YAFFS2 is

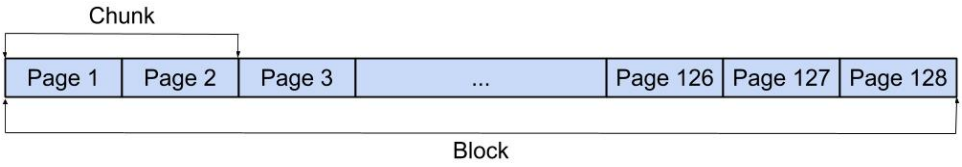

Figure 2.13: Example YAFFS2 memory unit sizes with two pages per chunk. This figure shows the relationship between *pages*, *chunks*, and *blocks*.

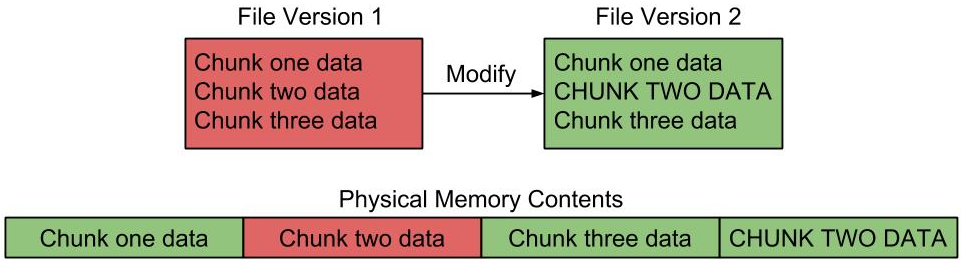

Figure 2.14: Simplified example of a file update operation in the YAFFS2 log-structured file system. The state of the "Physical Memory Contents" resulting from the file modification is shown.

used by a variety of devices, but most notably by Android smartphones. It was designed from the ground up to efficiently utilize NAND flash memory.

The basic unit of allocation of NAND flash as a called a *page*. That is to say, the flash memory you can assign to a file must be an integer number of pages (analogous to the sector for a magnetic disk). YAFFS2 defines its basic unit of allocation as a *chunk*. That is to say, the flash memory YAFFS2 can assign to a file must be an integer number of chunks. A chunk is typically simply 1 page, but it can be defined as multiple pages if required (analogous to the clusters for NTFS). As we discussed with SSDs, data in NAND flash can only be erased in *blocks*. Typically 32 to a few hundred chunks form a block.

Since flash memory does not allow overwriting of data without erasing the entire block, if a file is modified, you must allocate new chunks to store the changes. However, it would be very inefficient to copy the entire file's contents every time this happens, so YAFFS2 uses what's called a "log structure". Specifically, what YAFFS2 actually stores in the flash memory is a sequence of log entries specifying changes made to the files. Thus the order in which a file's data chunks are stored is based on the last time that particular part of the file was modified, not the actual ordering of the files contents.

Consider the example in figure 2.14. A file consisting of 3 chunks is saved

after modifying the second chunk's data. The modified chunk is written sequentially after the previously stored data in the log-structured file system, rather than overwriting the old chunk two data as NTFS would have done. This is obviously an extremely simplified overview of the operation of YAFFS2, and these chunks would be marked with sequence numbers and various metadata objects (not shown) to allow the file system to determine which chunks store the current file contents, but it illustrates two very important points about YAFFS2 (that are generally true of most flash file systems):

1. The physical contents of flash memory are often very difficult to interpret manually, requiring special tools to make sense of them.

2. As you can see, the chunks containing the old version of the file (with "Chunk two data") have not been overwritten, and are unlikely to be anytime soon, since the still current "Chunk one data" and "Chunk three data" are stored in the same block (much larger than a chunk). Thus YAFFS2 (and in general any log structured file system) leaves not just deleted files, but a very thorough record of many previous changes to a file.

As mentioned previously, the garbage collection in raw NAND flash must be implemented in the file system. The behavior of the garbage collector is of particular interest for mobile device forensics; I will describe the YAFFS2 implementation briefly to illustrate. The YAFFS2 garbage collector operates in one of two modes based on the number of free blocks available.

1. *Passive*: When it is not short on free blocks, it operates in passive mode. In passive mode, the garbage collector finds blocks with no in-use chunks or very few in-use chunks, and moves any in-use chunks to other blocks before erasing the block.

2. *Aggressive*: When free blocks become scarce, the garbage collector switches to aggressive mode. In aggressive mode, it selects more blocks with more in-use chunks, consolidates the in-use chunks in new blocks, and erases the old blocks.

2.8.1.4 Types of Evidence

Mobile devices offer a variety of evidence that can be very valuable for an investigation. Some of the most commonly useful types of mobile device evidence deserve special mention.

Call records are very useful since they provide exact times when individuals talked, and how long they talked. They are also very reliable, because they can be corroborated with records stored by the provider. The disadvantage is that they don't provide any direct information about WHAT was said.

SMS messages are particularly useful since they contain a full transcript of the conversation, and their timestamps are reliable, since they are added by the network provider, rather than the phone. The disadvantage of SMS messages is that they do not record when the user actually READ the message, only when the phone received it, and whether it had been subsequently opened. So they give time windows, such as "Bob became aware of X sometime between the time his phone received the message and when he replied to it."

Address books provide a convenient source of one of the most important types of investigative evidence: peoples' associates. This is especially critical in large investigations involving many individuals, such as organized crime.

Smartphones contain a wealth of potential evidence in *media files* (image, audio, video), GPS waypoints, email, Internet history, social network accounts, and potentially suspicious third-party applications. For example, the application "TigerText" is used to provide a secret text messaging service that can't be observed by the provider.

If a warrant can be obtained, the service provider also retains very useful *location* and *usage information*. For example, providers will typically record which cell towers the phone accessed at what times, which will give you an approximate movement history of the phone.

2.8.1.5 Sources

The primary source for this lecture's material was Eoghan Casey's textbook *Handbook of Digital Forensics and Investigation* [38].

2.8.2 Lecture: Mobile Device Forensics II

In this lecture, we will continue our discussion on mobile device forensics from last time. Today we will focus on preservation and acquisition of mobile device evidence, as these are the areas where mobile device forensics most differs from computer forensics.

2.8.2.1 Proper Handling of Evidentiary Mobile Devices

As with conventional computer forensics, steps should be taken to secure, evaluate, and document the scene before any evidence is collected. The same basic principles we discussed earlier apply to handling mobile device evidence. A complete chain of custody should be documented, only individuals with proper training should handle digital evidence, and care should be taken to prevent alteration of the evidence. However, to prevent alteration of the evidence, additional precautions need to be taken for mobile devices. Specifically the devices need to be isolated from the network. If the device is not isolated from the network, it will continue to receive communications that may alter or overwrite evidence on the device. The most damaging possibility is a remote wiping service. Many device manufacturers offer the option to send a command to the device over the network that will cause it to delete all its data, and thus much evidence.

2.8.2.2 Proper Network Isolation

To isolate a device from the network, you can remove the battery from the device (turning it off is not always sufficient), but it is often not desirable to turn mobile devices off, since forensic analysis of mobile devices is much more dependent on live analysis that computer forensics, as we will see later. The preferred method for network isolation is to place the device in a "Faraday cage" (typically a small bag lined with conductive material). A Faraday cage is a container that prevents any electromagnetic waves (including radio waves) from entering or leaving it. Note that once the device has been stored in the Faraday cage, it will continuously try to reconnect to the network, draining its battery very quickly. Thus to prevent loss of the evidence in the device memory, it is important to bring the device to the forensics lab

as soon as possible. Once in the lab, it should be take to a special room that is Faraday isolated and plugged in to a charger to keep it from dying. If no Faraday isolation bag is available, then placing the device in "airplane mode" is a common alternative, but this is not preferable because it requires interacting with the device's user interface. What is actually done in practice often depends on the available equipment as well as the regulations / standard practices for the organization acquiring the device. If the device is low on battery life, and you don't have a portable charger / battery pack, may be preferable to put it in airplane mode rather than a Faraday cage. If you don't have any Faraday isolated containers, obviously your choices are limited to turning the device off or putting it in airplane mode. Some standard practices differ depending on the type of device. For example, iPhones have mandatory hardware encryption of their NAND flash, so evidence must either be acquired using their file system interface, or the decryption keys must be recovered from RAM somehow. It is also important to find any removable media, such as MicroSD cards, and properly catalog and store them. This should be done in the lab, not at the scene.

Note: there is some disagreement in the literature and federal guidelines about whether it is better to turn a device off or place it in a Faraday cage [39], [40]. I present here the point of view that it is preferable to use the Faraday cage, because most criticisms of it revolve around the battery draining problem, which simply results in the phone shutting itself down, the same result as if you took the alternative and shut it down yourself.

2.8.2.3 Data Acquisition Layers

What is considered "best practice" is much less well defined for mobile device forensics than computer, or even network forensics. This is because unlike traditional computers, which generally use one of several well known operating systems, file systems, and hardware interfaces with well documented and often open source specifications, mobile devices usually use specialized, proprietary operating systems and file systems with special hardware interfaces. As a result, it is much more difficult to acquire evidence from mobile devices in a standardized, forensically sound manner. The techniques required vary widely based on the type and manufacturer of the device, but I will attempt to give you an idea of the general practices. Depending on the type and

manufacturer of the device, some of the techniques we will cover may not be possible, but even if they are, time considerations may prevent an investigator from using the most thorough and rigorous methods. Here I will give you an introduction to the general techniques for extracting information from mobile devices, sorted from fastest and least rigorous to slowest and most rigorous. As with computers, it is generally not advisable to operate directly on the evidentiary device, but there is not a universally accepted "correct" way to make a forensic duplicate of a mobile device.

The Scientific Working Group on Digital Evidence defines seven levels of access for mobile devices.

1. *Manual operation*: The easiest but least forensically rigorous method is to simply interact with the device through its user interface. This will inevitably result in alteration of the device's state, and only allows access to the data retrievable by the operating system. While this method may result in the loss of hidden data or deleted files, it is sufficient to retrieve the most common and important types of evidence, such as SMS message transcripts, call history, contacts, and installed applications. Also, this method will update the file system and application level access history and timestamps for everything you touch, so potentially important evidence such as if/when a user last accessed an application will be lost. This method is only possible if you can access the user interface, bypassing any security measures such as pass-codes, however it is usually very easy to circumvent the user level security on most phones. This process should be videotaped or photographed to at least ensure proper documentation.

2. *Logical acquisition*: A more forensically rigorous and time consuming method is so called "logical acquisition." Techniques in this category basically involve connecting to a communication port on the device and sending special commands to the operating system instructing it to output various pieces of information. Which of these techniques is possible and what data they can access is entirely dependent on the type of device and manufacturer and varies widely. You are restricted to only those commands supported by the operating system, and since the source code of these operating systems is generally not available, it is not possible to guarantee the integrity of the data returned, or that they

do not alter the state of the phone. An alternative methods for logical acquisition is a so called "Software Agent," a program which, when copied to the device and executed, will output all operating system accessible files (i.e. no deleted files).

3. *File System Access*: On some devices is is possible to acquire the entire contents of the file system, that is to say, every file the operating system can access, through a communication port. This depends on operating system and device driver support, but if available, it gives a very thorough but still easily readable view of the system. The downside to this method is that you are restricted to those files within the file system that the operating system can access. This excludes hidden areas, and depending on the way the device's operating system and file system handle them, deleted files. The distinction between logical and file system access can be a bit confusing; we will consider a logical acquisition to be essentially a partial file system acquisition, depending on the capabilities of the specific acquisition method.

4. *Physical acquisition (Non invasive)*: On some devices, it is possible to acquire the entire contents of RAM and non-volatile flash memory through the communication port or some proprietary interface. This method is very thorough and won't alter the device data, but the interpretation of the resulting binary file is quite difficult without a documented format (which is almost never available). The main benefit of physical acquisition is that it gets deleted files. Depending on the type of flash memory and file system used, often many deleted files are available. Another method used by some forensics tools to gain access to the full RAM and non-volatile flash memory is a *bootloader*. In this method, the forensics tool uses a security vulnerability to interrupt the device's startup process before the operating system is loaded, and insert a custom program that outputs the physical image of the device.

5. *Physical acquisition (invasive)*: It is also often possible to acquire a full copy of the entire contents of RAM and non-volatile flash using the device's JTAG port. JTAG is a standard protocol for debugging integrated circuits which allows a skilled (and very patient) user to arbitrarily access and edit the device memory and CPU state, and

it is enabled on most phones. This process will also allow you to recover deleted files. However, accessing the JTAG shift registers requires opening the device casing (and sometimes partially disassembling the device) and attaching leads to the internal circuit board. It also requires you to know which system processor and memory circuits are used and how they are connected on the system bus. You also need to know the location and functional mapping of the JTAG test points on the printed circuit board, the protocol for reading and writing memory, and the correct voltages to apply to the test points. As you can probably guess, this is not the most convenient process, and the resulting data will still need to be interpreted based on a, probably unknown, standard format. So, this is generally not done unless absolutely necessary.

6. *Chip-off access*: You can physically remove the flash memory chips from the device's circuit board and use custom hardware to directly read their contents. This method also allows you to recover deleted files. With this method it is always technically possible to read the contents of the device flash memory, but the device is permanently damaged and cannot be reassembled. Also, the contents of RAM are not acquired, and if the flash memory is encrypted, which is often the case, then you can't learn anything from it.

7. *Micro Read*: In the case where the phone has been physically damaged to the extent that the flash memory chips are partially destroyed, it is still possible to recover data by using an electron microscope to read the charge state of the individual flash memory cells. These microscopes are extremely expensive and are only available in a few digital forensics and research labs, so this type of analysis would only be undertaken for investigations of the most extreme importance (e.g. national security).

A common practice is to use a combination of methods: starting with a more forensically rigorous method (e.g. physical acquisition), and then performing a quick analysis by logical acquisition or manual operation. Thus, the original state of the device is preserved as much as possible, and the manual operation can be used to quickly find any "low-hanging fruit." The more rigorous evidence record can be used to verify the integrity of important

evidence discovered through the manual operation. The time consuming processes of exhaustive examination of the entire physical image of the device is rarely necessary.

There are various specialized digital forensics tools for mobile devices. Unlike computer and network forensics tools that typically all implement most the same functionality and distinguish themselves by their efficiency or ease of use, mobile device forensics tools are often very specialized and have very diverse capabilities. No one tool can be used for every device type.

2.8.2.4 Acquisition and Examination of SIM Cards

Sim cards contain several useful sources of evidence, as we discussed last time. Specifically, SIM cards contain the unique IDs for the SIM card and the user's account with the provider, an abbreviated contact list, and possibly SMS messages and recent call records. SIM cards have a standard format for storing their files, and several tools will automatically extract it (e.g. Forensic Card Reader, The Forensic SIM Toolkit, SIMCon, SIMIS, USIMdetective). SIM cards may be secured with a PIN to restrict access. Brute forcing this PIN can be problematic because the SIM card will lock after several (usually 3) failed attempts, but it is usually easy to get the unlock code from the network service provider.

2.8.2.5 Sources

The primary source for this lecture's material was Eoghan Casey's textbook *Handbook of Digital Forensics and Investigation* [38]. Other sources include [41], [40], [39], and [42].

2.8.3 Lecture: Mobile Network Forensics

In this lecture, I will briefly introduce some common digital forensics techniques for extracting evidence from mobile networks. In addition to providing useful supporting evidence for the type of investigations we have been focusing on so far, mobile network forensics techniques are often critical to coordinating investigation of large scale cases such as those involving international crime rings and terrorism. With the proper authorization, investi-

gators can gain access to the mobile network service provider's systems to extract evidence about the activities of suspects in real time. These techniques are closely related to wiretapping, and have similarly ambiguous legal restrictions and ethical considerations, that are important to keep in mind during an investigation. We will also introduce several useful techniques for investigative analysis, the final step in a digital forensics investigation before reporting.

2.8.3.1 Mobile Network Technology

Mobile networks are fundamentally based on radio frequency communications. This means that all devices within an area share the same physical transmission channel: the frequency band allocated for the mobile network.

Mobile networks can be broadly categorized based on the way they allocate access to this shared transmission channel:

- *Time Division Multiple Access* (TDMA) breaks transmission signal into multiple discrete time slots for each device and take turns using the shared channel.
- *Code Division Multiple Access* (CDMA) multiplexes different device signals into a one signal that is transmitted over the shared physical channel. The original signals are extracted from the shared signal using a code shared between the sender and receiver.

Mobile devices connect to the nearest *Base Transceiver Station* (BTS) over a radio link using TDMA or CDMA. The geographical region covered by a single BTS is called a *cell*. The company that operates this radio communication infrastructure is called the *Network Service Provider* (NSP).

2.8.3.2 Types of Mobile Network Evidence

Mobile network forensics allows for the easy acquisition of several very useful types of evidence:

- *Localization parameters*: information about the current or past locations of a mobile device

- *Usage logs / Billing records*: the NSP's internal business records of all the network activity of a device
- *Text / Multimedia messages*: NSP's often retain transient copies of text and multimedia messages on their servers
- *Intercepted data*: with proper authorization, investigators can intercept and record any communication over the mobile network

2.8.3.3 Localization Techniques

Probably the most important type of evidence gathered from mobile networks is location information. Location information can be important for assessing the alibis of suspects or whereabouts of victims. Mobile devices will connect to the network automatically unless their networking functions are disabled manually. Thus, they can be tracked by an investigator with access to the provider network even if the user is not making calls or using the phone at all.

The simplest and crudest way of tracking the location of a mobile device is by identifying the *cell* (BTS coverage area) that the device is currently connected to. This narrows the device location to the size of the cell, but that size varies greatly, from 32 km in rural zones to a few hundred meters in urban areas, depending on the density of BTS in the area.

This useful, but crude, estimate can be greatly improved by triangulation. In *Time Difference Of Arrival* (TDOA) analysis (a.k.a. *Multilateration*), the transmission latencies between the target device and several surrounding base stations are measured and compared to triangulate its position. Note that this requires three or more base stations to be within transmission range of the device.

2.8.3.4 Usage Logs / Billing Records

Another important type of evidence available in mobile networks is the usage logs of the NSP. Information contained in *Call Detail Records* (CDR) is combined into logs and provided to investigators. These logs will generally include the phone number of user, the phone numbers called, the IMEI/ESN number of the device, cell information (coarse location), SMS messages sent

(not including contents), date, time, and duration of calls, etc. While this information is generally available on the devices themselves, getting it from the NSP has several important advantages.

1. It would be extremely difficult for the mobile device's user to tamper with the NSP's usage logs to hide evidence
2. The NSP logs can be seized as soon as legal authorization is received, even while the mobile device's user is still at large.
3. The NSP has strong business incentives to diligently maintain its usage logs, while similar evidence on the mobile device will be progressively overwritten to save space.

2.8.3.5 Intercepted Data

An important, and controversial, source of evidence available in mobile networks is traffic interception: the capture of information in transit by a third party, using mechanical or electronic means, without the knowledge of the parties engaged in a supposedly private communication.

In response to an authorized request (according to the relevant local laws) by law enforcement, the NSP duplicates the suspect's communication line and directs it to a monitoring center (MC) operated by the investigators. This *monitoring center* is typically composed of an *interception server*, which collects and aggregates intercepted data, and *clients* that perform specific post-processing steps and display the results. Investigators operating the client machines get on-demand access to stored data and live conversations. These interception systems typically also run an analytics platform that processes all the unstructured data and allows for easier viewing and searching. For example, querying "rifle" also returns documents containing "AK-47" and querying "Larry Smith" will also return documents containing Larry's phone number.

Interception is a delicate topic, and frequently considered an invasion of privacy, but legal under certain restrictions. There are many laws and regulations designed to limit interception to protect individual privacy, but these laws and regulations vary greatly by country and region / state.

2.8.3.6 Guidelines for Mobile Network Interception

A detailed state-by-state or country-by-country examination of the laws and regulations governing traffic interception is beyond the scope of this course, but a knowledge of some basic guidelines should give you an idea of how interception can and cannot be used in an investigation.

Given the plethora of different network services and technologies provided by various entities, it important to define specifically from whose systems the data may be intercepted.

In general, *Network Service Providers* are required by law to retain certain records and provide them to law enforcement when given a warrant. However, it is not entirely clear what technically qualifies as a "Network Service Provider" and what records they are required to retain and provide.

While these definitions vary by state, there are some common qualifications that would exempt an organization from having to facilitate interception.

- Those entities offering electronic communication services directly to a limited group of people, rather than the general public, are generally not required to maintain records of the usage of those services. For example, a pottery club that runs a wireless network to give its members access to a shared printer.

- Those whose operations do not generate or process the relevant traffic data are not expected to retain it. For example, mobile phone manufacturers, such as Motorola, are not required to retain a database of call records from all their phones.

- Search engine administrators are, in some places, required to retain and submit users search records to authorities on a properly authorized request. This is a hotly debated issue. The search and browsing history of users is in many cases qualified as "content" since it easily allows reconstruction of exactly what the user was doing on the network, not just who they were communicating with, and in most countries this information is NOT required to be retained and is NOT subject to seizure by the authorities. However, in the USA it generally is.

If an entity qualifies as an NSP, they must retain, for the exclusive purposes of detecting and prosecuting crime, only the traffic data resulting from their technical operations providing and billing services.

Investigators also have their own set of obligations as to how intercepted data must be acquired and handled:

- Investigators must protect the data they acquire (i.e. when they are given private records, they are legally responsible for their continued confidentiality).
- They must acquire data in such a way as its integrity can be verified.
- They must avoid inflicting undue cost on the parties involved.
- They must respect any relevant transnational legislature.
- They must use the current state of the art techniques.

2.8.3.7 Privacy and Interception

One of the biggest concerns with mobile network interception is limiting violation of citizens' privacy. Many requirements have been enacted for technical controls, operational oversight, and auditing in an attempt to reduce the risk of abuse and detect any abuse quickly. Some example measures include:

- Authentication Systems must use "strong authentication" techniques (i.e. 2 different authentication technologies required for access).
- There must be a strict separation of technical functions for operators tasked with assigning credentials and those tasked with management of the systems and databases.
- Traffic data retained for investigative purposes should be stored on physically distinct systems from those used to process and store general traffic data.
- In many countries, regulations require that data be retained for a limited period then deleted without delay, including copies created for law enforcement in accordance with the law (not the USA though).
- An auditing system should be built into any system that operates on private user data, to facilitate oversight.

- All systems used for processing traffic should be thoroughly documented according to the accepted principles of software engineering. This description must include not only the system architecture but also the subjects or classes of subjects having legitimate access to the system and the exact positions in the network where data is gathered and processed.

- Confidentiality and integrity must be protected cryptographically by ensuring intercepted data is never transmitted or stored unencrypted.

2.8.3.8 Reconstruction Techniques [43]

Reconstruction is the process of combining various evidence and general case knowledge to form a picture of, or reconstruct, the past events relevant to the case. This is the final phase of the investigative process before reporting, and now that you have been introduced to some examination techniques and various types of evidence, you should have an idea of what you will be using to reconstruct events.

When attempting to reconstruct events, it is important to consider the independence of evidence sources. Since digital evidence is frequently circumstantial, multiple sources must be corroborated to draw conclusions confidently. However, two pieces of evidence should only be considered corroborating if they are drawn from independent events. In this context, "independent" basically means neither event caused of the other.

We will introduce three general techniques for reconstructing events to form useful intelligence.

1. *Temporal analysis* creates a timeline of events to help identify patterns and gaps. It is good practice to plot important events on a timeline as they are found, to organize information, make sure nothing is missed, and guide the investigation. Such a timeline often used to tie multiple sources of correlated information together into a single coherent picture, and is well suited for cases where a relatively small number of entities did many important things (i.e. ratio of important events to people involved is high).

2. *Relational analysis* focuses on geographic and communication / asso-

ciation relationships between important entities. This may be done by plotting individuals locations on an explicit map, or constructing a more abstract relationship graph. This type of graphical representation is useful when trying to organize many separate sources of information to reconstruct a complex event (i.e. the ratio of important events to important people involved is low).

3. *Functional analysis* focuses on determining how a particular function or program works, with the aim of understanding what it did under some specific circumstances in the past. For example, reverse engineering an unknown suspicious software package found on a phone to determine if it is eavesdropping malware. This is useful in determining the technicalities of exactly what a device did, why it did it, and whether the user was aware are important.

2.8.3.9 Sources

The primary source for this lecture's material was Eoghan Casey's textbook *Handbook of Digital Forensics and Investigation* [44].

2.9 Module: Malware Forensics

This section presents the material for the malware forensics module in narrative form[5].

Malware forensics is a sub-discipline of digital forensics that deals with the detection and analysis of malware.

Malware is a general term for software that performs some malicious function.

This course includes one lecture on malware, to familiarize the student with the terminology and basic concepts of malware forensics. A more in-depth treatment, examining the subtle science of reverse engineering malware and designing countermeasures, is beyond the scope of this course.

[5]This section is presented in narrative form, so the second person (e.g. "you") indicates the students and the first person (e.g. "I") indicates the instructor.

2.9.1 Lecture: Malware Forensics

In this lecture we will present a brief taxonomy of malware and introduce several common methods for malware detection. It is important for digital forensics examiners to understand the fundamentals of malware propagation and concealment techniques so they know the limitations of their detection methods and can accurately determine the condition of a system with these limitations.

The presence or absence, capabilities, and origin of malware are often critical to establishing alibis and intent in an investigation. A DF examiner should be appraised on the current state of the art in malware and anti-malware design, so he or she can:

1. Find malware if present
2. Determine the malware's origin
3. Determine the malware's purpose and past activities
4. Argue competently that a system is free of malware if they find none
5. Prevent malware from damaging evidence during an investigation

Techniques for removing malware are of only tangential interest to a DF examiner.

2.9.1.1 Types of Malware

There are many classifications and taxonomies of malware. We will discuss two that will be useful in characterizing malware behavior and capabilities.

The first classification distinguishes malware by *propagation mechanism*: the method they use to spread.

- *Viruses* propagate by "infecting" other files. The basic idea is that the virus adds a copy of itself to the victim file, in such a way that the virus code will be executed when the file is opened. Viruses use various methods to evade detection:
  - *Encrypted virus*: The virus code is encrypted using a different key each time it is copied. Key and small decryption subroutine are stored with virus

  - *Polymorphic virus*: Similar to an encrypted virus, but code "mutates" every time it is copied, changing appearance (e.g. swap order of independent instructions, change "2+2" to "5-1", etc.). The behavior of the virus does not change, and the part of the code that does the "mutation," the *polymorphic engine*, is not itself mutated.
  - *Metamorphic virus*: Like polymorphic virus but the code that does the "mutation," the *metamorphic engine*, is itself mutated.

- *Worms* propagate by exploiting vulnerabilities in software to gain unauthorized control over victim systems. Once the victim is infected, worms may use concealment methods similar to viruses to avoid detection. In the 2000's, worms were extremely prolific (e.g. SQL Slammer infected 75,000 hosts in 10 minutes [45]).

  Worms use many methods to gain access to victim machines. Some examples:

  - Email/IM: Worm sends itself as an attachment
  - USB: Worm copies itself to a USB drive and infects autorun (optionally)
  - Remote transfer/execution: worm exploits vulnerabilities in network services to gain access
  - Remote login: worm logs in to remote server, transfers a copy of itself, and executes the copy
  - Drive-by downloads: worm is embedded in in a web page/script. Exploits vulnerability in web browser

  Modern worms are often "delivery vehicles" for other malware. That is to say, the entire purpose of the worm is to download and install other malware on the victim machines. This type of worm has become known as a *dropper*.

- *Trojans* propagate using social engineering techniques to masquerade as a useful program, thus tricking users into downloading and running it. Once the trojan has been downloaded and installed, it's behavior generally falls into one of three categories:

  - *Additional behavior*: the trojan performs the function it advertised to the user to get it download it, plus hidden malicious functions.
  - *Modified behavior*: the trojan performs a modified, malicious version of the advertised function.
  - *Replaced behavior*: the trojan's behavior is totally malicious and unrelated to the advertised function.

The second classification distinguishes malware by *payload*: the actions they take (other than propagation).

- *System corruption* payloads are designed to cause damage to the victim system. This often takes the form of deletion or modification of important files, or defacing a public website, as a prank or to show off. The purpose of the payload was to publicly demonstrate that the system was compromised. These types of attacks were more common in the '90s and early 2000's when cybercrime was less professionalized. More often now these types of attacks are motivated by profit. *Ransomware*, for example, encrypts users' files and attempts to extort money from them to get the decryption key. Some rather impressive malware has even been employed to damage real-world facilities (e.g. the famous "Stuxnet" malware that targeted the centrifuges in the Iranian nuclear program).
- A *Botnet* is a collection of infected hosts, or *bots*, all controlled remotely by the botnet's "owner" or by someone who paid the botnet owner to rent the botnet. The payload of a botnet malware instructs the victim to connect to the designated botnet controller and await further instructions. Generally botnet malware tries to not disrupt the victim machine, only "borrows" it when desired to perform designated tasks en mass. Botnets are immensely useful because of their massive compute/transmission power and their inherent anonymity.
  - Sending spam
  - Launching new worms quickly
  - Distributed Denial of Service attacks (DDOS): overload a victim network with traffic

  - BitCoin mining
  - Clickjacking: botnet owner hosts a website with ads that pay per click. Makes his bots click them
  - Manipulating online polls/games
  - Brute forcing encryption/password hashes

- *Information theft* malware (keyloggers, phishing, spyware) is a notorious and prolific malware payload type in the USA and western Europe. These malware variants target several common types of information:

  - *Identity theft*: important financial information is stolen such as Credit card numbers, bank passwords, and Social Security Numbers, by logging keystrokes, for example.
  - *Spouseware* is a new type of spyware marketed to suspicious spouses. The suspicious spouse purchases the spouseware and installs it on his or her partner's device (typically smartphone) as if it were normal software. The spouseware eavesdrops on conversations, tracks location, forwards emails and text messages, etc.
  - *Corporate* or *national espionage* is a less common (but much more exciting) application of information theft malware.

- *Rootkit* payloads are designed to maintain root access and avoid detection. Since they have root privileged access to the system, rootkits can do almost anything on the infected host, limited practically only by the authors limited foreknowledge of what will be required to evade detection. Some example methods used by rootkits to evade detection are:

  - Rootkits will intercept and modify calls to system functions during scans.
  - Some rootkits start up before the OS, and run the entire operating system as a virtual machine.
  - Some rootkits infect the motherboard's BIOS, to place itself between the OS and the hardware platform.

Note that these categorizations are orthogonal. So, for example, a *worm* (propagation method) can be a *keylogger* (payload). The categories are not mutually exclusive either. Some kinds of malware use multiple methods to propagate based on their situation, and it is common for a malware payload to have more than one function. Still, these categories are commonly used when discussing malware, so it is important to understand what they mean.

2.9.1.2 Countermeasures

So far this all sounds quite dismal, but don't despair, anti-malware technology offers fairly robust countermeasures. There are four high level objectives one could focus on when implementing malware countermeasures:

- *Prevention*: block the malware's propagation mechanism. The ideal goal, but also very difficult.
- *Detection*: determine with certainty whether a system has been compromised or not. Easiest to achieve, and least useful.
- *Identification*: determine the nature of a compromise if found, either identifying a known malware or reverse engineering the behavior of a new one.
- *Removal*: completely expiate all traces of the malware. This may be more or less difficult depending on the malware, but it is always hard to know if you succeeded.

2.9.1.3 Malware Scanners

The most common type of malware countermeasure is, of course, the ubiquitous *malware scanner* or *antivirus scanner*. Modern antivirus suites are fairly sophisticated systems, but they were built up incrementally from simple roots (no pun intended). I will present this explanation chronologically based on succeeding "generations" of malware scanning techniques.

- *First generation scanners* use static signature matching to identify malware. This is the most obvious, simplest, and for many cases still the most effective method of detecting malware.

This type of scanning is fast, efficient, and easy to implementation. The efficiency makes this the de-facto choice for scanning large collections of data, and the simplicity means less development cost and software bugs. However, it can only detect known malware variants, which must contain a sufficiently rare, preferably unique, bit pattern in its files. There are many ways for malware to exploit this weakness (e.g. polymorphic viruses).

- *Second generation scanners* evaluate files based on heuristics (like rules-of-thumb).

  This is a slight improvement when used *together* with static signature scanning. Many of these heuristics are your kind of "Duct Tape" solution to the encrypted virus and other early malware concealment techniques. For example, to find a polymorphic virus, look for an encryption key at the beginning of a loop, decrypt the file, and check it against a static signature database to identify and remove it. These heuristics are more difficult to implement than first generation scanners and result in lots of false positives, however they are sometimes able to detect previously unseen malware, and resist simple concealment techniques like encrypted viruses and polymorphic viruses.

- *Third generation scanners*, also known as activity traps, identify a small set of actions that indicate the start of some malicious activity. They run in the background and monitor for these actions, and usually prevents the action from continuing and prompt the user to examine the activity and give explicit permission to override the alert. This protection is limited however, because individual actions are usually not clearly malicious. It isn't until several individual operating system function calls have been executed that the overall intent becomes clear. Thus, some damage may be done before the activity trap reacts and blocks the malware. Third generation scanners don't need a static signature for every malware type, only behavior patterns for the few types of malicious activity malware typically engage in. They pay for this in performance and convenience though. The program must always be running in the background and may consume lots of resources. Also, if the configuration and heuristics are not tuned carefully, it can result in

many annoying false positive alerts, and users will habitually override them, making the system useless.

- *Fourth generation scanners* use a combination of techniques in a comprehensive security suite usually including antivirus (signature/heuristic scanner), firewall, access controls, activity trap, and generic decryption. This is probably what most of you use. Common examples include Norton, Kaspersky, Avira, and Comodo.

  Generic decryption is a technique for malware detection and containment that exploits the fact that poly/metamorphic malware must decrypt itself at some point before executing its payload. To determine if an executable file contains malware, the generic decryption scanner "simulates" the execution of the file and scans the resulting memory space for known malware signatures. This "simulation" computes the results of the instructions in the executable code, but it is carried out entirely under the control of the generic decryption scanner, so the potentially malicious file is never actually given control of the computer. This scanner periodically pauses the simulation and scans the accumulated simulated memory space for any traces of known malware. This works great, but there are a couple major drawbacks: first, it only works on known malware variants; and second, the malware will only be detected if it decrypts its payload soon after execution begins and without outside input. If the malware waits to decrypt its payload a significant time after its execution, or waits for some external signal before triggering, then the scanner will not detect it.

2.9.1.4 Dealing With Rootkits

A *rootkit* is a particular class of stealth malware that somehow gains superuser or "root" privileges on the infected host. A program with root privileges can do almost anything on the host, including reading and modifying any file, access to and control of the operating system's utilities, and reading and modifying the process execution state of other programs on the same host. Heavy-handed application of these powers will quickly lead to system failures and detection, but the primary theoretical limit on the power of an installed rootkit is the ingenuity and foresight of its author.

Rootkits are especially difficult to detect, and almost impossible to reliably remove, because they can modify/control the very utilities that are attempting to find them. For example, a common technique for hiding rootkits is to intercept the operating system API call that reads files, and if the requested file is related to the rootkit, it simply returns nothing; with the end result that the rootkit is "invisible" from any program running under the OS's supervision.

While they are generally more specialized and less reliable than the techniques we have already discussed, there are methods capable of detecting known rootkits. For example, to detect a rootkit employing the above technique you could scan the entire system using the operating system API calls, then scan the entire system by directly interfacing with the hard drive, bypassing the OS API calls. Thus if there is a rootkit intercepting OS API calls that read its files, the two scan results produced will be different. Note that this is not foolproof, and a rootkit could easily be written to evade it. This gives you an idea of the constant malware vs. anti-malware arms race going on. Anti-virus developers will detect a new type of malware and update their systems to detect and remove it, then the malware authors will update their malware to circumvent the anti-virus detection and removal techniques.

2.9.1.5 Sources

The primary source for this lecture's material was Stallings and Brown's textbook *Computer Security: Principles and Practice* [46].

CHAPTER 3

CONCLUSION

Although still a new discipline, digital forensics has already become an important resource in our legal justice system. In addition to providing evidence in many types of investigation, digital forensics skills find broad application in areas such as data recovery and diagnostics.

The rapid growth of the field has imposed several significant difficulties on the digital forensics education community.

- Not least of these difficulties is the pressing need for standardization imposed by the importance of the field and the need to establish credibility within the forensic science community, even though the theory and practice of digital forensics are still rapidly evolving.

- Demand for continuing professional education and certification puts pressure on educators to develop training based programs that teach digital forensics as a stepwise laboratory procedure, neglecting the theoretical foundations of the techniques. While these type of programs are valuable for their part, without giving the next generation of community leaders a strong theoretical foundation, we cannot hope to secure a lasting position for digital forensics as a legitimate scientific discipline.

- Digital forensics is currently suffering from a lack of appropriate textbooks for a course in higher education. While there are many excellent resources available for practitioners wishing to update their skills, a textbook with an in-depth coverage of the foundational concepts and technology for digital forensics has yet to be written [5].

- The lack of adequate textbook resources increases the reliance of digital forensics education on the personal experience of the instructor, and finding qualified instructors is one of the most difficult challenges in establishing a digital forensics education program [10], [5].

- The interdisciplinary nature of digital forensics makes selecting appropriate prerequisites difficult. Digital forensics techniques have natural knowledge prerequisites in computer science and law, but incoming students are unlikely to have a background in both.

To help address some of these difficulties, we are developing a new certificate program in digital forensics. When completed, this program will consist of an introductory course and an advanced course with accompanying laboratory sessions. Our curriculum is designed to be easily adopted by other institutions, and we plan to distribute it as a curriculum package including everything a computer science professor would need to teach the course. We intend to continue revising and improving our curriculum based on feedback from the digital forensics research and education communities, student responses, and future technological developments. Thus we can simultaneously address unforeseen logistic problems we hear about and improve our curriculum to more closely meet the community's needs.

At the time of writing, we have developed the curriculum for our introductory course and taught a pilot class. The curriculum for the advanced course is currently under development, and we intend to offer an online version of both courses in the future.

REFERENCES

[1] A. Lang, R. Campbell, M. Bashir, and L. DeStefano, "Developing a new digital forensics curriculum," *Digital Investigation*, vol. 11, 2014.

[2] D. Wassenaar, D. Woo, and P. Wu, "A certificate program in computer forensics," *Journal of Computing Sciences in Colleges*, vol. 24, no. 4, pp. 158–167, Apr. 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1516546.1516575

[3] H. Chi, F. Dix-Richardson, and D. Evans, "Designing a computer forensics concentration for cross-disciplinary undergraduate students," in *Proceedings of the 2010 Information Security Curriculum Development Conference*, ser. InfoSecCD '10. New York, NY, USA: ACM, 2010. [Online]. Available: http://doi.acm.org/10.1145/1940941.1940956 pp. 52–57.

[4] S. Srinivasan, "Computer forensics curriculum in security education," in *Proceedings of the 2009 Information Security Curriculum Development Conference*, ser. InfoSecCD '09. New York, NY, USA: ACM, 2009. [Online]. Available: http://doi.acm.org/10.1145/1940976.1940985 pp. 32–36.

[5] J. Liu, "Developing an innovative baccalaureate program in computer forensics," in *Proceedings of the 36th Annual Frontiers in Education Conference*, 2006, pp. 1–6.

[6] J. Liu, "Implementing a baccalaureate program in computer forensics," *Journal of Computing Sciences in Colleges*, vol. 25, no. 3, pp. 101–109, Jan. 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1629116.1629134

[7] Forensic Science Education Programs Accreditation Commission, "FEPAC Accreditation Standards," American Academy of Forensic Sciences, Tech. Rep., November 2012.

[8] Forensic Science Education Programs Accreditation Commission, "Accredited universities," 2014. [Online]. Available: http://fepac-edu.org/accredited-universities

[9] P. Cooper, G. T. Finley, and P. Kaskenpalo, "Towards standards in digital forensics education," in *Proceedings of the 2010 ITiCSE Working Group Reports*, ser. ITiCSE-WGR '10. New York, NY, USA: ACM, 2010. [Online]. Available: http://doi.acm.org/10.1145/1971681.1971688 pp. 87–95.

[10] L. Gottschalk, J. Liu, B. Dathan, S. Fitzgerald, and M. Stein, "Computer forensics programs in higher education: A preliminary study," *SIGCSE Bull.*, vol. 37, no. 1, pp. 147–151, Feb. 2005. [Online]. Available: http://doi.acm.org/10.1145/1047124.1047403

[11] ACM/IEEE-CS Joint Task Force on Computing Curricula, "Computer Science Curricula 2013," ACM Press and IEEE Computer Society Press, Tech. Rep., December 2013. [Online]. Available: http://dx.doi.org/10.1145/2534860

[12] West Virginia University Forensic Science Initiative, "Technical Working Group for Education and Training in Digital Forensics," United States Department of Justice, Tech. Rep., August 2007. [Online]. Available: https://www.ncjrs.gov/pdffiles1/nij/grants/219380.pdf

[13] Scientific Working Group on Digital Evidence, "SWGDE/SWGIT Guidelines and Recommendations for Training in Digital and Multimedia Evidence," Scientific Working Group on Digital Evidence, Tech. Rep., January 2010. [Online]. Available: https://www.swgde.org/documents/Current Documents

[14] Illinois Science, Technology, Engineering, and Mathematics Initiative, "Digital Forensics Course Evaluation Report," University of Illinois at Urbana Champaign, Tech. Rep., January 2014.

[15] K. Popper, *The Logic of Scientific Discovery*. Routledge, 2005.

[16] E. Casey, *Digital Evidence and Computer Crime*. Waltham, MA: Academic Press (Elsevier), 2011, ch. 1.

[17] E. Casey, *Digital Evidence and Computer Crime*. Waltham, MA: Academic Press (Elsevier), 2011, ch. 3.

[18] E. Casey and B. Schatz, *Digital Evidence and Computer Crime*. Waltham, MA: Academic Press (Elsevier), 2011, ch. 6.

[19] E. Casey, *Handbook of Digital Forensics and Investigation*. Waltham, MA: Academic Press (Elsevier), 2010, ch. 1.

[20] E. Casey and C. Rose, *Handbook of Digital Forensics and Investigation*. Waltham, MA: Academic Press (Elsevier), 2010, ch. 2.

[21] C. L. I. Institute, "Daubert standard," 2014. [Online]. Available: http://www.law.cornell.edu/wex/daubert_standard

[22] Technical Working Group for Digital Evidence in the Courtroom, "Digital evidence in teh courtroom: A guide for law enforcement and prosecutors," National Institute of Justice, DC, Tech. Rep. NCJ 211314, Jan 2007.

[23] M. Gupta, M. Hoeschele, and M. Rogers, "Hidden disk areas: Hpa and dco," *International Journal of Digital Evidence*, vol. 5, no. 1, pp. 1–8, 2006.

[24] B. Carrier, *File System Forensic Analysis*. Upper Saddle River, NJ: Addison-Wesley (Pearson), 2005.

[25] S. L. Garfinkel, "Carving contiguous and fragmented files with fast object validation," *Digital Investigation*, vol. 4, pp. 2–12, 2007.

[26] A. Pal and N. Memon, "The evolution of file carving," *Signal Processing Magazine, IEEE*, vol. 26, no. 2, pp. 59–71, 2009.

[27] Y. Gubanov and O. Afonin, "Why ssd drives destroy court evidence, and what can be done about it," *Forensic Focus*, 2012.

[28] E. Casey, *Digital Evidence and Computer Crime*. Waltham, MA: Academic Press (Elsevier), 2011, ch. 17.

[29] R. Pittman and D. Shaver, *Handbook of Digital Forensics and Investigation*. Waltham, MA: Academic Press (Elsevier), 2010, ch. 5.

[30] D. Farmer, "A forensic analysis of the windows registry," *Forensic Focus*, 2009.

[31] T. Larson, "Digital forensics and windows 7 event logs," 2011. [Online]. Available: http://www.slideshare.net/ctin/windows-7-forensics-event-logsdtlr3#btnNext

[32] K. J. Jones and R. Belani, "Web browser forensics, part 1," *SecurityFocus*.

[33] K. J. Jones and R. Belani, "Web browser forensics, part 2," *SecurityFocus*.

[34] K. Jithra, "Microsoft office security, part 2," *SecurityFocus*.

[35] *Exchangeable image file format for digital still cameras*, exifStandard2.pdf, Japan Electronics and Information Technology Industries Association Std. 2.2, 2002. [Online]. Available: http://www.kodak.com/global/plugins/acrobat/en/service/digCam/exifStandard2.pdf

[36] The Institute of Internal Auditors, "Managing the business risk of fraud: A practical guide," April 2013.

[37] C. Manning, "How yaffs works," 2012. [Online]. Available: http://www.yaffs.net/documents/how-yaffs-works

[38] E. Casey and B. Turnbull, *Handbook of Digital Forensics and Investigation.* Waltham, MA: Academic Press (Elsevier), 2010, ch. 20.

[39] W. Jansen and R. Ayers, "Guidlines on cell phone forensics: Recommendations of the national institute of standards and technology," NIST, MD, Tech. Rep. 800-101, May 2007.

[40] C. Murphy, "Developing process for mobile device forensics," 2013. [Online]. Available: http://www.mobileforensicscentral.com/mfc/documents/Mobile%20Device%20Forensic%20Process%20v3.0.pdf

[41] Scientific Working Group on Digital Evidence, "Swgde best practices for mobile phone forensics," Scientific Working Group on Digital Evidence, Tech. Rep., February 2013. [Online]. Available: https://www.swgde.org/documents/Current Documents

[42] S. Willassen, "Forensic analysis of mobile phone internal memory," in *Advances in Digital Forensics.* Springer, 2005, pp. 191–204.

[43] E. Casey, *Digital Evidence and Computer Crime.* Waltham, MA: Academic Press (Elsevier), 2011, ch. 16.

[44] D. Forte and A. de Donno, *Handbook of Digital Forensics and Investigation.* Waltham, MA: Academic Press (Elsevier), 2010, ch. 10.

[45] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "The spread of the sapphire/slammer worm," 2003. [Online]. Available: http://www.caida.org/publications/papers/2003/sapphire/sapphire.html

[46] W. Stallings and L. Brown, *Computer Security.* Boston, MA: Pearson Education, 2012, ch. 6.