

**XML Accounting Trail: A model for introducing digital forensic  
readiness to XML Accounting and XBRL**

by

Dirk Jacobus Johannes Kotze

**Submitted in fulfilment of the requirements for the degree**

Magister Scientiae

in the Faculty of

**Engineering, Built Environment, and Information Technology, University  
of Pretoria**

March 2015



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Electronic Financial Data . . . . .	1
1.2	The Case for Digital Forensics and Forensic Readiness . . . . .	2
1.3	Scope and Purpose . . . . .	3
1.3.1	Forensics in the Context of XML Financial Information . . . . .	3
1.3.2	Problem Statement . . . . .	4
1.3.3	Research Method . . . . .	4
1.3.4	Contribution . . . . .	5
1.4	Approach . . . . .	6
<b>2</b>	<b>Digital Forensics</b>	<b>9</b>
2.1	Introduction . . . . .	9
2.2	Definition of Digital Forensics . . . . .	10
2.3	The Digital Forensic Process . . . . .	10
2.3.1	Phases of a Forensic Investigation . . . . .	11
2.3.2	Economics of a Forensic Investigation . . . . .	13
2.3.3	Forensic Readiness . . . . .	15
2.4	Digital Forensic Practice . . . . .	18
2.4.1	Regulatory Requirements . . . . .	19
2.4.2	Evidence: What Constitutes Evidence and How Should it be Handled? . . . . .	20
2.4.3	Evidence certainty . . . . .	24
2.5	Conclusion . . . . .	27
<b>3</b>	<b>XML and XML-based Accounting Formats</b>	<b>29</b>
3.1	Introduction . . . . .	29
3.2	What is XML? . . . . .	30
3.2.1	Overview of XML . . . . .	30
3.2.2	Example of XML Usage . . . . .	31
3.2.3	Why Use XML? . . . . .	32
3.3	XML Finance and Accounting Formats . . . . .	35
3.4	Concerns Around XML Usage - How XML's Design Provides Opportunity for Data Manipulation . . . . .	38
3.5	Conclusion . . . . .	39

<b>4</b>	<b>Compilers and Compiler Theory</b>	<b>41</b>
4.1	Introduction . . . . .	41
4.2	Analysis of the Source Program . . . . .	42
4.2.1	Understanding the Input . . . . .	43
4.2.2	Lexical Analysis . . . . .	44
4.2.3	Syntax Analysis . . . . .	45
4.2.4	Semantic Analysis . . . . .	45
4.3	Errors and Error Handling . . . . .	46
4.3.1	Errors . . . . .	46
4.3.2	Strategies to Handle Errors . . . . .	48
4.4	Conclusion . . . . .	49
<b>5</b>	<b>Devising a Method for Detecting Data Irregularities</b>	<b>51</b>
5.1	Introduction . . . . .	51
5.2	Analysing XML files . . . . .	53
5.3	Finding Data Irregularities . . . . .	55
5.3.1	The Large Data Set Approach Used by Anti-virus Engines to Identify Data Irregularities . . . . .	56
5.3.2	Using Automated Search Techniques to Locate Data Irregularities . . . . .	57
5.3.3	Utilising Error Handling Strategies to Locate Data Irregularities . . . . .	59
5.3.4	Applying the Compiler to Locate Data Irregularities . . . . .	61
5.4	Application of and Benefits Derived from the Use of a Compiler for Detecting Data Irregularities . . . . .	63
5.5	Conclusion . . . . .	69
<b>6</b>	<b>Moving from Evidence to Hypothesis</b>	<b>71</b>
6.1	Introduction . . . . .	71
6.2	Scientific Methods and Digital Forensics . . . . .	73
6.2.1	Using Scientific Method as a Criterion for Admissibility of Evidence in a Court of Law . . . . .	75
6.2.2	Applying the Standards of Admissibility and Science to Digital Forensic Evidence . . . . .	76
6.3	Applying Scientific Method to the Detection of Data Irregularities . . . . .	78
6.3.1	A Set of Hypotheses Based on the Evidence Gathered Using Compilers to Detect Data Irregularities . . . . .	81
6.3.2	Applying Scientific Rigour to the Set of Hypotheses . . . . .	83
6.4	Conclusion . . . . .	85
<b>7</b>	<b>XML Accounting Trail Model</b>	<b>87</b>
7.1	Introduction . . . . .	87
7.2	Developing the XML Accounting Trail Model . . . . .	88
7.2.1	Obtaining the Evidence . . . . .	89
7.2.2	Collecting the Evidence . . . . .	93
7.2.3	Storing and Protecting the Collected Evidence . . . . .	98
7.3	Assembling the Pieces — An Overview of the XML Accounting Trail Model . . . . .	99
7.4	Conclusion . . . . .	102

<b>8</b>	<b>Conclusion</b>	<b>105</b>
8.1	Problem . . . . .	105
8.2	Method and Results . . . . .	106
8.3	Self Evaluation . . . . .	108
8.4	Open Problems . . . . .	109



# List of Figures

3.1	Search query results illustrating lack of semantic awareness. . . . .	33
5.1	An overview of the process of detecting data irregularities using a compiler. . .	62
5.2	The classification levels of the data irregularities that can be detected using a compiler. . . . .	68
7.1	An algorithm showing how to obtain and use a trusted, secure timestamp. . . .	96
7.2	The basic components that combine to form the XML Accounting Trail model.	100
7.3	The operation of the XML Accounting Trail Model. . . . .	101
7.4	The role of the reference monitor in the XML Accounting Trail Model. . . . .	102





# List of Tables

2.1	Steps for implementing forensic readiness. . . . .	17
5.1	A tabular illustration of the juxtaposition of XML tags and compiler error types.	65
6.1	Core themes for investigative questions applied to the evidence gathered using a compiler to detect data irregularities. . . . .	82
6.2	An illustrative set of decision problems (or hypotheses), ordered by core theme, drawn from the evidence gathered using a compiler to detect data irregularities.	82



# Acknowledgements

I would like to thank several people who made invaluable contributions and assisted with this dissertation in countless ways.

First and foremost, I would like to thank my study leader, Prof. Martin Olivier, for the many hours of guidance and seemingly inexhaustible patience with my constant questions. It is much appreciated.

Also, a heartfelt thank you to my friend, Dr. Wynand Van Staden, for the valuable feedback after suffering through the torturous earlier drafts of this dissertation. Thank you for the valuable insight into the academic process and for the many hours of selfless sacrifice. Your detailed and considerate feedback elevated this dissertation to a new level of excellence and I would not have been able to complete this project without your input.

Furthermore, I would like to express my gratitude to Dr. Stefan Gruner, for the poignant conversation that stimulated and contributed to the ideas in chapter 5.

I would also like to thank my employer, PwC. Specifically, the following individuals supported me greatly in my studies and graciously allowed me the use of study time when needed: Sidriaan De Villiers, Michael Nean, Marcel Pillay, Barry Vorster and Herman Zulch. Your contribution is highly valued.

Finally, the most important thank you is for my family (especially my mom and dad) and my soon-to-be wife, Sheena. Thank you for your unwavering support in this endeavour and throughout my entire life. You are really special and I deeply appreciate who you are. Sheena, thanks especially for the countless meals, words of inspiration and your willingness to sacrifice our time together to allow me to finish this dissertation.

Mom, the last few months have been difficult with you being ill, and finally passing away on 09 September 2013. I really value who you are and who you have shaped me to be. As a small token of gratitude to the incredible role you played in my life, I dedicate this dissertation to you. Although this dedication you will not lessen the pain of your loss, this is my humble attempt at acknowledging the legacy you have left in my life.



# Abstract

In the 21st century, our worlds are becoming increasingly digitised and the business world is no exception. Financial transactions and accounting data are stored electronically, allowing for sharing and electronic processing. One of the most popular formats employed to store such data is the eXtensible Markup Language (XML).

Together with digitisation, crime involving electronic and digital means (cyber-crime) is rising sharply. Investigating acts of cyber-crime involve specialist skills, called digital forensics.

Accounting data stored in XML format is particularly vulnerable to unauthorised data modification (tampering) due XML's requirement to be human-readable. As a result, cyber-criminals can easily commit fraud or obtain financial gain by tampering with XML financial data in this manner. However, detecting such tampering is extremely difficult due to the so-called big data problem or needle-in-a-haystack problem. This involves searching for a particular item (in this case, the set of changes) in a large set of data (the entire XML accounting data file). To exacerbate the problem, it is not known whether tampering has occurred in any given XML accounting data file, causing one to possibly search for evidence of tampering which does not even exist.

Traditional approaches to isolate such tampering is not feasible. Firstly, due to the big data problem, using a sequential search of all data contained in the XML file to detect tampering is not efficient. Also, testing for data tampering using standard accounting rules is not possible, as modified data may still be valid in terms of accounting rules. Detecting data tampering in XML data therefore calls for a novel approach, forming the foundation of this work.

This study aims to enable an investigator to determine whether tampering occurred in a specific set of XML financial data as well as reconstructing the events leading to the tampering in order to determine the extent and detail of such tampering. In order enable the detection of potential tampering with data, this study proposes the creation of an automated tool to detect such irregularities in XML financial data. Using the parallel of forensic pathology, it is argued that XML financial data needs to be analysed for any artefacts (irregularities) that are not consistent with known normal (or "healthy") XML financial accounting data. This study furthermore argues that these artefacts are often noted in patterns, allowing one to attribute potential causality to groups of artefacts. It is also noted that compilers are typically used to parse input and detect any input not conforming to a pre-defined set of rules describing normal input. As a result it is therefore proposed that compilers and similar techniques should be used to protect financial data on the merit of their capabilities regarding parsing of data and error reporting and/or error correction.

Furthermore, the work performed as part of this study suggests a means to enable an investigator to reconstruct the events leading to the tampering in order to determine the extent and detail of the tampering. As data regarding the history and extent of changes is typically

not retained by either the operating system or the XML data format itself, it is proposed that instrumentation be employed to record the additional data necessary in order to reconstruct events.

This is achieved by using a combination of version control and audit logging to ensure that data is available to reconstruct tampering events. The XML Accounting Trail Model is therefore proposed to collect data about all modifications affecting the file containing XML accounting data. The model also proposes the combination of digital signatures and a reference monitor to ensure that all changes to the XML data files are recorded and that the XML Accounting Trail Model cannot be circumvented by direct editing of the XML file.

**Keywords:** Compiler, Cyber-crime, Data Irregularity, Digital Forensics, Forensic Readiness, XBRL, XML, XML Accounting. **Supervisor:** Professor Martin S. Olivier **Department:** Department of Computer Science **Degree:** Magister Scientia

# Chapter 1

## Introduction

### 1.1 Electronic Financial Data

In the global economy of the 21st century, the way in which we work and interact is progressively digital. Many activities have been affected by this digital revolution, and it should not be surprising that one of the primary beneficiaries of the digital age is the marketplace.

Internet and Communication Technology (ICT) enables businesses to be more effective whilst at the same time increasing productivity, client interaction and revenue. As a result, administration and overheads involved in managing finances also increased; and the need for managing the masses of financial information available to the modern business is commonly recognised [22, 48].

An increasingly important requirement in doing business is that of a better understanding and analyses of digitally stored financial data, transactions, and reporting on financial transactions [48, 7].

This growing need whereby financial data should be easily accessible to all stakeholders is recognised by Deshmukh [48]. He further observes that, in order for business to function effectively, a complete view representing all business data is required. This view includes financial data from all systems and applications responsible for processing financial data.

The deployment of techniques enabling financial data sharing between various systems and stakeholders is required to address the above need. These techniques are so in demand, that an entirely new field of practice, termed Business Intelligence (BI), was formed to address the need for a representative data view across all business systems [52, 95, 7].

The aforementioned techniques are heavily dependent on tools allowing the specification of data in a recognisable format [95]. In turn, this fuelled rapid growth in the use of eXtensible Markup Language (XML) in financial applications. Today XML is used in more than a hundred industry groups worldwide [37], underpinning various financial data applications. These typically range from the storage and retrieval of financial accounting data to the reporting of financial statements and business reports (through eXtensible Business Reporting Language or XBRL) [22, 29, 107].

XML's popularity is largely due to its design and its ability to include semantic information relating to data. XML's design [26] mandates, amongst others, that any resulting XML file be human-readable and be marked up in accordance with an agreed data definition.

Because XML is human readable, files containing XML data can easily be subverted. Responsible financial governance demands that steps be taken to ensure that such actions should

be detectable, and once detected, appropriate action should be taken to correct any misstatement resulting from such actions. The following section argues the case for digital forensics and forensic readiness which enables business to detect and investigate potential instances of unauthorised modification of financial data.

## 1.2 The Case for Digital Forensics and Forensic Readiness

The digital revolution spurred the growth of digital data and ICT. Correspondingly, the occurrence of cyber-crime grew at an alarming rate. According to the 2013 Internet Crime Report [109], released annually by the Internet Crime Complaint Centre, losses due to cyber-crime expanded rapidly in the last decade. The number of reported incidents increased 44-fold since 2001, when the report was released initially. Cyber-crime, in some form or another, was responsible for a reported net loss of \$781.8 million in 2013 alone. It is safe to assume that this is merely the tip of the iceberg as many incidents of cyber-crime are simply never reported due to reputational and other concerns.

Of the reported cases, incidents of fraud have the greatest impact [109]. The term *fraud*, derived from the Latin ‘*fraus*’ implying deceit or injury [117], typically falls into one or more of the following categories [9, 73]: corruption fraud, asset misappropriation, or fraudulent statements. Whilst corruption fraud refers to improper advantage having been obtained due to a person’s position; asset misappropriation is very similar to theft as possession of assets is transferred illegally. Fraudulent statements are the result of intentional deceitful tampering or misrepresentation of financial information.

According to the Association of Certified Fraud Examiners (ACFE), although fraudulent statements only account for around 5% of all fraud cases reported, this fraud type was responsible for the highest median losses (more than \$4 million) [9]. As such, this is the type of fraud that will be targeted in the work performed in this document.

The recent growth in cyber-crime has heightened the need for techniques to investigate and combat cyber-crime. As a result, the past two decades have seen rapid development in the area of *Digital Forensics*.

McKemmish [99] defines digital forensics as “the application of computer science and investigative procedures for a legal purpose involving the analysis of digital evidence after proper search authority, chain of custody, validation with mathematics, use of validated tools, repeatability, reporting, and possible expert presentation”.

Rowlingson [136] notes that the investigation time of digital forensic incidents is very expensive and the process of investigation is majorly disruptive to business. This is due to the requirement of leaving the crime scene intact so that evidence can be gathered. As a result normal business cannot resume, resulting in complete loss of revenue for the period from the incident to the conclusion of the investigation.

Forensic readiness, “the ability of an organisation to maximise its potential to use digital evidence whilst minimising the costs of an investigation” [136], has therefore emerged as one of the primary techniques to counter the high cost of investigation. Many regulatory bodies and good governance standards now require at least some form of forensic readiness.

Now that the topics of XML financial information and digital forensics have been introduced, the next step is establishing the scope and purpose of the work presented in this document.



## 1.3 Scope and Purpose

In the previous sections, it has been demonstrated that cyber-crime is growing rapidly and that occurrences of fraud relating to financial statements typically result in high average losses. Furthermore, it has also been demonstrated that financial information is increasingly being digitised, often with the use XML-related technologies.

However, the fields of XML and digital forensics are very broad, and it is necessary to narrow the focus to the specific information that will be relevant to this discussion. In doing so, the scope and purpose of the work will be established. The remainder of this section is dedicated to doing this.

As will be shown in the problem statement below, the use of XML for financial statements introduces several weaknesses that increase the opportunity for unauthorised modification of financial data. Often such unauthorised modifications are the direct result of fraud relating to financial statements.

A number of weaknesses in the use of XML were identified during a review of literature relating to XML. Due to XML's design requirement mandating that data be stored in a human-readable format [70], XML is particularly vulnerable to unauthorised modification via direct file access. Direct file access is defined as simply opening the file in a normal text editor, editing it and saving it again.

When considering the importance of financial and transactional data stored in XML, it is clear that the use of XML for financial applications greatly increases the risk of tampering with financial information with the purpose of deceit. Therefore, this dissertation also focuses on the prevention of unauthorised financial statement tampering (which is typically the result of fraud) by leveraging practices from forensic readiness.

Now that the scope of this dissertation has been established, the next step is that of establishing the problem statement (or purpose) which should be addressed by the work presented in this document.

### 1.3.1 Forensics in the Context of XML Financial Information

In the introduction to this section, it was noted that the scope and purpose of this work should be delineated. The aim of this section therefore is to refine the scope of the work by means of examining forensics in the context of XML financial information.

One of the key observations from the introduction is that XML financial information is subject to tampering. Controls around financial processes are not typically configured to detect tampering in a trivial way. In fact, research conducted by the ACFE [9] found that external and internal audits combined only detect around 18% of all fraud. Other commonly used detection mechanisms, such as combining reconciliation and IT controls, only detect a further 5%. In fact, well over 60% of all fraud is discovered by accident or specific tip-off.

One therefore concludes that the detection of misstatement and or unauthorised modification of financial data (tampering) is difficult and that conventional measures used to verify financial information are not very effective in detecting tampering.

Due to XML financial data's inherent weakness of being subject to tampering and its increasingly widespread adoption, a scenario where an investigator, business owner, or financial officer is presented with XML financial data and asked to determine whether the data can be relied upon (i.e. has not been tampered with) will become increasingly common.

Furthermore, in the event that the data is proven to be tampered with, the business stakeholders will be very interested in the detail of how the tampering occurred, what exactly was tampered with and how tampering can be prevented going forward.

### 1.3.2 Problem Statement

The problem statement can therefore be formulated as follows: *XML financial data is susceptible to tampering due to the human-readability property required by the XML data specification. Upon presentation of a set of XML financial data, how can one determine whether data has been tampered with, and reconstruct the past events so that the nature of tampering can be determined?*

The purpose of this dissertation is to introduce a means of detection of unauthorised modification of XML accounting data. Furthermore, this dissertation aims to enable the reconstruction of the events involving the tampering so that the extent of the tampering can be determined and corrected.

Note that full reconstruction may however not be possible as some of the reconstruction steps may fall outside of the digital forensics domain. One example is that of attributing tampering actions to a specific person — a user account password may have been shared, and proving who accessed the account at a specific point in time will not be possible using digital forensic techniques in isolation.

Having discussed the scope and purpose of the work, the question of how the problem will be addressed remains. In the next subsection, the methodology involved in addressing the problem (as formulated above) is discussed.

### 1.3.3 Research Method

In order to address the problem of detecting unauthorised modification of XML accounting data, it will be shown that tampering with XML data is typically characterised by a pattern of modifications. As a result, a method using automated means to detect modification patterns and deviations in the expected data pattern (a pattern of expected data which is present in legitimately generated XML financial data) will be proposed. Such deviations will be termed *data irregularities* and include, amongst others, changes resulting from unauthorised data modification. This method intends to significantly reduce manual interaction in the process of detecting potential unauthorised modifications to the XML accounting data, and also to increase the efficiency of such a detection process.

In addition, it will be shown that reconstruction of the events that led to the tampered XML financial data requires the creation of hypotheses surrounding the events leading to the data irregularities. These resulting hypotheses also require testing. Therefore, the scientific requirements for digital forensics are introduced and the process of creating hypotheses is demonstrated.

Subsequently, it is demonstrated that in order to test the hypotheses, data regarding the events that led to the modification is required. Similar to the introduction of a black box in aircraft, used for accident reconstruction, some form of instrumentation is required in order to gather and recall event data. In turn, the gathered data is necessary in order to test the hypotheses.

In order to collect data about the events, a means of instrumenting is required. A model, called the XML Accounting Trail Model, is therefore proposed. This model intends to collect data around events where XML financial data is modified and aims to reduce and/or eliminate unauthorised modification of XML accounting data.

The XML Accounting Trail Model's requirements and components will be derived by investigating various evidentiary requirements and by reviewing the methods with which XML data can be modified in an unauthorised manner. Following the derivation process, the model's components will be assembled and an overview of the model and its practical functioning will be discussed.

Having discussed the methodology that will be used to address the problem statement, the next important discussion point is the contribution made by the work. In the next section, the contribution offered by the work performed in this dissertation towards solving the problem statement is discussed.

### **1.3.4 Contribution**

The main contribution of this body of work is that of proposing the use of a compiler (parser) to parse XML financial data according to a pattern (grammar) and proposing that the reported errors be used to provide forensic information regarding whether or not the data was tampered with.

Using a compiler to search for patterns of data tampering employs an approach similar to that used during a forensic autopsy [89]. The standard procedure in performing a forensic autopsy (also termed a medico-legal autopsy) requires a detailed examination of the body. Instead of simply considering the reported or apparent cause of death, the pathologist is required to perform an extensive range of tests and observations. The pathologist achieves this by comparing normal autopsy appearances and artefacts with the appearances and artefacts observed in the body being examined. By applying this extensive method, the complete body of evidence and all potential causes of death are considered, enabling the pathologist to make an informed decision regarding the actual cause of death, rather than simply considering the apparent cause of death.

In the same way as a forensic autopsy, employing a compiler to analyse financial data ignores apparent instances of data tampering and instead executes a detailed check-list of tests to determine potential inconsistencies and/or traces of data tampering. This is a novel approach to investigating unauthorised modification of data, which serves to enable the investigator to determine all potential sources of tampering in a set of data.

Deploying the proposed methodology will enable investigators to significantly increase their confidence regarding whether data tampering occurred. It assists the investigator to detect whether the data file reviewed by the compiler can be considered normal (and therefore excluded from the investigation) or whether it requires further investigation, thereby reducing investigation time as the investigator is better able to triage the evidence on the scene. Furthermore, this solution will increase the detection rate of data irregularities and unauthorised modifications of XML accounting data.

A second contribution made by this work is proposing the use of version control and audit logging as part of the instrumentation required to determine the root cause of data tampering. The XML Accounting Trail Model proposes the collection of data about all modifications affecting the file containing XML accounting data. This data forms a body of evidence which

can be used in a forensic investigation to test hypotheses and determine the root cause of data irregularities. Additionally, the model proposes the combination of digital signatures and a reference monitor to protect XML files from unauthorised modification. This results in the XML Accounting Trail Model not only securing XML data from further tampering but also positioning itself so that it is able to record all data about all changes made to the financial data.

Although related work in the field of XML forensics was noted in [59], the XML Accounting Trail Model's ability to extract forensic evidence relating to unauthorised modification from XML financial data is a vital contribution to forensic science.

Now that the scope and purpose of the work is defined, the next step is that of outlining the approach taken in this dissertation.

## 1.4 Approach

This dissertation opens with a set of background chapters, aimed at introducing the various concepts that will be utilised in achieving the goals set out in the problem statement. Chapter 2 introduces the concept of digital forensics, and explains the digital forensic process. This chapter also discusses the practice of digital forensics, establishing the concepts of forensic readiness, evidence and requirements surrounding evidence, and finally, evidence certainty.

Chapter 3, in turn, discusses XML and provides background on the use of XML, the reasons for its use and how it is used in financial applications. The chapter concludes with a very important discussion on the weaknesses and concerns around XML.

An overview of compiler theory is presented in Chapter 4. Chapter 4 discusses the analysis of input and the phases of such analysis. Furthermore, the chapter notes the various types of errors that may occur during the analysis of input and subsequently concludes with the key notion of handling such errors. This forms a foundational part of the discussion in the subsequent chapter, which introduces the use of compiler theory to detect data irregularities.

Having introduced the ory behind compilers, Chapter 5 focuses on the proposal of using automated techniques in the determining whether tampering (unauthorised modifications) took place in XML financial data. This chapter marks a departure from providing background, and instead introduces the own work around which this dissertation is built.

This chapter opens with an analysis of XML accounting files, and then discusses means for detecting tampering by using data irregularities — irregularities caused by unauthorised modification of XML accounting data. The key purpose of this chapter is the proposal of using a compiler to analyse XML accounting files and to detect data irregularities.

The next chapter, Chapter 6, focuses on the reconstruction of events that led to the detection of unauthorised modification to the financial data. The reconstruction is introduced by arguing that digital forensics should become more scientific in its definition, in order to better satisfy the requirements of admissibility in a court of law. Scientific methods and the evidentiary requirements for admission in a court of law are then discussed and a key point of the discussion is the investigation of applying scientific principles. Subsequently, a set of hypotheses are derived.

Chapter 7 notes the need for instrumentation to capture the data needed to test the hypotheses derived in Chapter 6. The primary purpose of this chapter is to propose the XML Accounting Trail Model, aimed at protecting XML accounting data from unauthorised modification. The chapter also discusses the various approaches that could be taken to accomplish

this goal, and concludes with the proposal of the actual model and its application.  
Finally, this dissertation is concluded in Chapter 8.



# Chapter 2

## Digital Forensics

### 2.1 Introduction

In this chapter, a broad background on digital forensics will be provided to the reader. The purpose of this chapter is to assist the reader in contextualising the later work in Chapters 5, 6 and 7 within the parameters of forensic readiness, and to a lesser extent, the broader phases of a forensic investigation.

Initially, the focus of this chapter will be on providing a working definition of digital forensics. From this basis, the focus of the discussion will shift to that of examining the forensic process. This discussion is comprised of an overview of the phases of a digital forensic investigation, followed by a discussion of the economics of such an investigation. Leveraging the argument regarding the cost-intensiveness and disruptive effect of an investigation, this section concludes with a review of forensic readiness. The section on forensic readiness is of great importance for the work done in later chapters and will set the scene for many of the key arguments in the chapters to come.

The chapter on digital forensics is concluded with a section discussing the broader practice of digital forensics. As part of this discussion, regulatory requirements are reviewed, which in turn forms a key part of the conversation in Chapters 6 and 7. Also, evidence and evidence certainty are discussed in great detail.

Over the past few decades, interaction between humans has become increasingly digitised. The digital revolution made computers indispensable for various aspects of our daily lives: work, communication, commerce and even our entertainment. Touted as the new world of information, the digital revolution delivered many conveniences (many unthinkable, as little as forty years ago) now taken for granted, for example: instant communication, being a true global village and being able to easily trade and do business despite geographically disparate locations.

As with many things in life, this change was also a double-edged sword. Crimes traditionally limited to non-digital means such as extortion, fraud, money laundering and scamming were being translated to the digital world. According to the 2013 Internet Crime Report [109] (released annually by the Internet Crime Complaint Centre), 119,467 complaints resulting in loss were received. The total loss due to fraud amounted to US \$781.8 million. Losses have tripled since 2008 (a loss of US \$264.6 million), and grown by a factor of 44 (4392%) since 2001 (where losses of US \$17.8 million were recorded). These statistics clearly show that cyber-crime is a rapidly growing industry.

Consequently, the investigation of digital crimes plays an increasingly important role. Both the application and techniques of digital forensics, and the investigation of digital crimes, have grown tremendously within the last few decades. The next section provides an overview and definition of this field.

## 2.2 Definition of Digital Forensics

Prior to any discussion of digital forensics, it is important to first establish a working definition of the term.

Although definitions for digital forensics abound, and the topic has been at the centre of many debates, a comprehensive review of the definition of digital forensics falls outside of the scope of the work done in this chapter. The interested reader is referred to Casey [35], Cohen [41] and Wolfe [171], to name but a few of the authoritative sources on the subject.

The purpose of this section, instead, is simply to provide a concise, working definition of the term digital forensics. For such a definition, I prefer the clarity and conciseness of the definition provided by McKemmish [99] (on behalf of the Australian Institute of Criminology). He notes that digital forensics is “the application of computer science and investigative procedures for a legal purpose involving the analysis of digital evidence after proper search authority, chain of custody, validation with mathematics, use of validated tools, repeatability, reporting, and possible expert presentation”. This easily understandable definition provides a very good introduction to the field.

That said, no discussion on the definition of digital forensics can be complete without the formal definition of the field provided by Palmer [118]. Although less concise and simple than the definition by McKemmish, Palmer’s definition is widely regarded as the landmark definition of the field, and is one of the most comprehensive definitions available.

At the 2001 Digital Forensics Research Workshop, Palmer defined digital forensics as “the use of scientifically derived and proven methods towards the preservation, collection, validation, identification, analysis, interpretation and presentation of digital evidence, derived from digital sources for the purpose of facilitating or furthering the reconstruction of events found to be criminal or helping to anticipate the unauthorised actions shown to be disruptive to planned operations”.

Due to its rigour, the Palmer definition will be used as a basis definition throughout the remainder of this chapter. Now that a formal definition of the field of digital forensics has been provided, the discussion can progress to more advanced topics within the field, with a discussion on the digital forensic process.

## 2.3 The Digital Forensic Process

As mentioned in the introduction, the digital forensic process and the approach advocated by such a process is of vital importance in later work discussing the collection and retaining of evidence by the XML Accounting Trail model (Chapter 7), as well as the detection of data irregularities in XML accounting data (Chapter 5).

The process of a digital forensics investigation consists of various phases, as alluded to in Palmer’s definition of digital forensics [119]. The next section investigates these phases in more detail.



### 2.3.1 Phases of a Forensic Investigation

In the previous section, it was stated that the first important area to discuss is the digital forensic process.

This section introduces the reader to the forensic process by presenting various views on the phases of an investigation, as suggested by several models describing the process of performing a digital forensic investigation.

Reith et al [133] stated in 2002 that digital forensics is a “relatively new science” and that a “standard or consistent digital forensic methodology” does not yet exist. This situation has however changed quite significantly in recent years. Whilst a standard “one-size-fits-all” methodology remains hard to define due to the complexity of the field and the wide array of scenarios that an investigator might be confronted with, there is a significant amount of material available regarding forensic models or frameworks.

Olivier [114] cites two requirements for a general model: 1) it “captures the essential aspects of a system or process, while it ignores the non-essential aspects”; and 2) a model has to be “expressed clearly and concisely”.

At this point it should be noted that the purpose of this section is not to investigate or critique all of the models available. Instead, it aims to provide an introduction to some of the more common models and to distil the common phases or elements from these models into a more generic set of phases involved in a digital forensic investigation. As mentioned in the introduction of this chapter, these phases become the context within which the XML Accounting Trail model will be derived. Given that the purpose of this section is not a comprehensive overview of all models, such a detailed discussion falls outside of the scope of this chapter. The interested reader is referred to the work of Kohn et al [90], Baryamureeba and Tushabe [13], and Reith et al [133].

Given the research noted above, the author deems the following five models to be representative of the majority of opinions on the phases of a digital forensic investigation. As such, discussion will be limited to the five models below:

- the McKemmish model [99];
- the United States Department of Justice model [108];
- the Kruse and Heiser model [92];
- Lee et al’s Scientific Crime Scene Investigation Model [94]; and
- the Integrated Digital Investigation Model by Carrier and Spafford [31].

McKemmish [99] notes that forensic computing encompasses four key elements, namely 1) the identification of digital evidence; 2) the preservation of digital evidence; 3) the analysis of digital evidence; and 4) the presentation of digital evidence. Although rather basic, this is a good foundation for the rest of the discussion.

The United States Department of Justice (DoJ) [108] proposal for a model uses language very similar to that of McKemmish. As with McKemmish, their model too has four steps: 1) collection; 2) examination; 3) analysis; and 4) reporting.

The ‘collection’ phase of the DoJ model correlates with McKemmish’s ‘identification’ and ‘preservation of digital evidence’, as it includes “evidence search, evidence recognition, evidence collection and documentation” [108]. Also, the DoJ ‘examination’ and ‘analysis’ phases

are related to the ‘analysis of digital evidence’ step in McKemmish; and the DoJ ‘reporting’ phase falls under McKemmish’s ‘presentation of digital evidence’ step.

It can thus be concluded that McKemmish’s model and the DoJ model are equivalent in content.

The third model to be contrasted is that of Kruse and Heiser [92]. It has three basic components: acquiring the evidence, authenticating the evidence and analysing the data.

‘Acquiring the evidence’ correlates with the ‘collection phase’ of the DoJ model, as does ‘analysing the data’ and ‘analysis’. As is rightly pointed out by Kohn et al [90], the reporting component is absent from Kruse and Heiser’s work, resulting in a model less comprehensive than the previous two models. When considering its inferiority and the fact that the Kruse and Heiser model did not introduce any new content, this model is deemed inferior in relation to prior models, and will therefore be excluded from further consideration.

The fourth model to be contrasted and compared to prior models is Lee’s Scientific Crime Scene Investigation Model (SCSIM) [94]. Like the DoJ model, it also consists of four steps: recognition; identification; individualisation and reconstruction. As Kohn et al [90] point out, these steps are all ‘investigative’ in nature. Consequently the SCSIM is also considered inferior to the DoJ and/or the McKemmish model and it too will be excluded from further consideration.

Baryamureeba and Tushabe [13] point out that the DoJ model too is flawed due to its failure to adequately define the analysis phase. Baryamureeba and Tushabe comment that the DoJ model “confuses analysis with interpretation despite these being two distinct processes” [13]. As such, the DoJ Model too should be excluded as a comprehensive overview of the phases of which a digital forensic investigation should be comprised.

The final model to be discussed is the Integrated Digital Investigation Process model (IDIP) by Carrier and Spafford [31]. This model is exquisitely detailed and consists of four phases which in turn consists of a total of seventeen sub-phases.

As a full discussion of each of the phases falls outside of the scope of this chapter as this chapter only intends to provide an overview of the phases of a digital forensic investigation. The reader interested in a detailed discussion of the philosophy of digital forensic investigations is referred to Carrier [31] or Baryamureeba and Tushabe [13] for a full discussion of the IDIP model. Instead, a brief overview of the four phases of the IDIP model will be presented.

The IDIP model lists the following phases: 1) Readiness Phase; 2) Deployment Phase; 3) Crime Scene Investigation Phase (both digital and physical); and finally 4) the Review Phase. It is clear that this model incorporates all four stages of the DoJ model (into Phases 3 and 4) whilst adding additional value in the form of Phases 1 and 2. Also, this model does not suffer from the main criticism leveraged at the DoJ model by Baryamureeba and Tushabe [13].

Furthermore, none of the prior models introduce the idea of preparation for a forensic investigation. This notion of ‘readiness’ is key to the work presented in this dissertation and will be discussed in greater detail in Section 2.3.3. As a result, the prior models are not sufficient, and the IDIP model is viewed as the most complete of all the models. This model will therefore be used as a basis for discussing investigations throughout the remainder of this dissertation.

For the purposes of the rest of this dissertation, the reader should note that Phases 1 and 3 will be of prime importance, given the later work in Chapters 5, 6 and 7. The discussion now turns to the economics of a forensic investigation.

### 2.3.2 Economics of a Forensic Investigation

Now that we have discussed the various models and phases of a digital forensic investigation, we return our attention to the economics of such an investigation.

Each of the above-mentioned phases contributes to the cost of a forensic investigation. It will be argued that a forensic investigation is a very expensive undertaking and is also very disruptive to business. These arguments will in turn then become the foundation to a second argument advocating the need for businesses to minimise the cost and duration of a forensic investigation. One of the most common ways of achieving cost containment on a forensic investigation is to prepare for an investigation in advance and retain evidence as a matter of standard operational procedure. This is called ‘Forensic Readiness’ and will be the focus point of the next section.

The cost of a forensic investigation will now be analysed based on five primary economic factors involved in computing the cost of an investigation as per Stephenson [150]. These are: Investigation Complexity, Cost of Labour, Cost of Examination Tools, Availability of Forensic Skill and Loss of Revenue due to Investigation.

The primary driver of the cost of a forensic investigation is the ever-increasing complexity of investigations. Stephenson [150] notes that forensic investigations are continually becoming more complex, and hence more expensive to perform, due to the following reasons: remote crimes, obfuscation of evidence, the amount of data to analyse, and false positives.

Each of these reasons will now be briefly discussed:

**Remote Crimes** - Due to the fact that computers are inter-linked in networks, a cyber-criminal is not constrained by geographical boundaries. In turn, this means that a digital forensic investigator does not have a traditional jurisdiction anymore, but can effectively be called on to investigate crimes from all over the globe.

**Obfuscation of evidence** - Cyber-criminals easily cover their tracks, and acts of destroying evidence or even planting false evidence are often seen. A forensic investigator therefore needs to be exceedingly critical of evidence and needs to carefully ascertain its validity, escalating investigation time and cost.

**Amount of data to analyse** - Moore’s Law states that “the number of transistors that can be placed inexpensively on an integrated circuit doubles approximately every two years” [85]. This trend is however not limited to processing power, and applies equally to storage capacity. Kryder observed a similar trend (Kryder’s Law) stating that the “magnetic disk ... storage density doubles annually” [164]. Also, the scope of appliances in digital forensic investigations has increased tremendously: not only computers but tablet PCs, smart phones, digital cameras and electronic surveillance systems have become de facto sources of evidence in digital forensic investigations. In the real brick-and-mortar world, this can be compared to an ever growing crime scene with no clearly distinguished borders.

**False positives** - False positives can never be completely eliminated, and often a forensic investigator is dispatched, only to find (after significant effort) that there was no discernible incident and that it was a false alarm.

A second economic factor driving up the cost of a digital forensic investigation is that of investigation time. Tan [154] observes that the investigation time typically far exceeds the time

taken to perpetrate the crime. Empirical evidence from the HoneyNet Project<sup>1</sup> shows that a 30 minute attack required an average of 48 hours of investigation [154].

Tan [154] further remarks that investigations rarely result in finding all relevant evidence. None of the teams participating in the HoneyNet Project managed to collect all relevant evidence.

Using the data from the HoneyNet Project, a 30 minute attack means at least 48 billable hours of investigation, excluding any time spent on recovery, hardening of the compromised system or even the conclusion of the entire forensic investigation. Based purely on man-hours and salary cost, forensic investigations are very expensive indeed.

The third economic factor driving investigation cost is that of examination tool expenses. Moore [105] notes that the number of different technologies in use by consumers today have a very negative cost effect, as each of these technologies (such as different mobile phone manufacturers and models) require tailored tools. Worse, such technologies often use proprietary formats to store data, resulting in further cost in order to purchase specialised investigation tools.

In order to provide the reader with an understanding of the significant license cost involved in the use of forensic tools, one may consider the approximate cost of some well-known forensic packages used for general investigations of desktop computers and mobile devices [151, 121, 1].

At the time of writing, the *EnCase* package retails for between US \$3000 and US \$5000, depending on the additional features selected. This software provides various features for use in data acquisition, data analysis and reporting of amongst others, desktop forensics and mobile forensics. Another package, *Forensic ToolKit*, or better known as *FTK*, retails for approximately US \$6000 (license fee only). This package is typically used for the analysis and investigation of desktop computers as well as mobile devices such as tablets and cell phones. It also provides features for data acquisition, data analysis and reporting.

Yet another package, *X-Ways Forensics*, retails for around US \$ 1600. This package also provides data acquisition (such as imaging), data analysis (such as disk analysis, file classification, etc.) and reporting functionality for both mobile and desktop forensics.

The above packages however only provide the software tools to assist with a digital forensic investigation. Typically, hardware tools such as write blockers, evidence bags, disk duplicators, mobile phone kits, SIM cloning kits, etc. are also required. These tools also easily run into hundreds of US dollars per item. For example, the hardware provided by Paraben [121], a well-known provider of hardware for mobile phone investigations start at around US \$28 for a SIM card reader (excluding software), and rise to about US \$4995 for the full Mobile Kit (a package providing a software analysis tool, SIM card reader, power kit, phone evidence bags, and more).

Another economic factor driving the cost of a digital forensic investigation is the lack of forensic skill [18, 130]. Factoring in the well-known economic principle whereby cost is directly dependent on supply versus demand, the cost of an investigation is affected very negatively indeed. Furthermore, forensic investigators are typically required to attend training and obtain certifications in using the above-mentioned tools such as *EnCase* and *FTK* [151, 1]. These courses are often structured so that multiple certification levels exist, with each of these certifications requiring course attendance and examinations. Such courses easily run into sev-

---

<sup>1</sup>A contest in which participants were supplied with disk images of a compromised honeynet system, and then expected to investigate it and compile a report on the findings of their forensic investigation.

eral thousand US dollars per attendee, and certification is a major contributor to the cost of a forensic investigation.

The final economic factor responsible for driving investigation cost is revenue loss. Rowlingson [136] notes that business continuity and/or incident recovery is directly opposite in terms of goals from investigation of the incident. Investigation demands that the crime scene be kept pristine, whereas business continuity demands that business be conducted as soon as possible after an incident.

As a result of normal business practice, evidence will be damaged, discarded or ignored and consequently business practice cannot continue during the process of an investigation. This drives not only cost of investigation but also results in the complete absence of revenue during the course of the investigation. As can be seen by the economic factors mentioned above, the cost of a forensic investigation can easily run into tens (if not hundreds) of thousands of US Dollars.

Although required by industry standards and legislation, many businesses can simply not afford a full-blown forensic investigation. In the next section, a popular way of reducing such investigation cost by gathering evidence ahead of time is discussed.

### 2.3.3 Forensic Readiness

In this section, the concept of forensic readiness is suggested as a means to minimise investigation cost and disruption to a company's business practice. This section concludes the larger discussion point on the digital forensic process and proposes the practice of pre-empting a forensic investigation by implementing various steps to ensure that evidence is retained even before an incident takes place. This practice is termed 'forensic readiness'.

Carrier [31] refers to 'readiness' as one of the phases of the IDIP model that he proposed. In fact, he specifically mentions *Infrastructure Readiness*, noting that it is important to "ensure that the needed data exists for a full investigation to occur". Carrier goes even further, stating that this phase is especially important for those who "maintain the environment that could be the target of a crime".

Electronic evidence is easily lost or damaged. Such evidence often requires expert knowledge and special procedures to properly preserve it, in order for it to later be presented as valid evidence in court [136]. It is therefore vital to ensure that the infrastructure is ready to preserve such evidence in addition to making the evidence as easy as possible to retrieve [31].

As demonstrated in the previous section, digital forensic investigations often becomes very expensive, and the cost increases steeply as the investigation time increases, due to cost of investigative time and loss of revenue [136]. Digital forensic investigation cost can however be kept under control if evidence is available right from the start of the investigation. Firstly, the investigation time will be significantly reduced as the investigators need not conduct an exhaustive search to locate potential evidence. Secondly, due to the decrease in investigation time, business down time (and loss of revenue) is kept to a minimum as well.

Ensuring that evidence is easily accessible and ensuring the constant availability of relevant and up-to-date forensic information is called forensic readiness. Rowlingson [136] alludes to the availability of evidence in his definition but notes that forensic readiness is more than just keeping evidence. Rather, forensic readiness is "the ability of an organisation to maximise its potential to use digital evidence whilst minimising the costs of an investigation".

From Rowlingson's definition [136] above, the two main objectives for forensic readiness

can be summarised as follows: *maximising the applicability and usefulness of the evidence generated by an incident whilst minimising the cost of the investigation.*

Forensic readiness essentially allows for a two-pronged strategy in assisting business. It enables compliance with the investigative overhead of regulatory and industry requirements, whilst lowering the (often prohibitive) cost of an investigation. Although forensic readiness is starting to be actively researched by the research community [155], the specification and implementation is not yet consistent or widely applied in industry [155, 130].

However, reduced investigation cost is not the only advantage posed by employing forensic readiness. Forensic readiness also offers additional substantial benefits to an organisation upon implementation, such as [136]:

- The demonstration of due diligence and good corporate governance by the business. This is beneficial when audits are performed and saves cost and time during an audit cycle.
- Meeting regulatory, legislative and industry requirements, saving cost in potential fines and poor publicity for lack of compliance.
- Multiple scenarios exist where the application of the gathered evidence could result in benefit to the company beyond that of a forensic investigation. Examples include deterring would-be criminals; aiding in legal action (both against and in favour of the business); providing evidence during disputes and assisting in interfacing with law enforcement (in the case of an event requiring the involvement of law enforcement). Having evidence on hand will save time and cost when dealing with internal human resource-related matters as well as lawsuits or other complaints against the company from third parties.
- Aiding in the efficient and rapid investigation of an incident, rapidly increasing the success factor of an investigation.
- Reducing the costs of any forced disclosure of data (regulator, legal or court-ordered disclosure) by ensuring that data is readily available.

Having considered the benefits of forensic readiness, the next step will most likely entail implementing forensic readiness. Below, several activity points for implementing a forensic readiness programme will be discussed, as suggested by Pangalos and Katos [120], Rowlingson [136] and Tan [154].

Note that the purpose of this discussion is to provide an introductory overview regarding the process of implementing forensic readiness and should not be treated as an exhaustive treatise on the subject. An overview of these steps, together with a short description can be found below in Table 2.1. However, a detailed discussion of the various stages and implications of implementing forensic readiness falls outside of the scope of this discussion and the interested reader is referred to Rowlingson [136].

Table 2.1: Steps for implementing forensic readiness.

<b>Implementation Step</b>	<b>Description</b>
Determine and define business scenarios requiring digital evidence.	Perform a quantitative survey to determine the risk and potential business impact posed by various crimes and/or disputes. Gathering business scenarios is foundational in determining the proceeding of an investigation and helps to eliminate excessive and unnecessary requirements.
For each scenario, determine potential sources and types of evidence.	Determine the capabilities of current systems in terms of producing potential evidence, and identify areas of shortcoming. Once shortcomings are identified, measures should be put in place to gather such evidence on a regular basis.
Determine the requirements for evidence collection.	Business needs to determine whether evidence collection only cater for internal investigation or for full-scale legal and justice requirements. Furthermore, an evidence requirement manifest (a document stating which evidence is needed from which systems) should be drawn up in order to facilitate an understanding of evidence requirements between the systems administration personnel and the parties requiring evidence.
Establish the capability to securely gather legally admissible evidence.	Evidence should be gathered in such a manner that is it is legal and admissible in a court of law. Policies should be put in place informing users that all actions are being monitored, and requiring consent towards such monitoring.
Draw up a policy governing the secure storage and handling of evidence.	The purpose of such a policy is to eliminate (as far as possible) tampering with and unauthorised access to evidence. Such occurrences could render evidence tainted, diminishing its worth in any judicial or legal proceedings. The objective should be to secure the evidence for a longer term and ensuring that it can be easily retrieved from safe-keeping.
Institute monitoring and intrusion detection.	Business should ensure that monitoring is set up in such a way that it will detect and deter major incidents. Furthermore, different types of monitoring should be used, from binary checking and network analysis to anti-virus programs. Refer to Stephenson [150] and Tan [154] for an in-depth discussion on such technologies.
Define circumstances requiring full, formal digital investigation.	Triggers, determining when a full investigation is required, should be applied to the raw data captured by the previous step. A definition of a suspicious event should be established and an action plan, indicating who should be contacted in the case of an event, should be set up. Such an emergency contact should be available on a 24 hour basis, 7 days a week.

*Continued on next page*

Table 2.1 – *Continued from previous page*

<b>Implementation Step</b>	<b>Description</b>
Train relevant staff members in the appropriate response to an incident.	Staff should be briefed on incident awareness, ensuring that all staff members involved understand their role in the investigative process. Staff should also be trained to not compromise evidence. Often well-meaning bystanders completely destroy the integrity of a crime scene, rendering potential evidence of little or no use in a court of law [31].
Thorough documentation of all incidents.	All incidents and their impact should be described and logged. Such information is often indispensable in making sense of the situation in the aftermath of an event. Also, this information is crucial in providing a permanent, post-event record of which actions were taken.
Regular legal review and oversight.	Legal review of all incidents deemed in need of a full, formal digital investigation is required in order to assist with the actions to be taken in response to the incident.

Ensuring that sound forensic readiness principles such as the above are in place, is often key in keeping investigation cost, effort and business disruption to a minimum. However, forensic readiness may be costly to implement, for example: efforts to train staff, establishing secure evidence storage or drafting and updating policies [136].

Considering the argument above, one concludes forensic readiness is beneficial as the advantages posed by being forensically ready in case of an incident considerably outweighs the inconveniences and upfront cost involved in the process. Forensic readiness is sure to be of vital importance in the years to come, especially if cyber-crime continues to grow at the astonishing rate mentioned in the introduction of this chapter. The practice of forensic readiness also forms a key part of the model proposed in Chapter 7.

In the next section, the practice of investigating a digital forensic incident will be discussed.

## 2.4 Digital Forensic Practice

The previous section focused on the digital forensic process, with emphasis on the phases of a digital forensic investigation, the economics of such an investigation and the concept of forensic readiness. In keeping with the aim of providing the reader with context around digital forensics for use in later chapters, this section will discuss the practical aspects of the digital forensic process.

Specific emphasis will be placed on current practices relating to investigation and regulatory requirements, as well as a review of evidence including the characteristics, integrity and certainty of evidence.

All business owners face risk as part of their standard business practice. This comes in many forms, such as the risk of loss of revenue, loss of ability to conduct business or risk of the breach of regulatory stipulations, to name a few [152].

One such primary risk is that of loss of the ability to conduct business. In the event that this risk is not addressed, the business may be forced to close down. This primary risk is often



largely constituted by systems risk [152]. Straub and Welke [152] define ‘systems risk’ as “the likelihood that an organisation’s information systems are insufficiently secluded and protected against certain kinds of damage or loss”.

Given the modern reliance on digital systems to perform business and the potential devastation that cyber-crime can wreak on such systems, business stakeholders are increasingly faced with severe systems risk.

As a result of this risk, regulatory requirements have become increasingly strict and no discussion on digital forensics can be complete without at least a cursory discussion on the regulatory requirements enforced with regards to incident handling.

## 2.4.1 Regulatory Requirements

As such, before discussing evidence and evidence certainty, it is important to briefly discuss the most prominent regulatory requirements that affect digital forensic investigations. Note that an exhaustive review of such requirements falls outside the scope of this discussion and will therefore be excluded.

The various regulatory bodies active in regulating and advancing responsible and proper business conduct are becoming more and more aware of the deficiencies in incident response. In addition, these bodies are beginning to require increasing levels of due diligence in all business operations [129]. A part of such due diligence is management’s awareness and appetite with regards to systems risk and the handling of any incident threatening business systems.

Due diligence, a term first introduced in the United States’ 1934 Security Exchange Act, is defined as “the preventative exercise aimed at the identification, disclosure and limitation of risks” [125].

Power [129] notes that a key step in ensuring due diligence is that of creating a statutory audit requirement applicable to corporate business enterprises. This is done by including a requirement for regular audit (often external in nature) of accounting and business practice in the legislation dealing with the registration of a corporate business enterprise [72, 53, 113, 68].

In addition to audit requirements, business sectors are increasingly governed by industry regulations requiring due diligence with regards to managing and containing systems risk. Examples of such legislation include the Health Insurance Portability and Accountability Act of 1996 (HIPAA) [66] for the healthcare sector in the United States, the Payment Card Industry Data Security Standard [123] in the finance sector (applicable to all enterprises engaging in card transactions on a worldwide basis), the Federal Information Security Management Act of 2002 [69] applicable to all federal agencies in the United States and the Financial Services Modernisation Act of 1999 [67] for the finance sector (again in the United States), amongst others. Similar laws are enacted by countries such as the United Kingdom and bodies like the European Union.

Both audit standards (such as COBIT and the ISACA guidelines) and industry regulations (such as those mentioned above) seek to increasingly moderate the way in which risk management is practised in the industry [120]. A number of industry guidelines (such as Section G28 of the ISACA Guidelines [78]) and standards now require forensic capabilities as well as a proper forensic investigation in case of an incident.

These guidelines typically stipulate requirements for a proper forensic process and policy. Furthermore, these guidelines are also very strict in specifying the circumstances, events and rules of engagement under which evidence may be collected in the case of an incident requiring

digital forensic investigation [120].

With these requirements in mind, it is very important to be clear on the definition of evidence, when evidence may be collected and how evidence should be verified (pronouncement of certainty of evidence).

## **2.4.2 Evidence: What Constitutes Evidence and How Should it be Handled?**

In the previous section, an overview of the regulatory requirements of evidence has been provided. However, no discussion of digital forensics can be complete without discussing the concept of evidence, its definition and how one should treat evidence in order to preserve its integrity.

The section will therefore focus initially on the definition of evidence, progressing to discuss legal requirements in terms of evidence.

As mentioned in the previous section, it is vital to be able to differentiate between an incident and a false alarm. Incidents are typically classified as being either a criminal or civil matter. The lines between these can however become quite blurred as a criminal matter may give rise to civil matter, and a civil matter may also give rise to a criminal matter [15].

A criminal case occurs when an individual is punished for the commission of an act that is deemed a crime by government legislation [15]. Depending on the legal system in use, such a case typically occurs between the state and the legal or natural entity who committed the crime. In contrast, a civil case usually involves a dispute regarding the duties and/or rights that individuals or organisations legally owe each other.

Most incidents requiring digital forensic investigation lead to a criminal case [150]. As such, the presumption of a criminal case will therefore be used throughout the remainder of this chapter.

### **Evidence: A Definition and Brief Explanation of the Need for Evidence**

In the event of a criminal case, Casey [35] notes that there are three key items that a judge or magistrate needs to be convinced of, in order for prosecution to be successful:

- that a crime was committed;
- that evidence exists proving the commission of crime; and
- that the crime was committed by the accused person.

Solon and Harper [146] as well as Casey [35] conclude that evidence is of critical importance in the prosecution process. Without evidence, there is no proof that a crime was committed or that the crime was committed by the accused.

However, prior to discussing the concept of evidence in detail, it is of great importance to first define the concept of evidence. Although many definitions exist, this dissertation will use the definition provided by the Oxford English Dictionary [117]. A detailed discussion of the various definitions of evidence and philosophy behind these definitions is out of scope for the work presented in this dissertation and the interested reader is referred to Cohen [41], Keane [84] and Solon and Harper [146].

The Oxford English Dictionary defines evidence as “something serving as proof; grounds for belief; testimony or facts tending to prove or disprove any conclusion”. One can therefore conclude that evidence in the legal sense is some form of proof (typically testimony or facts) that allows a conclusion to be proven or disproven.

In the next section, a brief overview of the legal process regarding evidence will be provided. It should be noted that many different legal systems exist around the world and that each of these have their own nuances and specific requirements. South African law is largely founded on British law, with some elements of Dutch law (and its strong historic influence of Roman law) as well as minor aspects of tribal law as practised prior to the Dutch colonisation of South Africa [160]. The purpose of this discussion is not to discuss the various legal nuances of each legal system as it relates to evidence, but rather to obtain a basis of legal requirements pertaining to evidence. As a result, this dissertation will use the law perspective of the British law (refer to Keane [84]). The reader interested in the specific nuances of other legal systems and their definition of evidence is referred to Kaptein et al [81] and Khan et al [86].

### **Law of Evidence and Burden of Proof**

Now that the term ‘evidence’ has been defined, the initial question becomes *what should the evidence constitute in order to be considered as valid evidence?* The first key consideration for any legal case is that of *what should be proven*. This is known as ‘the burden of proof’ and is governed by the ‘law of evidence’.

The area of law dealing with the types and quality of evidence as well as the presentation thereof is called the ‘Law of Evidence’ [84]. As part of the law of evidence, two items will be discussed: 1) the concept of ‘facts in issue’ and 2) the concept of ‘burden of proof’.

The first concept is that of ‘facts in issue’. This is defined as the facts that the prosecution must prove in order to obtain a conviction in the case. A subset of these facts, called ‘relevant facts’ or ‘evidentiary facts’ pertains to the facts surrounding the evidence submitted to the court by the legal parties.

The second concept of the law of evidence is that of burden of proof [84]. The burden of proof can again be subdivided in 1) the legal burden; and 2) the evidential burden. Of these two, the evidential burden is much more important to the issue at hand, namely the relevant legal requirements for the collection of evidence.

For completeness, the legal burden is briefly defined as the ‘probative burden’, the component of persuasion that the legal parties must perform in order to convince the court of the merit and validity of the case. As previously mentioned, this burden falls outside the scope of this discussion. The interested reader is referred to Keane [84].

The burden of proof relevant to this discussion, the evidential burden of proof, is defined as “obligation of a party to adduce (to offer as proof in discussion or analysis) sufficient evidence of a fact to justify a finding on the fact in favour of the party so obliged” [84]. In simpler terms, the evidentiary burden is a requirement for the introduction of sufficient evidence to convince the court of a specific fact and to justify a finding of this fact.

### **What Should be Considered as Evidence**

In the previous subsection, both the need for evidence and the purpose for evidence were explained. Tying this concept back to the topic at hand, that of the digital forensics practice, the question of what should be considered as evidence becomes very important.

As such, this subsection will discuss what should be considered as evidence in a digital forensic investigation.

In the field of digital forensics, evidence can be either physical or digital in nature [31]. Physical evidence might be in the form of fingerprints (to tie an individual to certain equipment); computer equipment (such as input devices, laptops, cell phones, etc.) or even access passes to prove that an individual had access to a certain location at a certain time.

Digital evidence, on the other hand, constitutes a record of actions in the digital realm, such as log files; data files containing information such as documents or spread sheets or even records of communication such as emails or chat logs.

In cases where guilt or innocence depends on the outcome of the trial, all evidence presented should be beyond questioning or doubt. This requires that proof be given that the evidence presented has not been tampered with and is authentic [35]. Determining the authenticity and integrity of digital evidence is quite difficult, as the domain of digital forensics is quite recently established and there is a significant lack of forensic skill in the police force and in the judiciary [18].

The topic of what exactly is considered as evidence is a very contentious one, and a more detailed discussion on this debate and the impacts thereof can be found in Section 6.2.1. A full discussion of this debate falls outside of the scope and intention of the discussion at hand.

For the purposes of this discussion, the content of a landmark decision on evidence admissibility [35], the case of *Daubert v. Merrell Dow Pharmaceuticals Incorporated* in the United States in 1993, is sufficient. These criteria are commonly known as the ‘Daubert Standard’, or ‘Daubert’ for short.

Daubert states that the theory or technique offered as evidence should be [35, 171]:

- empirically testable (i.e. it has been or can be tested and should be refutable and falsifiable);
- subject to review by peers as well as publication in a peer-reviewed environment;
- generally accepted by the relevant scientific community;
- governed by the existing standards and controls applicable to its operation (which should be regularly maintained); and
- subjected to and contrasted against the known or potential error rate.

Throughout the remainder of this section, the above criteria (and in particular the third item regarding being generally accepted) will form the basis for discussion. Note also the last point of Daubert, the mention of the known or potential error rate, which will be further explored in the final section of this chapter where evidence certainty will be discussed.

### **Digital integrity of evidence and the preservation thereof**

Up until now, the discussion revolved around the definition of evidence, the legal requirements imposed on evidence and what should be considered as evidence. However, no discussion of evidence can be complete without mention of how the evidence should be preserved and stored. In the previous subsection, Daubert was introduced to assist in answering the question of what should be considered as evidence.

Point 4 of Daubert is of particular relevance in this discussion as it specifically mentions the standards and controls applicable to the operation of the evidence gathering process. These standards and controls are key in preserving the digital integrity of evidence.

Most digital forensic professionals should concur that digital evidence is very fragile and could easily be altered, destroyed or manufactured by a user [35]. Given the requirements of point 4 of the Daubert Standard, the question posed by Hosmer [76] becomes vitally important: “How do we prove the integrity of digital evidence?” Without the correct preventative steps, it is clear that the credibility of digital evidence may be questioned and even destroyed in a court of law.

As mentioned before, digital evidence can originate from a wide array of sources, including PDAs, notebooks, computers, cellular phones, email messages, and chat-room logs [35, 41, 32]. In addition to considering the relevance or materiality of the evidence, the investigator should also consider the trust-worthiness and credibility of the evidence very carefully.

Many vendors today provide solutions to extract evidence from digital devices (EnCase, FTK and Paraben, to name a few). Although extracting the evidence in both a forensically and scientifically sound manner is of importance (see Chapter 6 for a detailed discussion on this point), the storage of the evidence post-extraction is an important concern as well. Evidence should be stored in such a manner that the digital integrity of the evidence remains intact [35].

Vanstone et al [161] define digital integrity as follows: “the property whereby digital data has not been altered in an unauthorised manner since the time it was created, transmitted or stored by an authorised source”. Digital integrity is established, to varying degrees, by using any one or combination of the following measures [76]: checksums, one-way hashes, digital signatures, and trusted timestamps. Each of these will now be briefly explained:

**Checksum** - An error-checking method for data which applies a mathematical (polynomial) function that concatenates the data, resulting in a small integer value. This value can be applied to digital integrity by running the checksum again at a future time and, should the later value match the original, it can be argued that at least a very basic level of integrity exists. Of course, this method is not as robust and precise as a hashing (see below), as an identical checksum does not guarantee that the data has not changed.

**One-way hash** - Hashing was specifically devised to protect data against unauthorised change and works by applying a one-way mathematical function, resulting in a fixed length large integer value [143]. Per definition, it is exceptionally unlikely (depending on the algorithm used, of course) to find another data set results in an identical hash value. As such, hash values are particularly suitable and convenient to prove data integrity.

**Digital Signature** - This is a value, affixed to a document or data set, computed using a combination of the content of the document and a secret known only to the sender (i.e. a private signature key) [65]. This allows for the binding of the identity of the signer together with a guarantee of data integrity, making digital signatures very convenient and useful for providing data integrity. Digital signatures, however, still lack a very important component, namely time. Even when using a digital signature, it is not possible to determine when the signing of the evidence occurred and no information about the period of the integrity of the signed digital evidence is available.

**Trusted timestamp** - This is essentially a combination of timestamps and digital signatures, whereby a timestamp is generated based on the identity of the server issuing the times-

tamp, the time (or offset from a specific time) that the timestamp is generated and the content of the document or data for which the timestamp is generated [169].

Now that various methods to preserve and prove digital integrity have been discussed, one is tempted to conclude that digital integrity alone is sufficient to prove to a court of law that the evidence at hand is sufficient and accurate. This is however not the case. Digital integrity is necessary but not sufficient to convince the court that the evidence at hand is sufficient and accurate. Instead, a second component, that of a chain of custody, is required as well to ensure that what is considered as evidence is acceptable to a court of law.

This is known as ensuring ‘evidence authenticity’, proving to a court that “a) the contents of the record have remained unchanged [equivalent to digital integrity]; b) that the information in the record does in fact originate from its purported source; and c) that extraneous information such as the apparent date of the record is accurate”, according to Casey [35].

This is accomplished by documenting and maintaining a chain of custody, known as the “continuity of possession of evidence”, again according to Casey [35].

Chain of custody ensures that there is a clear picture of how evidence was handled and also provides information as to the life-cycle of the information. Each person who handled the evidence may in theory be called upon to testify that the evidence produced in the courtroom is exactly the same as the evidence examined by them. Given such a precedent, it is recommended that the number of individuals handling the evidence be kept to a minimum [35].

Should a proper chain of evidence not be established, opposing counsel may argue that the evidence was handled improperly and request to bar the evidence from consideration by the court. The rationale is that the evidence may have been altered, contaminated or even manipulated in order to give a particular impression.

One of the challenges faced by an investigator is the fact that evidence authenticity is not a binary concept (i.e. the investigator can vouch for evidence authenticity or not). Instead, there are varying degrees of certainty on the part of the investigator regarding the authenticity of evidence.

Chain of evidence and digital integrity are vital to ensuring that evidence can be presented to a court of law with confidence, knowing that the evidence is not contaminated or tainted, and can be relied upon to satisfy the burden of proof.

### **2.4.3 Evidence certainty**

The current discussion revolves around the practice of digital forensics. In the previous section the notions of evidence, what is viewed as evidence and how evidence should be treated were introduced.

The Daubert Standard was introduced in relation to the manner in which evidence should be treated. One criterion of the Daubert Standard refers to considering a known, or potential error rate as a benchmark for the admissibility of evidence.

This section explores the idea of determining an error rate, which will be illustrated by evidence certainty. Evidence certainty is an approach whereby the error rate is considered as part of the evidence in order to establish how ‘certain’ one can be of the evidence [35]. This is a concept that is rapidly gaining recognition as being of vital importance in legal matters.

Casey [35] notes that computers can introduce errors and uncertainty in multiple ways, infinitely complicating the already difficult task of proving evidence authenticity.

In the due course of a digital investigation, errors and uncertainties are likely to arise regarding the evidence. Examples abound: the drifting of time on clocks; timestamp delays or even log files that are rotated and overwritten.

In any of these examples it would be difficult to state with certainty which parties are responsible for the events or where the events originated [34], due to fact that evidence suggesting such information may be skewed or incorrect. The magnitude of the already difficult task is increased significantly when multiple systems and devices become involved [35] (as typically is the case in today's networked environments). Finally, as if the challenges mentioned previously are not enough, Hosmer notes that [76] misdirection of investigators has become a trend as cyber-criminals are increasingly manufacturing false evidence.

Errors and uncertainty originating from computer records make it difficult to meaningfully assess the trustworthiness of digital evidence, or to ensure that evidence addresses the burden of proof. Yet, despite the potentially severe ramifications caused by basing case outcomes on such potentially faulty and/or incorrect information, industry seems loathe to incorporate the evaluation of evidence certainty into everyday practice [34].

Evidence certainty, however, becomes vitally important if the Daubert Standard of admissibility is applied to evidence. As mentioned earlier, the Daubert Standard requires that evidence includes a quantification of the technique's potential rate of error. This leads Casey [34] to conclude that forensic examiners have a "duty to estimate how closely the measured values represented in their data approximate reality" or to express the degree of certainty with which they can attest to the evidence authenticity.

Given the examples above and the conclusions that Casey draws [34], it is clear that all digital evidence, at the very least possesses, some form of uncertainty and that an estimation of such uncertainty is to be expected as part of expert testimony by a conscientious witness.

Furthermore, forensic examiners who neglect to account for error, uncertainty and loss during the course of their investigation may (as a result) reach incorrect conclusions [35]. In presenting such incorrect conclusions, an investigation may be compromised if investigators are challenged to justify their findings under cross-examination. One is led to conclude that failing to account for possible error, uncertainty and loss during investigations will lead to damage to the reputation of digital forensic science as a whole.

The concept of evidence certainty will now be discussed in somewhat more detail, specifically with focus on the types of uncertainty and how such uncertainty may be measured when working with evidence. Casey [34] classifies uncertainty as being of one two types, namely temporal uncertainty and uncertainty of origin.

Temporal uncertainty is associated with the time-line and sequence of events. One example is that of 'clock drift' (discussed in Willassen [169] and Casey [34]), responsible for discrepancies between 'computer time' and 'real-time'. Consider the situation where a very fine-grain time resolution is of essence, in the case where sniffer logs from several systems (recording hundreds of events in the same second) are used as evidence. In this case, temporal uncertainty is compounded due to the use of several systems (implying a network). Routers and other network devices are more likely to have incorrect clocks as the times on these devices are not frequently used [169], resulting in severe difficulty in creating a time-line for network activity involving several routers and/or switches.

On the other hand, uncertainty of origin is defined as uncertainty in establishing the origin of evidence and is often of prime importance in apprehending offenders. Consider the example of email in a networked environment [35]. An example of uncertainty of origin that is commonly

encountered is that of forged email [34]. The ‘From’ header from an email message can very easily be falsified and therefore has a high degree of uncertainty. Other headers such as the ‘Received’ header can also be falsified, but this is rarer, thereby making ‘Received’ headers more reliable than ‘From’ headers in terms of certainty of origin.

Uncertainty can often be increased due standard practices employed to manage an IT environment. Failing to consider such uncertainty may lead to incorrect conclusions. One such example is that of trying to determine an IP address based on evidence involving logs from an Internet Service Provider (ISP) using Network Address Translation (NAT) to conserve IP addresses, as related by Casey [34]. NAT is a fairly standard practice amongst ISPs, but failing to account for its use will render the evidence incorrect.

Given the above example, one may very well ask how uncertainty should therefore be measured and dealt with. In answer to the obstacle of measuring evidence certainty, Casey [34] created a scale to measure certainty, called the *Casey Certainty Scale* (hereafter referred to as the certainty scale).

The certainty scale is designed to address the need for providing estimates of certainty and to provide a framework for measuring certainty. In addition, it assists in translating the derivation of certainty (a very technical matter) into a non-technical result that can be interpreted by an audience with little or no technical background (for example, the average judge or juror). Similar types of scales have proven useful in areas such as meteorology, such as the Beaufort Wind Scale and the Fujita Tornado Scale.

On the certainty scale, evidence is ranked in terms of certainty by using 7 levels or grades, namely:

- C0 - Erroneous or incorrect;
- C1 - Highly uncertain;
- C2 - Somewhat uncertain;
- C3 - Possible;
- C4 - Probable;
- C5 - Almost certain; and
- C6 - Certain.

The certainty values provide “a method to assist the digital evidence examiner to denote the level of certainty he or she has in a given piece of evidence in a given piece of context” [35].

The certainty scale is intended to assist audiences in understanding how much weight an investigator has given to a specific piece of evidence. In doing so, audiences can assess not only the pure evidence but also gauge the role played by the evidence in the final conclusion(s) made by an investigator and how certain the investigator is of such conclusions.

The following example illustrates the concept: An email is sent from the corporate accountant’s email account to that of the CEO, claiming to take credit for stealing a large sum of money. The evidence shows the accountant was not logged in to the computer at the time, and when interviewed, the accountant stated that he was on leave.



The investigator has graded the email header with a certainty of C2 (somewhat uncertain), whilst the rest of the content has been assigned a value of C1 (highly uncertain) due to signs of tampering.

An audience (such as jurors or judiciary officials) can now appreciate that the forensic evidence stating that the accountant was not logged in at his computer shows signs of tampering. Furthermore, this evidence can most likely not be trusted. Therefore, although the evidence appears to indicate the contrary, the accountant can most likely be considered to have been logged on at the time the email was sent. Furthermore, although there is no corroborating evidence, the email is shown to have originated from the accountant's mailbox. It is therefore increasingly likely that the accountant, in fact, was responsible for the ft.

As illustrated by the example, the scale is flexible enough to grade different parts of the evidence whilst maintaining the same framework. Additionally, the example also shows how the certainty scale allows non-technical people to understand the context and nuances of evidence certainty. As the scale is not based on technicalities, the audience can easily relate to it and use it to successfully evaluate evidence from a legal point of view without being caught up in the technical details.

From the discussion in this chapter on the digital forensic process and the practice of digital forensics, one can conclude that the purpose of digital forensics is to allow for the collection of digital evidence in such a manner that the evidence is beyond reproach.

Furthermore, the reader is left with an understanding of the essence of the investigation process, the economic impact of an investigation and an appreciation for what is seen as evidence in a digital forensics investigation. Finally, the reader has been made aware of the importance of keeping the integrity of the evidence and the role of establishing a known error rate for evidence by means of grading evidence based on certainty.

## 2.5 Conclusion

The purpose of this chapter was to introduce the broader background of digital forensics and to highlight the forensic processes, as well as the advantages of instituting a forensic readiness strategy.

The chapter opened with a brief definition of digital forensics and then proceeded to discuss the broader concept of the digital forensic process. The digital forensic process was broken down into a brief comparative review of five different models describing the digital forensics process.

The chapter subsequently focused on the economics of a digital forensic investigation, when performed according to the process highlighted previously. It was noted that digital investigations are typically very costly, and the various factors influencing the cost of a digital forensic investigation were discussed. It was concluded that the main drivers of investigation cost are the time taken to perform a thorough investigation and the loss of revenue suffered during the investigation.

These factors were used to direct the discussion towards forensic readiness. It was proposed that the institution of a forensic readiness strategy may minimise the time necessary for a digital forensic investigation and that investigation cost and loss of revenue can therefore be minimised.

The purpose of this part of the chapter was to provide the reader with background know-

ledge of forensic readiness. Forensic readiness forms one of the primary focuses of the later work detailed in Chapters 5 and 7.

Having completed the discussion on the digital forensic process, the focus subsequently shifted to the digital forensic practice.

This section of the chapter opened with an overview of the various regulatory requirements in terms of digital forensics and how these affect the process. One of the key items noted in this overview was the importance of evidence and how such evidence should be treated.

The topic of digital forensic evidence therefore became the focus of the next part of this chapter. This part of the chapter opened with a definition of evidence and a synopsis on the need for evidence. Subsequently, the legal perspective was discussed, including how the law of evidence and burden of proof influences evidence. This led to an overview of what should be considered as evidence and how the integrity of evidence should be protected.

This part of the discussion provided the reader with important background to evidentiary requirements in a legal construct, used in Chapter 7 and formed the basis for the discussion around the scientific properties of digital forensics in Chapter 6.

As part of the discussion on what should be considered as evidence, the Daubert Standard (used by some to determine the admissibility of evidence) was referenced and specific emphasis was placed on the criterion requiring a known error rate. Again, this forms the foundation for the discussion in Chapter 6.

Finally, the discussion focused on evidence certainty and the Casey Certainty Scale was introduced as a means of grading evidence based on the possible rate of error associated with it. It was noted that the conscientious investigator should not only present raw evidence, but should also provide audiences with a view on the certainty involved in each evidentiary fact and how this relates to the bigger burden of proof.

Given the information in this chapter, the reader should by now have an overview of the field of digital forensics as well as specific topics that will form the foundation for further discussions in later chapters. In the next chapter, the concept of XML and XML accounting will be introduced.

## Chapter 3

# XML and XML-based Accounting Formats

### 3.1 Introduction

This chapter provides a general overview of XML and also provides an overview of XML accounting formats: defining the concepts, discussing how these are used, as well as providing a synopsis of the different accounting formats available. Furthermore, emphasis is placed on the key weaknesses in XML which, together with its usage in storing financial information, form the foundational aspect of the further work proposed in Chapters 5 and 7.

The purpose of this chapter is to provide the reader with an overview of XML, give the reader insight into the pertinent issues facing data storage using XML and finally, most importantly, introduce the concept of XML accounting formats. XML and specifically the use of XML accounting, is a central theme of the work presented in Chapters 6 and 7.

Since nearly all information is computerised in today's world, computers have become pivotal in data processing, especially concerning business data. In recent years, the production and processing of information has become the main commodity and currency responsible for driving business. Webster [166] terms this phenomenon the 'knowledge economy'.

The astounding rate of information production results in overwhelming quantities of information being available, resulting in 'information overload' [55]. Both Feather [55] and Gartner [61] observe that this phenomenon is increasingly found in business. Information overload in business often results in poor decisions which fail to consider the required factors, resulting in financial loss. In 2009, the well-respected research firm Gartner [61], predicted that throughout 2012 "more than 35 per cent of the top 5,000 global companies will regularly fail to make insightful decisions regarding significant changes in their business and markets".

Typical business environments, suffering from information overload, often have business data stored in a variety of systems and applications. Piecing together these various sources of data into a coherent picture is not an easy task and Hoffman et al [74] conclude that two factors are key to complicating the process, namely: a lack of semantic awareness and a lack of open formats.

Computerised searches for information often return irrelevant information for search terms. Such results often contain the search term but do not take context into account, rendering it irrelevant, for example the search for the term 'school' in Use Case 3.2.1. This is due to lack of semantic awareness. Computers typically cannot distinguish the semantic meaning of the

information, therefore often returning irrelevant information.

Also, when data is received in a different format of which the specification is not open, such data can often not be interpreted by the recipient computer and ends up being unusable.

By enabling both computerised interpretation of data and providing a flexible and simple way to format data [70], XML greatly assists business. In the next section, an overview of XML is provided, whilst the subsequent section explains how using XML enables business to combat information overload.

## 3.2 What is XML?

As mentioned earlier, data is typically scattered over various financially relevant systems. Furthermore, data sharing between these systems is typically impractical due to the lack of a common data format to facilitate such an exchange [7]. Data formats play a key role and are discussed further in Section 3.2.3.

In the introduction of this chapter, it was stated that timely, relevant and accurate financial information is essential to business. However, business is regularly unable to leverage such information [61] due to a combination of data fragmentation and data incompatibility [7].

This is one of the key reasons for the introduction of the eXtensible Markup Language (XML) [70]. XML enables the exchange of data using a simple and common data format.

In this section, the idea of data exchange using XML will be discussed in detail. XML will play a central role in the solution developed in later chapters.

### 3.2.1 Overview of XML

Prior to a detail discussion of XML, a brief definition is required.

XML is a simple meta-language, derived from SGML (the Standard Generalised Markup Language) [149]. SGML was initially intended to allow for the generalised markup of documents, but due to its rigid document definition suffered from a lack of adoption [87]. As a result, XML was subsequently developed by the World Wide Web Consortium (W3C), allowing for markup without a rigid definition [149].

XML is therefore defined as a markup meta-language. XML describes the rules and constructs which, in turn, can be used in the creation of a specific instance of markup language [70, 8]. The definition above is comprised of two components relevant for subsequent work, namely ‘extensibility’ and ‘markup’.

Extensibility is a core software engineering principle, defined as “an implementation that considers and allows for future growth” [8]. In the context of XML, it means that the XML markup language can easily be modified and extended in order to support additional or even different target data.

Markup refers to the ‘tags’ or ‘marks’ which are inserted into data to “identify pieces of information” [74]. Therefore, markup is simply a way of adding information to data in order to describe it. The length attribute in the sample XML document in Figure 3.2.1 is an example of such a markup attribute. It provides additional information regarding the tag value, in this case stating that the value, short, is a length attribute of the document.

Markup tags delineate semantic meaning, for example enabling the consumer of marked up data to know that the sample document in Figure 3.2.1 has a short length and has a name,

Sample.xml.

XML tags are not fixed by a standard (i.e. allowing only specific tags), and therefore XML is known as an *extensible markup language*. The creator of the XML file may use his or her own discretion to decide the tag structure of the document, unlike other markup documents in languages such as HTML.

The above XML overview can be better illustrated by means of the following example.

### 3.2.2 Example of XML Usage

Consider a sample XML document, as introduced in Example 3.2.1.

**Example 3.2.1.** *Sample, short XML file:*

```
<?xml?>
  <document>
    <name type='`first`'> Sample.xml </name>
    <length> short </length>
    <purpose> To illustrate the usage of XML and its meta tags.
    </purpose>
  </document>
</document/>
```

In Example 3.2.1, the structure of an XML document is illustrated and the usage of markup tags to delineate data is practically illustrated.

Each pair of XML markups (or tags) defines an element, which in turn may act as either text (e.g. the ‘purpose’ tag in Example 3.2.1) or a container for another element (e.g. the ‘document’ tag in Example 3.2.1).

From this example it is easy to see that XML documents have a particular structure and are constructed according to particular rules. Dykes and Tittel [51] and Kirkpatrick [87] note the below key rules for constructing well-formed XML, acceptable to an XML parser. Each of these will also be linked back to Example 3.2.1 in order for the rule to be observed in practice.

- All XML documents should start with an XML declaration. This statement notifies the parser that the document is an XML document and is identified as the ‘<?xml? >’ line in the example.
- All XML documents start from a single root. A single XML tag should be the parent tag (root tag) of the document enclosing all other tags, such as the example’s ‘< document >’ tag.
- All XML tags must be closed. Tags should always appear in pairs and should always be surrounded by angled brackets ‘<’ and ‘>’. Furthermore, the closing tag should always start with a forward slash (‘/’), e.g. the ‘< /document >’ tag Example 3.2.1.
- Empty tags should be formatted appropriately. Empty tags may use a short-hand notation called the self-closing tag, such as the ‘< document/ >’ line in Example 3.2.1).

- Tags should be properly nested. XML tags should always be closed in the order in which they are opened, note how the ‘purpose’ tag is closed prior to closing the ‘document’ tag in the example.
- Tags should be of the same case. XML is case-sensitive, so XML tags should stick to the same case, generally recommended to be lower case. Note how the ‘document’, ‘purpose’ and ‘length’ tags are all in lower case.
- Attribute values are specified in quotes. Whenever an attribute for an XML tag is defined, its value should be specified using quotation marks, such as setting the ‘type’ attribute to ‘first’ the ‘name’ tag of Example 3.2.1.

Of course, XML is far more complicated than simply applying a couple of simple rules. A full discussion of XML structure falls outside of the scope of this discussion, but the interested reader is referred to [51, 93, 8].

Now that XML has been briefly introduced by way of an example, the next key point for discussion is the rationale behind using XML.

### 3.2.3 Why Use XML?

The current section provides an overview of XML to the reader. In the previous subsections, XML has been defined and a brief overview of how XML is constructed has been provided. However, the question of XML’s importance and motivation for its use is still not addressed. These will therefore become the focus of this subsection.

Two main drivers were instrumental in creating a need for an extensible language like XML: 1) the need for accurate information classification and 2) data exchange, most notably business-to-business communication and business-to-consumer communication [74, 64].

#### Information Classification by Embedding Semantics

Berners-Lee et al [17] note that there is a growing need for data integration across multiple data domains (or disciplines) such as biology, chemistry, accounting, etc. Increasingly, data is needed from multiple fields and the data should be integrated in order to obtain a holistic view.

In this subsection, it will be demonstrated how XML aids in providing meaning to data and how it assists in integrating data from various sources. However, such data integration is significantly more complex than one might imagine at first glance.

Consider a sample use case.

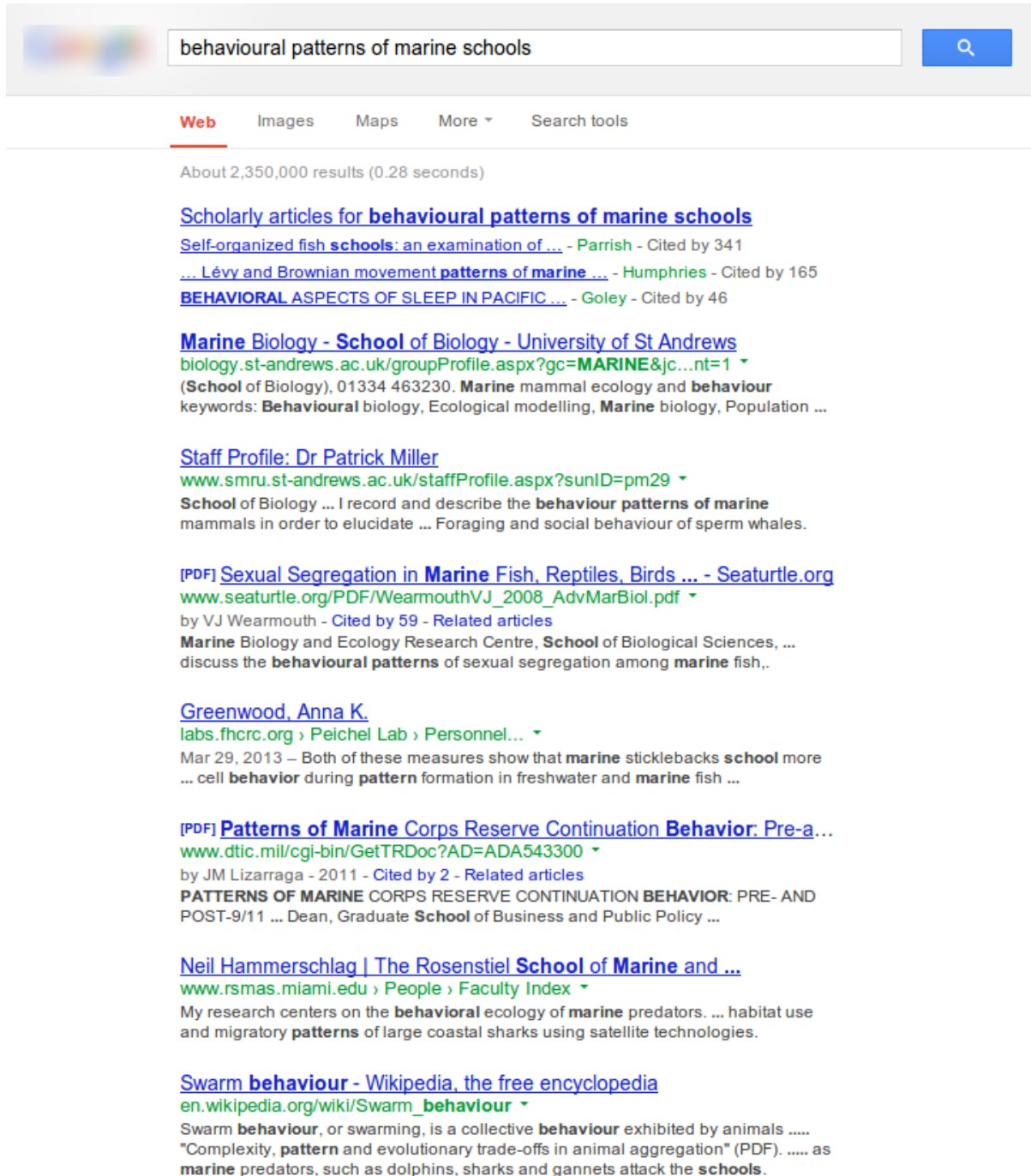
**Use Case 3.2.1.** *Information is required regarding the behavioural patterns of large congregations of marine animals (known as schools).*

Figure 3.1 shows the results of the search query “behavioural patterns of marine schools”, entered into a popular search engine, on 17 June 2013. Note the results returned by the search engine and how it returned various results that are not in any way relevant to Use Case 3.2.1. The result set will now be broken down by each key area of misinterpretation.

Firstly, note the results for the term ‘school’. The search results related to the term ‘school’ returned academic institutions, not the collective noun for marine animals as was intended. Furthermore, consider the results for the term ‘marine’. The search returned results related

to the United States' military institution of the same name, ignoring the intended meaning of "animals residing in the ocean".

Finally, the search yielded tangentially related search results instead of only returning relevant results. For example, the search returned the name of a professor at an academic institution that is interested in swarm behaviour, instead of information about the behaviour of collections of marine animals.



The screenshot shows a search engine interface with the query "behavioural patterns of marine schools" entered in the search bar. Below the search bar, there are tabs for "Web", "Images", "Maps", "More", and "Search tools". The search results are displayed below, showing approximately 2,350,000 results in 0.28 seconds. The results include:

- Scholarly articles for behavioural patterns of marine schools**
  - [Self-organized fish schools: an examination of ...](#) - Parrish - Cited by 341
  - [... Lévy and Brownian movement patterns of marine ...](#) - Humphries - Cited by 165
  - [BEHAVIORAL ASPECTS OF SLEEP IN PACIFIC ...](#) - Goley - Cited by 46
- Marine Biology - School of Biology - University of St Andrews**
  - [biology.st-andrews.ac.uk/groupProfile.aspx?gc=MARINE&jc...nt=1](http://biology.st-andrews.ac.uk/groupProfile.aspx?gc=MARINE&jc...nt=1)
  - (School of Biology), 01334 463230. Marine mammal ecology and behaviour
  - keywords: Behavioural biology, Ecological modelling, Marine biology, Population ...
- Staff Profile: Dr Patrick Miller**
  - [www.smru.st-andrews.ac.uk/staffProfile.aspx?suniD=pm29](http://www.smru.st-andrews.ac.uk/staffProfile.aspx?suniD=pm29)
  - School of Biology ... I record and describe the behaviour patterns of marine mammals in order to elucidate ... Foraging and social behaviour of sperm whales.
- [PDF] Sexual Segregation in Marine Fish, Reptiles, Birds ... - Seaturtle.org**
  - [www.seaturtle.org/PDF/WearmouthVJ\\_2008\\_AdvMarBiol.pdf](http://www.seaturtle.org/PDF/WearmouthVJ_2008_AdvMarBiol.pdf)
  - by VJ Wearmouth - Cited by 59 - Related articles
  - Marine Biology and Ecology Research Centre, School of Biological Sciences, ... discuss the behavioural patterns of sexual segregation among marine fish,.
- Greenwood, Anna K.**
  - [labs.fhrcr.org](http://labs.fhrcr.org) › Peichel Lab › Personnel...
  - Mar 29, 2013 – Both of these measures show that marine sticklebacks school more ... cell behavior during pattern formation in freshwater and marine fish ...
- [PDF] Patterns of Marine Corps Reserve Continuation Behavior: Pre-a...**
  - [www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA543300](http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA543300)
  - by JM Lizarraga - 2011 - Cited by 2 - Related articles
  - PATTERNS OF MARINE CORPS RESERVE CONTINUATION BEHAVIOR: PRE- AND POST-9/11 ... Dean, Graduate School of Business and Public Policy ...
- Neil Hammerschlag | The Rosenstiel School of Marine and ...**
  - [www.rsmas.miami.edu](http://www.rsmas.miami.edu) › People › Faculty Index
  - My research centers on the behavioral ecology of marine predators. ... habitat use and migratory patterns of large coastal sharks using satellite technologies.
- Swarm behaviour - Wikipedia, the free encyclopedia**
  - [en.wikipedia.org/wiki/Swarm\\_behaviour](http://en.wikipedia.org/wiki/Swarm_behaviour)
  - Swarm behaviour, or swarming, is a collective behaviour exhibited by animals .....
  - "Complexity, pattern and evolutionary trade-offs in animal aggregation" (PDF). ..... as marine predators, such as dolphins, sharks and gannets attack the schools.

Figure 3.1: Search query results illustrating lack of semantic awareness.

As can be seen from the example above, search engines employing search methods which

do not recognise semantic context have great difficulty in establishing semantic meaning (i.e. context and meaning of the search terms that go beyond the actual occurrence of a word). In the example, the lack of provision for semantic meaning disallowed the search terms to specify that information on schools of fish is required instead of information on places of learning.

From the search results in Figure 3.1 one concludes that the absence of provision for semantic context leads to either valid, but inaccurate search results; or exclusion of relevant search results. Valid, but inaccurate search results are results that contain the search terms but are not related to the desired results (when considering the semantic intention). Exclusion of relevant search results, on the other hand, is a result that did not contain the exact search terms, such as “behaviour of swarms in ocean settings” as opposed to “behaviour of marine schools”.

Berners-Lee et al [16] noted the difficulty in successfully integrating data using current search strategies and proposed a solution termed ‘The Semantic Web’.

It proposes that data be published directly in language which allows for semantic information to be added to the source data. XML was developed, amongst others, to allow semantic data to be added to raw data, enabling the creation of the Semantic Web [17]. XML is seen as a foundation for creating information which conforms to the specifications of the Semantic Web and allows one to “recruit the right data to a particular use context” by means of employing tags to provide semantic meaning to specific data instances.

## Data Exchange

XML can also be used to facilitate sharing of data by making use of customisable data formats. This section will focus on how XML can be used to exchange data by making use of an open data format (a common, well defined data format) and how XML can be used to mark up data in order to fit the defined format.

Prior to overviews XML’s role in data exchange, however, it is necessary to define data formats and explain the difference between proprietary and open data formats. When storing data on a computer, a *data format* is used. A ‘data format’ refers to the way the data is laid out and organised within a data file [122]. Software packages typically elect to each use their own data format, because of data protection and specific performance gains. However, such usage usually also implies that data cannot be easily exchanged. This is due to the absence of data translation middleware<sup>1</sup> to cater for translation between the data formats required.

Data formats can be classified into either ‘proprietary’ or ‘open’ formats [3].

A proprietary format, also known as a ‘closed’ format, is a format of which the internal specifications governing the arrangement and representation of data is the intellectual property of another party and is often kept secret [3]. Typically proprietary formats require royalty fees to use, and constricts the user by requiring proprietary software to interpret the data contained in the format [10].

An open format, on the other hand, is a format of which the specification is known, i.e. made public, and:

- is maintained by an independent standards organisation; or
- has not been recognised as intellectual property; or
- the owner thereof has relinquished his or her right to the intellectual property.

---

<sup>1</sup>Software located between two or more separate systems, typically responsible for data interchange [56].



Proprietary file formats pose two significant drawbacks to storing data, namely: inaccessible data, and obsolescence [10, 112]. Each will be briefly discussed below.

Proprietary formats introduce the risk of the encoded information not being accessible by the recipient, due to the recipient not having access to the correct proprietary software to decode it. This leads to inaccessible data due to the format it is stored in, despite the fact that the stored data itself is perfectly intact and not damaged in any way.

Furthermore, proprietary formats used to encode data may become obsolete, rendering the information stored in it inaccessible. Again, although the data itself may be completely unharmed, it is inaccessible due to the data format not being readable.

One of the primary drivers for the development of XML is the scenario where the recipient is not able to read and utilise received data. As can be concluded from the above, this is a very real risk when utilising proprietary data formats. XML has been key in the process of establishing open formats, as it allows for the markup of data according to well-defined rules of known specification, called Data Type Definitions (DTD) [112, 149, 3].

A second driver for the development of XML was the risk of data obsolescence due to support for specific proprietary formats being discontinued. In the case where the company responsible for proprietary format 'X' goes bankrupt or its software does not remain backwards compatible, information stored in format 'X' will not be accessible anymore. In such a case, the rationale for rather storing data in an XML format is simple, as the data remains human-readable and can be extracted easily. Because XML data formats are inherently open, such formats will be far less likely to become obsolete than closed, proprietary formats [10].

XML accomplished the goal of enabling wide data exchange remarkably well. Roy and Ramanujan [137] note that XML especially excels at "seamless and efficient" data transfer between applications. They further note that due to the text-based format of XML, all platforms can easily interpret it, especially when XML is used as the least common denominator for expressing information.

These characteristics combine to make XML a very suitable medium for information exchange between organisations or between different platforms within organisations.

### **3.3 XML Finance and Accounting Formats**

In the previous section, the rationale behind the use of XML was discussed. Furthermore, it was pointed out that business is heavily dependent on the availability of accurate and timely financial data, which is made possible by the use of data exchange formats such as XML.

This section will now explore the use of XML with regards to financial data, considering that XML financial information becomes a central theme in the later work performed in Chapters 5 and 7.

XML, released on 10 Feb 1998, [163] has been widely adopted in information exchange [75, 110, 37]. More than 100 industries worldwide (e.g. airlines, financial, insurance and news to name but a few) have adopted the use of XML in the operation of their business [37].

A key application of XML is found in the financial sector [110]. One of the largest XML-related successes in business is the popular format for reporting business results, called XBRL (the extensible business reporting language) [110].

XBRL is amongst the best-known and most widely used XML standards and is chiefly used to define and exchange financial information such as financial statements [29]. XBRL

allows for reporting business and financial information, and provides standard tags to delineate financial data, e.g. tags for accounts receivable, sales and net profit. Also, XBRL allows for the specification of additional financially relevant information, e.g. the currency of monetary entry or even the period in which the item was reported [29].

Leveraging XBRL allows for rich financial data, providing a much deeper insight in the flow of financial information between different stakeholders and software packages than previously achieved using closed and proprietary formats of financial data exchange. XBRL illustrates the demand for exchanging financial data in a widely recognised format, and its swift adoption rate amongst business enterprises indicates the dire need for financial data exchange.

To show just how popular this format is, consider the following statistics. In February 2005, the United States of America's Securities and Exchange Commission (SEC) issued a ruling stating that registrants may submit XBRL-tagged information [29]. By May 2009, 340 of the Fortune 500 (the 500 largest companies listed on the New York Stock Exchange) started to file quarterly and annual reports using XBRL [98] and XBRL reporting finally became mandatory for companies with a market capitalisation in excess of US\$ 5 billion on 15 June 2009.

XBRL is however not the only instance where XML has proven popular for the exchange of financial information. XML is used not only for financial reporting (such as XBRL) but also for storing and exchanging other financially relevant information [12], such as:

- real-time, electronic exchange of securities transactions using the Financial Information eXchange Protocol (FIX);
- data interchange for financial markets using the Market Data Definition Language (MDDL); and
- inter-bank and other sensitive financial transactions between financial institutions, using SWIFT standards (a standard developed by the Society for Worldwide Interbank Financial Telecommunication).

In addition to being deployed to store numerous instances of financially-related data, XML is also increasingly used to store financial source data (i.e. accounting data). Storing accounting data with XML enables the simple exchange of financial data between different accounting applications and also different stakeholders in the accounting process, e.g. financial institutions, debtors, creditors and management.

A multitude of XML accounting formats exists, due to the freedom XML allows in specifying formats for storing data. Many of these are not formal standards, but instead are created by user groups on an ad-hoc basis. In order to allow for a more formal discussion, such standards will be excluded from the remainder of this discussion. Instead, focus will be restricted to accounting formats specifically created by standards organisations, due to the wider adoption of such standards.

Several notable formally-defined XML accounting formats exist [48, 64, 74, 107], such as OFX, IFX, UBL and XBRL GL.

The Open Financial Exchange, better known as OFX, evolved as a result of a combined effort between Microsoft (leveraging Microsoft Money's Open Financial Connectivity (OFC) format), Intuit (developer of the popular QuickBooks accounting software as well as the Quicken personal finance software) and Checkfree [64, 74].

OFX was developed to facilitate financial transactions between accounting packages and banking institutions. In addition, it allows for bill payment, investment and financial planning, amongst other features [64].

Another XML accounting format, the Interactive Financial Exchange (IFX) is developed by the Interactive Financial eXchange Forum [77]. IFX is an XML format detailing electronic bill presentation, used for payment and business-to-business communication (banking, payment, etc), automatic teller machine communications and consumer-to-business communication such as payments and banking.

The Universal Business Language [24] consists of a library of standard business documents in the XML format (e.g. invoices and purchase orders) and has been developed by OASIS. OASIS, the Organisation for the Advancement of Structured Information Standards, is a non-profit organisation dedicated to driving the “development, convergence and adoption of open standards for the global information society” [24]. Major stakeholders of OASIS include well-known organisations such as IBM, Novell and SAP.

Finally, XBRL GL is a subsection of XBRL, the eXtensible Business Reporting Language [107]. It caters explicitly for the exchange of General Ledger, balances and complete accounting ledger information. Although not a fully specified accounting format, the motivation behind XBRL GL is to allow users to export the critical accounting information (the General Ledger) from one accounting application and then import it again using another accounting application.

Using XML formats to exchange financial information is efficient and advantageous in many ways, such as cost savings, reduction in opportunity for data capturing fraud, publicity, and regulatory compliance.

As XML eliminates much of the manual interaction and entry of financial information between different systems [127], it allows for cost savings and provides less opportunity for data capturing fraud. Also, formats like XBRL are currently fashionable and using such formats shows potential investors that the company in question is a ‘thought leader’ [107].

Finally, compliance with regulatory requirements and international best practice is made significantly simpler when using an XML format that enforces compliance automatically, such as XBRL [29].

Whilst increased regulatory compliance and diminished opportunity for fraud may be argued, MacMillan [98] holds an alternate view. He notes that using XML in financial data will not necessarily lower the levels of fraud, as one could still input incorrect numbers and details into financial information stored in XML format. MacMillan further notes the results of a study concluded by the North Carolina State University in June 2009 revealing that multiple serious errors occurred in XBRL-documents submitted during the SEC XBRL-filing trial in 2005. Microsoft (one of the 22 Fortune 500 companies taking part in the SEC XBRL-filing trial), for example, reported two US\$ amounts in millions instead of billions due to a data entry error.

This shows that errors (both deliberate and bona fide) may still occur despite the use of a more transparent data format such as XBRL. Such instances seem to suggest the use of an automated tool to check XML accounting data for errors. Such a tool and ideas related to this tool are discussed in Chapter 5.

Although XML is an innovative technology that solves very pertinent and pressing problems, there are very real risks to its usage and careful thought should be given to how it is deployed. This is especially true in risk-sensitive environments like the financial sector, where a single incident of misreporting can have the grave effect of severe monetary loss. The next section is dedicated to discussing some of the weaknesses noted by the use of XML.

### 3.4 Concerns Around XML Usage - How XML's Design Provides Opportunity for Data Manipulation

As noted in the above sections, XML has become vitally important in data exchange, especially financial data. Given the ubiquity in the use of XML for exchanging financial data, and the importance of this data, no discussion on XML can be complete without discussing the potential weaknesses associated with XML.

One of the cornerstones of the further work discussed in Chapters 5 and 7, is that of the possible exploitation of the human-readability requirement of XML.

The purpose of this section is to provide an overview of human-readability and other weaknesses found within XML. Furthermore, the possibility of data manipulation (by means of exploiting these weaknesses) will be introduced, as part of a brief synopsis of XML using digital forensic investigation as the departure point.

In an article from 2000, Roy and Ramanujan [137] raise several /significant drawbacks to the use of XML:

**Limited data type support** - since XML is a text-based format, it does not directly support complex data types like images or video. The only way to circumvent this limitation is by encoding the data as text (for example base-64 encoding of images as used by email).

**Absence of data typing** - XML does not check for errors in the source data stored within a document.

**Lack of standard vocabularies and/or tag sets** - common data sets are not described by a standardised XML data type definition. Each implementer is free to use its own definition of key terms such as 'customer' or 'invoice'.

**No inherent security features** - XML does not provide any inherent security features to protect its content when transmitted over public networks such as the internet. Add-ons like Secure Socket Layer (SSL) encryption must be used to encrypt the payload before the XML content can be sent over public infrastructure.

Despite Roy and Ramanujan's article [137] being written in 2000, the lack of security features remain a grave concern. The lack of proper security in XML is noted by several other authors in the context of XML drawbacks [70, 131, 162, 167]. Lesser concerns above, such as limited data type support and the lack of standard vocabularies, are also found in more recent work on XML problems and weaknesses.

In "The challenges that XML faces" [70], Grado-Caffaro and Grado-Caffaro emphasise two XML problems, namely XML divergence and security weaknesses in XML.

XML divergence is caused by the myriad XML types and ways to use the meta-language. Due to the many ways that XML is used, all its different implementations and the lack of standardisation across XML (a concern listed by Roy and Ramanujan [137] as well) Grado-Caffaro and Grado-Caffaro are concerned that the technology may "splinter into various versions that work in different ways" [70].

In addition to the lack of standardisation, Grado-Caffaro and Grado-Caffaro [70] also note a number of security weaknesses in XML, reflecting on similar weaknesses as Roy and Ramanujan [137].

Another key source of concern regarding XML security is noted by Vaughan-Nichols [162], namely that XML is presented in plain text which makes it vulnerable to man-in-the-middle attacks during transmission. He also points out another risk — hackers could use the data tags as additional clues determining which data files to target, e.g. looking for tags that state “credit card number” will lead hackers to documents containing card numbers which can be stolen.

Furthermore, Vaughan-Nichols [162] also notes that XML has difficulty in dealing with large data files. He states that large collections of data (such as the data found in Bio-Informatics data sets) expand vastly when adding XML tags and descriptors. This, in turn, places unnecessary data load strains on corporate networks, file servers and storage infrastructure. In essence, one can summarise this concern raised by Vaughan-Nichols as stemming from the verbosity caused by marking up data in XML format.

Ray [131] expands on Roy and Ramanujan’s concern regarding XML’s limited data type support and furthers Vaughan-Nichols’ criticism that XML fails to scale when the data sets become very large by stating that: 1) XML is not compact and 2) XML is not optimised for speed of access.

Ray also notes that XML documents are designed to be loaded in their entirety before being used as a data source. Combining this observation with Vaughan-Nichols’ comment regarding large data files, one is led to conclude that a very powerful system would be required to process large data sets in XML format, such as historical financial statements for all companies listed on the New York Stock Exchange, to mention an example.

However, in my opinion, the single most vital concern out of all the above remains the weaknesses around XML security. Grado-Caffaro and Grado-Caffaro summarise these concerns succinctly [70]:

- The open nature of XML provides a target for hackers.
- An inherent flaw exists in the XML processor’s assumption that XML tags accurately describe their content.
- XML content is in human-readable form and such unprotected content may be modified in an unauthorised way by a third party.

The key item raised by Grado-Caffaro and Grado-Caffaro [70] is the final criticism above and it forms the central theme addressed in the later chapters. In “New XML-Based Files: Implications for Forensics” [59], Garfinkel and Migletz concur with this risk being the greatest of all the security concerns posed by XML usage. They rightly observe that the ease of modifying XML, introduced by human-readability, makes it exceptionally easy to engage in malicious modification of XML data.

### 3.5 Conclusion

In this chapter XML was introduced. In particular, XML as a way of providing meaning to data was discussed, as well as how the use of XML eases data exchange. Also, it was shown how the design goal, allowing for data exchange (human readability), can be exploited in an attempt to modify data in an unauthorised manner, such as committing fraud.

The chapter pointed out the great need for timely, relevant and accurate financial data in the business context. It was further noted how the exchange of data and semantic awareness

of data became the main drivers behind the development of a data format that is open and not proprietary.

Subsequently, XML was introduced as a solution to the problem of data exchange. The chapter provided an overview of XML, defining it and explaining its functioning and usage.

In addition to introducing the reader to XML, the chapter also provided several arguments for the usage of XML and pointed out that XML is used for two key reasons: 1) information classification by means of semantic data awareness, and 2) data exchange.

At this point, it was noted that businesses rely heavily on data exchange and the use of XML in business applications (by means of XML accounting formats and XBRL) was introduced. An overview of several XML accounting formats was provided, including a brief discussion on the motivations for using such formats.

Finally, the chapter concluded by noting several concerns around the usage of XML, in particular one key weakness affecting XML security. This weakness is summarised by Garfinkel and Migletz [59] as follows: the ease of XML modification (as a result of the human-readability required by the standard) makes it exceptionally easy to engage in making malicious modifications to XML files. This consideration becomes foundational for the work performed in later chapters.

# Chapter 4

## Compilers and Compiler Theory

### 4.1 Introduction

The purpose of this chapter is to provide the reader with a background regarding compilers as well as a definition of a compiler. This chapter briefly overviews the compilation process, with specific focus on the lexical, syntactic and semantic analysis parts of the compilation process. Subsequently, the concept of grammar and the parsing of such a grammar are also discussed. Finally, the chapter also considers parser errors and error handling.

One could now ask: “What is the motivation for including a chapter on compiler theory”? The rationale behind including a chapter on compiler theory is that compiler construction and error handling plays a fundamental role in future work (see Chapter 5).

Compiler theory is a subject with a rich history which has been studied over many years. As a result, compiler theory has developed a deep understanding of grammar specifications and the formal methods involved in testing whether input conforms to the grammar specifications [5, 145].

Therefore, using set theory and formal mathematics [102, 145], it is possible to prove that parsing and analysis techniques (subsets of compiler theory) are able to evaluate input against a given set of rules (a grammar) and conclude correctly whether the input could be generated using the rule set provided. This enables one to determine whether a specific set of input complies with a set of specifications or does not [5].

Using this reasoning, compiler theory can be applied to assist with forensic investigation of XML accounting data files. Consider the use of known file exclusion [151] functionality provided by most computer forensic software. Known file exclusion is based on argument that all known operating system files can be fingerprinted using hashes, and if these hashes match a file under investigation, can be excluded from the investigation as the file has already been proven as ‘cleared’ or ‘normal’. As a result, the surface area of the investigation is considerably narrowed, leading to greater efficiency and less investigation time.

Similarly, using compiler theory to determine whether a set of input compiles with a specified grammar, will enable an investigator to determine whether an XML accounting data file is consistent with the rules governing such a file (i.e. the file is normal and can be excluded from the investigation) or whether there are elements within the file which are not consistent with the rules governing such a file (i.e. the file is not normal and should be targeted for investigation).

It is therefore of key importance to provide an overview of compiler theory as it is foundational to the later work performed in Chapter 5. It is understood that many of the readers of

this dissertation may be intimately familiar with compilers and how they function, as this is a subject of study in many undergraduate Computer Science degrees. However, as the wider audience may not be familiar with this topic, it is important to discuss some key compiler related concepts (such as syntactic and semantic analysis, parsing errors and error handling) as these form the foundation of the later work presented in Chapter 5, which discusses the detection of data irregularities in XML accounting data using automated means.

Although the creation of a compiler is a complex task, the basic definition of a compiler is relatively simple. At its core, a compiler is a piece of software responsible for translation. Muchnick [106] defines compilers as “software systems that translate programs written in higher-level languages to into equivalent programs in object code or machine language for execution on a computer”.

Cooper and Torczon [44] note that for any compiler to function correctly, it must be able to perform two tasks: a) make sense of the input and b) translate any input received into output with equivalent functionality. Aho, Sethi and Ullman [4] confirm Cooper’s suggestion and remarks that compilation comprises of two stages, one to handle each task as suggested by Cooper. These are the analysis and synthesis stages.

Analysis is the breaking down of the input (also known as the source program) into its building blocks and creating an intermediate representation of these building blocks [4]. Synthesis on the other hand, uses the intermediate representation generated by the analysis stage to reconstruct the desired target language (which is typically machine language [106]).

It should be noted that the ‘compiler’ used in Chapter 5 is not a compiler in the traditional definition of the term, taking input and translating the input into output using analysis and synthesis. Instead, the ‘compiler’ referred to in Chapter 5 only consists of an analysis phase. It can therefore be argued that the term ‘compiler’ is not the correct term to refer to the means of automated detection of data irregularities, and that the term ‘parser’ should be used instead. This point is addressed in Chapter 5 where it is proposed that the term ‘compiler’ used throughout that chapter be redefined (see Definition 5.3.1).

However, for the remainder of this chapter the term ‘compiler’ will be used to indicate both analysis and synthesis. As synthesis is not required for the later application of the theory of compilers in Chapter 5, it will not be discussed in this chapter. Instead, the interested reader is referred to Chapters 8, 9 and 10 of ‘Compilers: Principles, Techniques and Tools’ by Aho, Sethi and Ullman [4]. The interested reader can also obtain further information in the work of Cooper and Torczon [44], Das [46] and Muchnick [106].

The remainder of this chapter focuses on analysis of the source program, introducing lexical, syntax and semantic analysis. The chapter then concludes by considering the various errors that could be encountered by the analysis phase and investigates strategies to handle such errors.

The next section introduces the analysis stage of the source input.

## 4.2 Analysis of the Source Program

As mentioned in the introduction of this chapter, the analysis phase is responsible for reducing the input to its building blocks and creating an intermediate representation of these building blocks.

Aho, Sethi and Ullman [4] remark that analysis is constituted out of three different, but consecutive, processes namely: ‘linear’ or ‘lexical’ analysis, ‘syntax’ analysis or ‘parsing’ and



semantic analysis.

Lexical analysis is responsible for reading the stream of characters making up the input file and grouping these characters into tokens. The next analysis phase, syntax analysis, groups the tokens into a hierarchy (tree) of nested collections called grammatical phrases with a collective meaning. The final analysis phase, semantic analysis, comprises of a number of checks to ensure that the components created during the syntax analysis phase fit together meaningfully and correctly.

In essence, these phases can be summarised as tokenisation, analysis of the construction of the input (what it looks like) and analysis of the meaning of the input.

Before discussing the various analysis phases, however, it is necessary to look at the structure of input and how this structure is defined. No discussion of analysis can be complete without first defining that which is analysed.

### 4.2.1 Understanding the Input

In order to understand how analysis of the input takes place, one first has to understand the input. As mentioned previously, for the sake of example, XML accounting data will be used as input in this chapter as it forms the basis of input in the work presented in Chapter 5.

One of the first key considerations regarding input is determining the rules defining the structure of input, in order to decide whether input received is valid or not. These rules are called the ‘syntax’ of the input [106]. Simply defined, syntax is the definition of what the input program looks like [4].

In order to specify the syntax, a formal definition of the rules governing the validity of the input is needed. This formal definition is called a ‘grammar’ and describes the hierarchical structure of the constructs used in the input.

Many variations and types of such formal rule definitions exist. A discussion of these types and variations fall of the scope of this discussion as the purpose of this chapter is not a detailed discussion of grammars. Instead, the interested reader is referred to Aho, Sethi and Ullman [4], Das [46] or Kakde [80] for an in-depth discussion of grammar types and the functionality and comparison of various grammars.

A grammar has four components [4, 46]:

- Start Symbol - a symbol designated by one of the non-terminal symbols.
- Terminal Symbols - a set of tokens, or sequence of characters with a collective meaning [4]. A terminal symbol represents an atomic unit, i.e. a unit that cannot be broken down into smaller units.
- Non-terminal Symbols - a set of units that can be broken down into smaller units.
- Productions - a set of productions, each consisting of a non-terminal (called the left-hand side), the derivation symbol, and sequence consisting of tokens and/or non-terminals, called the right-hand side.

Consider the following example to practically illustrate the above, based on the illustrative example by Aho, Sethi and Ullman [4]. A markup construct in XML is defined as below.

```
<num> 12345 </num>
```

In more generic terms, this can be stated as: `<tag> value </tag>`.

The markup construct consists of the opening angular bracket, a **tag** keyword, a closing angular bracket, a value, another opening angular bracket, a forward slash and a repeat of the previous keyword **tag** followed by a final closing angular bracket.

Using the variable *constr* to denote a construct and the variable *val* to denote the value, the construct can be written as:

$$constr \rightarrow < \mathbf{tag} > val < / \mathbf{tag} > .$$

The arrow above can be read as “may have the form”.

A rule such as the above is called a production. In the production, elements like the keyword **tag**, the angular brackets and the forward slash are called tokens. Variables like *val* consist of sequences of tokens and are called ‘non-terminals’.

A context-free grammar can therefore be seen as a grammar in which the left-hand side of each production is made up of a single non-terminal symbol only [44].

The concept of ‘grammar’ is key to the later work performed in Chapter 5 and form the foundation of the discussion on compiler errors.

Now that the specification of the input and a number of foundational definitions have been defined, the discussion can progress to discuss each of the analysis phases in detail.

An overview of each of these phases will be provided in the following subsections starting with the first of these, namely lexical analysis.

## 4.2.2 Lexical Analysis

Prior to discussing lexical analysis in detail, it is important to first define the concept of a token. A token is a “logically cohesive sequence of characters” [4] making up an entry that is “a legal member of the vocabulary of the language [106]” in which the input is written. Considering that XML accounting data is used as input in the later work in Chapter 5, it may be helpful to provide token examples using the same input. The following are examples of such tokens: language keywords (such as ‘account number’ or ‘transaction id’), punctuation (such as ‘,’ or ‘.’), delimiters (such as ‘<’ or ‘>’) and values (such as numbers).

The first of the analysis phases, lexical analysis, takes as input a character stream (originating from the input) and generates a stream of tokens [44]. In turn, this output is used for syntax analysis in the next phase of input analysis.

Together with this function, the lexical analyser is also responsible for the initialisation of a symbol table. A symbol table is a structure used to store each identifier, with fields for the attributes of the identifier. These are later used during semantic analysis to confirm whether the meaning of the input being analysed is correct.

In addition to tokenisation, the lexical analyser also performs two important secondary tasks.

The first of these secondary tasks is that of stripping out input that does not need to be parsed [4]. Such input typically consists of comments or characters that do not have semantic meaning, such as white space, tab characters, blanks or newline characters.

The second task performed by the lexical analyser is that of determining the validity of the tokens. This is done by applying a rule set (or grammar) to the tokens, and signalling a lexical error in the case where a character string cannot be tokenised. For a more detailed definition and an example of such an error, refer to “Errors and Error Handling” in Section 4.3. The lexical analyser also correlates the error messages from the compiler with the source code, for

example keeping track of the number of newline characters so that an error can be reported at a specific line number [4].

Lexical analysis and, by extension, input analysis will be of great importance in the further work performed in Chapter 5.

In next section the subsequent analysis phases, syntax analysis, will be discussed.

### 4.2.3 Syntax Analysis

The second stage of the analysis phase is that of syntax analysis. Syntax analysis is often called *parsing* [4, 44] and the parser is responsible for this part of the analysis.

As mentioned in the previous subsection, lexical analysis generates tokens from the input stream. The parser takes the output from the lexical analyser (the stream of tokens) as input. The parser is responsible for two main tasks: verifying the correctness of the tokens (and handling any errors occurring in the verification process) and constructing a parse tree or syntax tree [44, 80].

Firstly, the parser verifies that the tokens in the stream make sense given the syntax specified by the grammar. In the event where the tokens do not conform to the grammar, the parser will report an error to the user, called a syntax error. Also, the parser is responsible for recovering from the error and continuing processing if possible [153].

The second task of the parser is to construct a parse-tree, defined as a representation of a string's syntactic structure according to a context-free grammar [80].

The construction of parse trees and the various approaches facilitating such construction is a complex topic that falls outside of the scope of this discussion. Instead, the assumption is made that the output of the parser is a type of representation of a parse tree for the stream of tokens produced during lexical analysis. The interested reader is referred to [4], [46] or [80].

In the next section, the third and final stage of the analysis phase, semantic analysis, is discussed.

### 4.2.4 Semantic Analysis

The third and final stage of input analysis is that of 'semantic analysis'.

During semantic analysis, the semantic analyser builds up a base of knowledge about the detail encoded in the input program [44], such as how values are represented and the flow of values between variables. This 'construction of a knowledge base' is done by introducing semantic information (information regarding the meaning of the elements) to the parse tree, as noted in the previous section.

At the same time, the semantic analyser populates and pads the symbol table (initially constructed by the lexical analyser) with information on the types and values of each of the variables encountered. This is done by static checking, defined as checking that occurs prior to run-time [4]. Finally, the semantic analyser reports and handles (where possible) any errors encountered during the semantic analysis. The most common error handling strategy is that of static checks [44].

Before continuing, it is important to understand static checking in order to gain insight in the types of errors which may be encountered by the semantic analyser. Static checking consists of type checks, control flow checks, uniqueness checks, and name-related checks. Each of these checks will now be briefly explained.

Type checks, the first type of static checking, ensure that the compiler reports an error if an incompatible operation is applied to an operator, e.g. a string is added to an integer. Control flow checks, a second type of static checking, ensure that statements that force flow of control to leave a particular construct have a target to which flow of control can be transferred. A third type of static checking, uniqueness checks, guarantee that an object is only be defined once.

Finally, name-related checks enforce the rule that a keyword should appear for the requisite number of times, such as for a name that must appear at the beginning and end of a loop in Ada. In such a case, static checking is responsible for ensuring that the same name is used throughout.

Static checking is especially useful for the further work done in Chapter 5 as it assists in the post-event forensic analysis of an XML file. As this chapter is not intended to be a detailed work on the ory of compilers, a more detailed discussion on static checking and the concepts associated to the types of static checks is deemed outside of the scope of this discussion. More information on these topics can be obtained in [4], [46] and [106].

The purpose of static checking is to ensure that certain classes of input errors relating to variables and variable assignments are detected early and reported to the user in such a way that these can be addressed before runtime [4].

Now that all three of the analysis phases have been discussed, it becomes necessary to examine the errors that may be encountered during the analysis phase.

The next section focuses on the types of errors that may occur during the analysis phase and the ways of handling such errors.

## 4.3 Errors and Error Handling

In the previous section the phases of lexical, semantic and syntactic analysis were discussed. In addition, it was briefly mentioned that errors may occur in each of these phases. No compiler could operate correctly if these errors were not handled and as such no discussion on compilers can be complete without a discussion of the various types of errors and the strategies that one can employ to deal with these errors.

This section will therefore focus exclusively on the errors that may occur during compiling as well as the error handling strategies employed by compilers in general. Errors form a very notable foundation in the approach suggested as part of the future work in Chapter 5. These are discussed in the context of the further work in Section 5.3.3.

### 4.3.1 Errors

Due to the importance of errors and their foundational role in the later work, an overview of the types of errors is discussed in this chapter.

Input to a compiler is often flawed — if a compiler was to only process correct input, its design would be much simpler [106]. A good compiler should assist its user in identifying and correcting errors in the input.

Each of the analysis stages discussed in the previous section may encounter errors, resulting in three main error types: lexical errors, syntax errors and semantic errors.

Each of these error types will now briefly be discussed. In order to make the explanation of these errors as relevant as possible, and considering the use of XML data in the later work

presented in Chapter 5, examples of such errors will be given using XML-based input.

The first type of error is a lexical error. It is generally defined as an error that occurs when the lexical analyser cannot tokenise the input and that input is rejected by the lexical analyser [80]. An example of such a lexical error would be the error caused by a misspelled token (identifier, keyword or operator).

Aho, Sethi and Ullman [4] observed that few errors are discernible at the lexical level alone, as a lexical analyser has a very localised input view. An unrecognised token may occur due to a misspelled keyword, or an undeclared variable, and the lexical analyser in isolation has no way of determining which of these caused the error.

Consider an example in order to make the abstract concept of a lexical error more practical. Instead of using a single opening angular bracket ('<'), a double opening angular bracket is used by accident ('<<'). This will result in a lexical error because the '<<' token is not defined.

The second error type which may occur during analysis is that of a 'syntactic error'. A syntactic error occurs when none of the token streams match a rule in the grammar set specified for the input. A syntactic error is usually the result of an error occurring in the syntax of the sequence of characters (tokens). For example, a syntactic error would be caused by a statement without the proper closing tag, such as not closing the opening tag <tag>.

The majority of error detection and recovery occur during the syntax analysis phase, due to the fact that most errors are syntactic in nature and/or are exposed when the token stream does not match any rules specified in the grammar. Also, modern parsers are exceptionally efficient at detecting the presence of syntax errors.

The third and final error type is that of semantic errors. A semantic error occurs when the logic of the program is incorrect and the resulting statement does not make sense. An example of such an error would be using non-corresponding opening and closing tags, such as <tag> and </result>.

Aho, Sethi and Ullman [4] introduce a subtle distinction between logic and semantic errors, with the former being an error that is only realised at runtime, such as an infinite loop, and the latter being errors that are detected at compile time, such as attempting to assign a variable of type string to a variable of type integer. Cooper and Torczon [44] as well as Das [46] fail to make this distinction and simply refer to semantic errors.

Due to the nature of the compiler application in the later work introduced in Chapter 5, and the fact that it excludes synthesis, logic errors will be excluded from the discussion. This is as a result of the compiler not being required to generate output in another target language (the purpose of the compilation is not translation of input into output, but rather only the application of input analysis). Therefore, due to the absence of output, logic errors will typically not be encountered.

That said, logical errors may be possible it may therefore be required to cater for specific logic errors in the creation of a grammar, in order to detect specific data irregularities. This is however left to future work. Please refer to Chapter 8 for this and other potential future work on detecting data irregularities.

Now that the various error types have been defined, the next important aspect of the analysis process can be discussed — that of error handling.

### 4.3.2 Strategies to Handle Errors

Previously, various error types were identified and defined. However, only understanding the error types is not enough — it is very important to also know how to respond when such errors are encountered. Discussing error handling strategies is even more vital when considering that these constitute a major part of the further work in Section 5.3.3, as error handling strategies are key in locating data irregularities.

Error handling strategies are foundational to the correct functioning of a compiler. It is to be expected that input received will not always be correct, and without a proper error handling strategy the compilation process will fail in such an event [44].

In order for compilers to function efficiently, there are a number of important considerations for error handling strategies [4, 44, 80]. Error handlers should:

- report the presence of errors in a manner that is accurate and clear to understand.
- recover quickly and efficiently from each error so that subsequent errors do not remain undetected.
- not significantly slow down the compilation of correct input.

Furthermore, merely reporting that an error occurred would significantly complicate the process of correcting the input. Instead, the compiler should endeavour to report as much information about the nature and location of the error as possible [80]. Specifically, it should report the exact place in the source input where the error is detected, as there is a very real possibility that the actual error occurred within the previous few tokens.

In the author's research various methods were encountered to handle errors [4, 44, 80]. However, there were four types of error handling which were consistently noted by various authors and as a result it was decided to limit the discussion to the following error handling strategies: panic mode, phrase level correction, error production and global correction.

Each of these strategies will now be briefly discussed.

#### **Panic Mode**

Panic mode is the most basic error handling strategy of all and is also most widely used [44]. Panic mode entails discarding the input symbols one by one until a member of a pre-defined set of synchronising tokens is found [138]. Synchronising tokens are tokens whose role in the input are clearly defined, and are usually delimiters (such as a `>` in the case of XML input).

While panic mode correction often discards a considerable amount of input (without checking this input for further errors), it is very simple to implement and is guaranteed not to degenerate into an infinite loop. This method may be used in circumstances where multiple subsequent errors are not expected in the input.

The second type of error handling, phrase level correction, is discussed next.

#### **Phrase-level Correction Mode**

As previously mentioned, error handling strategies are key to the successful operation of a compiler.

The second strategy to handle errors, phrase level error handling, entails performing local corrections on the remaining input when encountering an error [4].

Basically, the parser replaces a prefix of the remaining input by a specific string which allows the parsing process to continue. An example of such a local correction would be the replacement of a `<` with `< /` in XML input to achieve the correct closing of a tag.

Consider the example above. This type of error handling is especially useful. Instead of producing a number of unrelated errors due to unexpected content in the value of the tag and/or the creation of a supposedly new tag (which is actually the closing tag without the preceding forward slash), it detects that the tag is incorrectly closed. By assuming that the tag should be closed, the amount of irrelevant errors (or noise) in the error output is greatly reduced.

Although this error handling mode is extremely useful to correct simple oversights in the input stream, it may lead to an infinite loop if not used correctly. Furthermore, it suffers a major challenge in instances where the actual error occurred before the point at which an error was detected.

This brings us to the third kind of error handling, namely error productions, which is discussed in the next section.

### **Error Productions**

Error productions, another form of handling potential compiler errors encountered during the analysis phase, involve supplementing the grammar specified for the compiler input by including rules that produce erroneous constructs [80]. By using this additional grammar, one can construct another parser that can generate detailed and appropriate error output by indicating exactly which improper construct was used in the input.

This type of error handling, as well as the last type of error handling discussed in the next section, is extremely useful in the detection of data irregularities. Refer to Section 5.3.3 in Chapter 5 for a more detailed discussion on how this error handling strategy can be used to detect data irregularities in XML.

The final error handling strategy, global correction, will be discussed next.

### **Global Correction**

Global correction, the final strategy for handling errors encountered during the analysis phase, involves generating the minimum number of changes required to arrive at a syntactically correct input [153].

Aho, Sethi and Ullman [4] note that this error handling strategy is only of theoretical interest as it is too costly to implement in the real world (in terms of performance).

However, this particular error handling strategy can be very useful in determining the exact error that occurred (the root cause of the compilation errors) and is therefore very well suited towards reconstruction of errors. As a result, this error handling strategy is of key importance in the later work introduced in Chapter 5. Refer to Section 5.3.3 for a detailed discussion on how this error handling strategy can be used to detect data irregularities in XML accounting data.

## **4.4 Conclusion**

The purpose of this chapter was to provide an overview of compiler theory relevant to the later work in Chapter 5 to the reader not familiar with compiler theory. In the introduction of

the chapter, it was argued that is possible to mathematically prove that input complies with a specific grammar, using compiler theory. It was further argued that this technique could be used to assist an investigator by excluding files which match a particular grammar from the investigation.

Furthermore, a basic definition of a compiler was provided, stating that a compiler in its simplest form is a translator responsible for translating input into output of a different specification. It was noted that a compiler operates in two phases — analysis and synthesis. Next, the author argued that a discussion on synthesis is outside of the scope of the chapter as translation to a target language is not required of the compiler used in Chapter 5.

Subsequently, the chapter focused on syntax and grammars, followed by an overview of the three stages of analysis, namely: lexical, syntax and semantic analysis.

The chapter continued by noting that errors are a logical by-product of the analysis phase. As a result, it was important to discuss errors and error handling. It was noted that three main types of errors occur, namely: lexical, syntax and semantic errors (mirroring the phases of analysis).

Finally, the chapter concluded by reviewing the various error handling strategies proposed to deal with the above error types, and noted the importance of errors and error handling strategies to the later work on detecting data irregularities in Chapter 5.



# Chapter 5

## Devising a Method for Detecting Data Irregularities

### 5.1 Introduction

In the Introduction (see Chapter 1), the problem statement was formulated as follows: *XML financial data is susceptible to tampering due to the human-readability property required by the XML data specification. Upon presentation of a set of XML financial data, how can one determine whether data has been tampered with, and reconstruct the past events so that the nature of tampering can be determined?*

This chapter marks the shift from providing necessary background in order to understand the work, to discussing proposed methods to address the problem statement. Specifically, the aim of this chapter is to address the question of whether XML financial data has been tampered with by proposing an automated means to assist in detecting data irregularities amongst a large data set, focusing specifically on XML accounting data.

In addition to the proposal of using an automated means to detect data irregularities, the chapter also investigates the development of such an automated tool. However, the specifics of the tool such as how it should be constructed, is not part of the scope of work discussed.

The work contained in this chapter is significant, as it offers a novel approach to finding data irregularities in XML data files. Furthermore, the observation that data irregularities can be grouped (and are therefore searchable by using a pattern matching technique) is new, as is the concept of employing compilation techniques to check for such data irregularities in XML.

The intention behind the development of an automated tool to detect data irregularities is the augmentation of investigations dealing with fraud in accounting data. This tool is not intended to replace an investigator and take over his or her work. Instead, the focus of such a tool is to assist an investigator in performing his or her duties by flagging potential data irregularities from a mass of data in a time-efficient and rational manner.

The reasoning employed in the use of the automated tool can be explained using an example from forensic pathology. In forensic pathology [89], the apparent cause of death is initially set aside during the post-mortem. The body is carefully dissected in a structured manner and the pathologist constantly compares evidence observed to what is considered “normal” in healthy bodies so that all irregularities can be noted. Using this information, the pathologist finally concludes on the cause of death and can independently confirm how death occurred.

Similarly to the forensic pathology approach of carefully dissecting in a structured manner

and noting inconsistencies, the automated tool will carefully analyse the financial data in a structured manner and constantly compare evidence observed to what is considered “normal” in a tamper-free XML financial data file. Financial accounting rules, structure of the XML and many other factors can be used to establish the normality of the evidence observed. Finally, the tool will then log all the irregularities and present these to the investigator, allowing the investigator to conclude on whether or not the data has been tampered with.

Finding irregularities is a key step in the process of determining whether data has been tampered with. Before a meaningful discussion can be facilitated around the manner in which such an automated tool operates, it is important to first define the concept of a data irregularity, as provided below.

**Definition 5.1.1.** *Data Irregularity: Any direct modifications (i.e. modifications not made by using the designated XML accounting software), to XML accounting data.*

Although file corruptions, formatting irregularities (e.g. white-space inconsistencies) and other syntax-based irregularities are legitimate causes for concern when discussing the data integrity of XML files, these are not the focus of this chapter. As a result, data irregularities such as the ones mentioned above will not be considered as part of the corpus of data irregularities defined at the start of the paragraph.

Consider the following example to illustrate the concept of a data irregularity.

**Example 5.1.1.** *BigCorp (Pty) Ltd uses an accounting package called XMLAccountingSoft to manage their accounting records. Data is stored in an XML file called financialData.xml in the local XMLAccountingSoft directory on the accounting server. XMLAccountingSoft is accessed remotely by Janice, the Financial Manager of BigCorp. She is the only person in BigCorp with access to XMLAccountingSoft.*

*BigCorp is unaware that their system administrator, Ben, has a significant gambling problem and is in urgent need of cash to clear his gambling debt. Despite several attempts, Ben has not been able to compromise Janice’s password and username for the XMLAccountingSoft administrator login as Janice has been diligent in following BigCorp’s security policy.*

*However, Ben has recently discovered another angle of attack. Ben studied XML and became aware of the fact that XML accounting data can be manipulated due to its human readability property. Ben also has remote access to the XMLAccountingSoft server should the need for emergency maintenance ever arise.*

*Waiting until after the month end audit, Ben logs in to the accounting server remotely and opens the financialData.xml file using the default text editor on the server. Ben now modifies the data file by decreasing the stock on hand. Ben intends on selling the stock in order to clear his gambling debt. Ben however forgets to add the equivalent entry to the sales account in order to balance the decrease in stock, saves the file and quits his session. Over a period of several weeks, Ben smuggles out physical stock corresponding to decrease in the stock on hand and sells it for a tidy profit. Meanwhile, BigCorp does not suspect anything (as their stock on hand matches the accounting records perfectly).*

The direct modification of the accounting data, coupled with Ben’s error of not creating a correct double entry, is a prime example of the types of irregularities falling under the definition of the term ‘data irregularity’ as used in this chapter. Depending on the specifics of a modification to XML data, as well as the type of data stored in the XML file, data irregularities

may be detected immediately or, in other cases, may only be detected when the change impacts a system or process dependent on the data that was modified.

In the case where XML is only used to store data which is of little value (such as the language settings for a program's user interface), the existence of data irregularities have a negligible impact. However, in the case where the XML data is of high importance (such as financial data), data irregularities are often malicious and typically point to a criminal offence [135]. In such cases it is of extreme importance to detect these irregularities as early as possible and act on them if need be. This is emphasised by Quinn's [130] research, concluding that the likelihood of solving digital crime is inversely proportional to the time elapsed since the criminal act took place.

Regular checks on sources of important, business-critical data (including computer files) are mandated by legislation as well as due diligence (see Chapter 2, Section 2.4.1). However, performing such regular checks is a difficult task as manual searching through all data irregularities and evaluating the results for correctness and authorisation is a hard and laborious task, similar to the needle-in-a-haystack problem.

The chapter now continues by introducing the analysis of XML files. Also, the chapter investigates the process of finding data irregularities and discusses the use of automated tools to look for data irregularities. The chapter concludes with an analysis of the benefits derived from using a compiler for detecting data irregularities, as well as the application thereof.

The next section introduces the process of analysing XML files for data irregularities.

## 5.2 Analysing XML files

As mentioned earlier, the purpose of this section is to propose an automated method of detecting data irregularities, as defined in Definition 5.1.1. In this section, the groundwork for the development of such an automated tool is laid through analysis of XML file content and expansion of the definition of a data irregularity based on the observations made in this section. Paramount is the conclusion that data irregularities can be grouped by pattern, which proves fundamental in subsequent sections where the automated tool is derived.

Bray et al [26] mention that the XML parser accepts certain character strings whilst ignoring others. Character strings that are ignored typically fall in one of the following two classes:

- Comments: Strings prefaced by '`<!--`' and ended by '`-- >`'; and
- White Space: These characters typically vary from spaces to tab characters, new lines, carriage returns and line feeds, to mention a few.

Considering the above, one is led to conclude that changes to an XML document can be partitioned in two ways:

- changes that impact the semantic meaning, consisting of character strings that are not ignored by the XML parser; and
- changes that do not impact the semantic meaning, consisting of character strings that are ignored by the XML parser.

One is further led to conclude that changes that do not impact the semantic meaning should not be classified as data irregularities as per Definition 5.1.1. This is due to the observation

that such changes do not affect the data contained in the XML file, but only serve to modify formatting. Changes of this nature will therefore be excluded from the definition of a data irregularity. As such the original definition of data irregularities (as per Definition 5.1.1) should be modified to exclude changes that “do not impact the semantic meaning of the XML file”.

**Definition 5.2.1.** *Data Irregularity: Any unauthorised modification to XML accounting data that impact the semantic meaning of the XML accounting data.*

Accounting software typically have several rules built in to ensure that data do not contain errors [12], for example enforcing the double entry system (for each credit, a debit of equal value needs to be passed). Yona [173] notes that these checks are often extended in financial applications to form controls — specific checks designed to enforce oversight and to reduce the likelihood of unauthorised modifications of financial data. Examples include setting up a purchase initiator and approver, which are required to be separate individuals, or enforcing authorisation for expenses in excess of a specific threshold.

Considering the use of controls and automated rules, data irregularities can occur in one of two ways, namely: direct modification of the XML accounting data file, which bypasses all controls and rules in the application responsible for generating the XML data; or passing a transaction from within the software responsible for generating the XML data which passes the controls and rules, but which is still not legitimate.

During a typical investigation, investigators are often faced with looking for small inconsistencies and irregularities in a proverbial haystack of masses of accounting data, the so-called needle-in-a-haystack problem [35]. In such an instance, the investigator suffers from a condition termed information overload, defined by Feather [55] as the point where there is so much information that it is “no longer possible to use it effectively”.

Hardy and Bryman [71] note that grouping information by type and then looking for trends is often used to facilitate data analysis in the case where interpreting massive amounts of data is required. This approach is key in combating information overload and narrowing the search parameters. In the remainder of this section, I will be exploring the idea of grouping information and looking for patterns in the accounting data to assist in the identification of the data irregularities as defined above.

Finding trends or patterns in information is one of the primary investigative tools at the disposal of a criminal investigator [14]. One of the most useful ways of establishing such patterns is abstracting the pattern of events that occur during the incident. Various examples of such patterns exist.

For instance, consider the double entry system used in accounting [165]. For each debit entry in one account, a corresponding credit entry is required in another account in order to ensure that the debit and credit sides of the books are in balance. In an accounting software package, one can assume that general accounting rules, such as the double entry system, will be applied to the generation of XML financial data. Therefore, in the event where this ‘pattern’ of double entries is broken (by either a lone debit or credit entry), one may assume that an entry into the books have been made by a means other than the accounting software package.

Similarly, the motive behind data irregularities can be deduced by abstracting the pattern of changes. The types of data irregularities constituting fraud should differ vastly from irregularities constituting malicious modification by a disgruntled employee. As the purpose of the modifications in the two scenarios vary greatly, so should the actual set of irregularities caused by the modifications.

To illustrate the above, consider the scenario where books are kept in an environment which sees a large number of daily transactions, such as an online gambling shop. Many bets are placed, and some payouts are processed on a daily basis. Consider now the situation where a malicious system administrator puts through an additional double entry for each transaction, removing a small fraction of each transaction and payout and paying all the fractions to a suspense account, which is used for personal expenses. This is called the salami attack [124].

Abstracting the pattern of changes from the above illustration, one will note an account containing a very high number of entries for very small amounts. If an investigator is familiar with the ‘salami attack’ and applies the thought process outlined earlier regarding deducing the motive by abstracting the irregularities, the investigator will be able to deduce that the motive behind the occurrence of the irregularities was that of fraud.

This is where the real impact of the grouping of data irregularities into patterns is revealed. By knowing in advance which types of motives yield which types of data irregularities, an investigator can create rules based on the expected irregularities for the scenario under investigation and automate the search for these. Should one such set of derived data irregularities be found, the investigator will be led to reasonably conclude that an incident of the suspected type may have occurred.

In the previous example of the ‘salami attack’, if an investigator was therefore aware of the possibility of such an attack prior to an investigation, the investigator could create a set of rules that looks for an account containing a high number of transactions for insignificant amounts to detect incidents of a ‘salami attack’. By specifying such patterns of individual data irregularities comprising various incidents (i.e. normal usage, accounting error, data entry error, criminal incident, etc.) one can start to interpret and make sense of the changes that took place in the history of an XML accounting document.

However, it should be noted that the existence of a pattern does not necessarily indicate a crime (correlation does not equal causation). Consider the case of pattern analysis known as trend analysis [6], where time and date stamps of transactions are analysed for patterns. Typically, transactions that occurred outside of business hours will be flagged for review, as most accounting transactions will take place within business hours. However, it may be possible that the accounting system has been down during business hours at month end, and that the accountants need to work late in order to capture all the necessary transactions prior to the start of a new month.

Patterns therefore can be used as an indication of events that require more in depth analysis, but cannot be relied upon in isolation as a litmus test for fraud.

In the next section, the process of detecting data irregularities will be discussed.

### **5.3 Finding Data Irregularities**

In previous sections, the concept of a data irregularity has been well defined and the possibility of the use of patterns and motive to detect possible data irregularities has been established. The next step is that of deriving a practical method of locating such patterns. Patterns are made up of individual data irregularities forming a part of the large data set comprising of all changes made to a specific XML financial data file containing accounting data. In this section the focus will therefore be on the techniques employed to find potential data irregularities, so that these techniques may be automated in order to deliver a tool capable of detecting these irregularities

without human intervention.

The computer virus detection industry is a prime example of a real-world environment where patterns need to be recognised amongst large sets of data [21], and it is therefore useful to leverage its approach to the challenge of the needle-in-a-haystack mentioned in the previous section. Given the similarity of pattern matching in large volumes of data, I will briefly discuss the approach taken by anti-virus engines to identify data irregularities in large data sets and then apply this approach to the location of patterns of data irregularities in XML accounting data.

### 5.3.1 The Large Data Set Approach Used by Anti-virus Engines to Identify Data Irregularities

Bidgoli [21] concludes that there are two main virus-scanning approaches employed by anti-virus engines: signature-based approaches and heuristic-based approaches.

Signature-based scanning works on the idea that each virus can be identified by a particular identification string, called its signature [21]. By creating a database of all known virus signatures, targeted computer files can then be inspected for viruses matching any of the collection of signatures contained in the virus database. If no match is found, the anti-virus engine will pronounce the target as not infected.

Of the two approaches, signature-based scanning is the most commonly used and is known for its accuracy and speed. However, the accuracy and speed comes at a price as this approach can only identify viruses for which a signature exists in its virus database. Typically, only viruses that are known and have been seen before can be detected. Should the target be infected with a novel virus or even a previously unseen variation of a known virus, the anti-virus engine would not flag the target as being infected. Even minor variations to a virus will cause it to not be detected by the traditional signature-based approach.

Additionally, signature-based scanning may yield false positives [20], meaning it may flag a file as containing a virus when it does not. This is due to the fact that only a small portion of a scanned file's code (or fingerprint) is used to match against the database. Naturally, as with hash algorithms [142], collisions do occur, which results in a false positive. However, as with hash-collisions, false positives in signature-based scanning occur rarely.

Heuristic scanning, the second approach, is also widely used in anti-virus engines. This approach is more generic in nature than signature-based scanning, as it is based on a generic, pre-defined set of rules (rather than looking for unique signatures) [21]. Heuristic scanners look for common characteristics of viruses, which have either been identified by experts or observed from machine learning. By examining each target file, the anti-virus engine makes an educated guess as to whether the target deviates from the norm and shows signs of malicious behaviour as predicted in its rule set.

Though heuristic-based scanning has a better success rate in detecting viruses than signature-based scanning, it may generate false positive or false negative results as well [21]. As a result, an anti-virus engine using only heuristics may identify a harmless program as being a virus. Also, the rate of false negatives or positives in heuristic-based scanning is much higher than with signature-based scanning, and it is generally not effective when used on its own. Furthermore, heuristic-based scanning is also much more processor intensive and significantly slower than a signature-based approach.

Given the drawbacks above, one concludes that neither of the above approaches is completely effective when used in isolation, as both suffer from significant shortcomings. Therefore, most anti-virus vendors (including prominent names such as Kaspersky and BitDefender [82, 156]) use a combination of these approaches, basing their results on a combination of the result sets generated by each of these approaches.

The same approach holds when searching for patterns of data irregularities. One can either look for each pattern of data irregularities (strict pattern matching) in a similar manner as signature-based scanning, or look for deviations from the norm in the same way as heuristic-based scanning. As with anti-virus vendors, the best solution is to use a combined approach whereby both approaches are employed simultaneously, as neither can be truly effective on its own.

Examples of such patterns of data irregularities may include regular transactions made outside of business hours, transactions made during periods where the staff member making the transactions is on leave or even transactions where the transaction ID numbers do not consecutively follow each other. For a concrete discussion of such patterns and a classification of such pattern examples, please refer to Table 5.1.

Now that a real-world approach to pattern matching in large data sets has been established, it is important to consider how such a search could be executed and therefore the process of executing such a search will be discussed next.

### 5.3.2 Using Automated Search Techniques to Locate Data Irregularities

Previously, a real-world approach to pattern matching in large data sets has been established. Subsequently, considering how such a search could be executed is of importance. Therefore, this subsection will focus on the process of making the case of using automated methods for executing the search, as well as discussing how such an execution would look.

Data irregularities may be found using either manual or fully automated searching.

Manual searching is typically performed with the assistance of pattern search tools (such as `grep`, a popular UNIX search tool) allowing the user to search for character strings in the target data. Not only is such a technique excessively time-consuming and prone to error, it also relies on the thoroughness of the individual performing the search. As humans tend to be bored by repetitive work [132], these searches are prone to mistakes and typically have a lower rate of success when matched to fully automated searching.

Fully automated searching, on the other hand, is typically conducted by the use of programmatic means and therefore tends to be much more accurate and time-efficient than manual searches. As can be deduced from the above, utilising an automated searching technique has distinct advantages, especially when coupled with a hybrid search approach using both baselining and specific pattern search.

In automating the process of testing for data irregularities, one needs a tool that is able to follow the pattern of reasoning described above. This tool should not only be able to recognise patterns, but also classify them and match them to a rule set created for the data file that is under investigation. Finally, the tool should be able to take unique action based on whether input matches patterns or not, whether patterns match rules or not and whether the input can be scanned or not.

The process of automatically searching through a set of data for certain patterns, matching them to rules and then taking specific actions based on the results forms the parameters to a

problem that is by no means unique. This specific problem is well known in the domain of compiler construction and compilers have been developed as a solution to this particular challenge [44]. In compiler construction, it is necessary to look for tokens (patterns of characters) and to take certain actions when finding these tokens. It is therefore logical to conclude that the automation of the search for data irregularities should be accomplished by utilising the same techniques as those used for constructing a compiler.

As mentioned in Chapter 4, a compiler is defined as “a program that reads a program written in one language - the source language- and translates it to an equivalent program in another language - the target language” [4]. The compiler searches for patterns or tokens during a process called parsing, which are then compared with a set of rules. Based upon this comparison, certain actions are then taken.

Additionally, compilers excel at recognising patterns and acting on the recognised patterns based on a set of pre-defined rules [4]. Not only is creating a compiler an established method for creating a tool for automated pattern searching, it also establishes a rich environment which can be customised very easily. This is due to the various demands placed upon compilers and is necessitated by the complexity of the compilation process.

As already mentioned in Section 4.1, the automated search for patterns only makes use of the analysis stage of the compiler. However, compilers are not only supposed to analyse input, but are ultimately used to create output in the form of code generation [4]. For the purposes of automating the search for data irregularities, such functionality is not required, and reference is only made to the parsing and error handling capabilities of the compiler, as opposed to the traditional definition of a compiler which includes both the analysis and code generation.

It should therefore be noted that whenever the term compiler is used throughout the remainder of this chapter, the definition as below is intended.

**Definition 5.3.1.** *Compiler: An automated parsing and analysis tool used to detect data irregularities (see Definition 5.2.1) by comparing input to a rule set and producing errors if the input does not match the rule set.*

Constructing a compiler to address the automated search component for the detection of data irregularities in XML enables: [36]:

- the classification of input, based on patterns as well as pre-defined rule sets; and
- the identification of patterns recursively, based on a decision tree.

This is exactly the flexibility and feature set that is required to successfully automate the search for data irregularities in XML. Not only does this approach cater for strict pattern matching, it also caters for the detection of deviations from the norm, and can easily be used to alert the data inspector in the case of data irregularities being detected.

In *Compilers: Principles, Techniques and Tools*, Aho et al [4] observes that the process of compiling is composed of two parts — analysis and synthesis. The analysis phase breaks up an input stream into its fundamental building blocks, by using a pre-defined grammatical structure; whereas the synthesis phase constructs the target program (typically in assembly language) from the symbol table.

As mentioned previously, the automated tool for detecting data irregularities is not required to perform any logical processing on the data irregularities detected and constructing the target program holds little value for the detection of data irregularities. The synthesis phase therefore



is not included in the scope of this discussion. The interested reader is referred to Aho, Sethi and Ullman [4] for a detailed discussion on this topic.

Applying the analysis phase to the detection of data irregularities, one observes that it is the pattern-recognition part of the analysis phase which will be used to recognise patterns of data irregularities.

Should the input be ill-formed or syntactically incorrect, the analysis process halts. The compiler is now expected to deal with this unexpected occurrence, and, as was explained in detail in section 4.3.2, the compiler is forced to employ an error handling strategy to handle the error. This enables the data inspector to utilise pre-defined behaviour which should be applied in the case where a data irregularity occurs that does not match the pre-defined rule set.

In the next section, error handling strategies and the use of such strategies to classify irregularities will be discussed.

### 5.3.3 Utilising Error Handling Strategies to Locate Data Irregularities

In the above section, the basis from which the tool will be developed was established. This was done by drawing a parallel between the process of constructing a compiler and the process of constructing a tool to locate data irregularities. One of the key areas of compiler construction is determining how errors should be handled. When used correctly, error types can be invaluable in determining the type of irregularity and how such irregularities should be treated. This section will focus on using error handling strategies to further classify irregularities and ease the burden of the investigator.

Previously, in Section 4.3.2, the compiler error handling strategies and associated error types have been listed and defined. This chapter expands on these errors by illustrating their usage in detecting data irregularities.

As previously mentioned in Section 4.3.2, the following compiler errors can occur:

- Panic mode;
- Phrase-level detection;
- Error production mode; and
- Global correction.

Panic mode can be used to display error messages or warnings to the data inspector. This particular mode does not really have any interpretative value for the data inspector, and simply alerts the data inspector to a data irregularity that does not match any rule and is therefore an outlier. Therefore, panic mode will only be useful in a small minority of cases, as the data inspector will rarely require a mere list of possible data irregularities. In the ideal case, one should endeavour to provide more analysis and background information in order to assist the data inspector to make sense of the output from the compiler.

The second error handling strategy, phrase-level correction, has far greater potential in assisting the data inspector to make sense of the compiler output than panic mode. This principle is best explained by way of an example: Let us say that panic mode lists a great number of missing accounting entries. Which is more likely — that some accounting entity is missing (or not declared) or that the accounting data contains a massive number of errors? When applying logic to the situation, it is fairly obvious that the former is the more likely scenario.

The phrase-level correction error handling strategy is well-suited to minimising system noise, such as assisting the data inspector to conclude that a tag has not been closed correctly in Example 5.1.1. Using this strategy enables the data inspector to make further sense of the data. Consider the analogy of a graph. When faced with a graph that has a data point which is blocking a curve fitting, phrase-level error correction allows one to safely remove the outlying data point under certain conditions (as well as the noise associated with this data point), in order to see whether the curve would fit in the absence of the data point.

The third error handling strategy, error production mode, is very useful in detecting data irregularities. In using this strategy, the data inspector may opt to explicitly add specific patterns of data irregularities to the rule set containing the search criteria. In doing so, the data inspector enables the compiler to effectively and quickly point out whether certain aberrations exist within a given set of data input or not.

The purpose of such an exercise can be explained by the following example: say that the data inspector is aware that a particular type of fraud is prevalent in the department whose XML accounting data is currently under investigation. Although the data inspector wants to ensure that the XML accounting data is free of data irregularities, the data inspector is also most interested in knowing right away whether or not the particular type of fraud typical to this department is present. The data inspector can now create custom rules containing generic patterns of this particular type of fraud in the rule set used for pattern matching. When the compiler is then executed on the department under investigation's XML data, the inspector will be able to deduce whether or not the particular type of fraud specified in the rules is in fact present in the accounting data.

The final error handling strategy, 'minimum change' has significant potential in identifying data irregularities. Although this strategy is computationally very expensive and difficult to implement, it has the potential to indicate to a data inspector which change(s) were made resulting in the data irregularities and assist the data inspector in isolating particular instances of data irregularities.

The minimum change strategy's ability to indicate the shortest path to a correct data set (i.e. a correct transaction set when referring to XML accounting data) can therefore be very useful under specific hypotheses. Refer to Chapter 6 for a more detailed discussion of the process of hypothesis forming and testing.

Consider the example where a third party abused his or her privileges and made a single unauthorised change to the data. In this instance, the minimum change error handling strategy will be able to find the offending original transaction (causing the deviation and subsequent errors), and remove its logical effects. Another example would be the case where a third party repeatedly abused privileges and repeatedly modified the data. In this instance the minimum change strategy can assist by minimising the deviations and removing the transaction trees (where the offending transaction is seen as a root node with its subsequent transactions as its children).

The minimum change strategy therefore becomes an important tool at the disposal of the data inspector which can allow him or her to make better sense of the data when he or she requires it. The output of the tool can greatly assist the data inspector, especially in the scenario where initial panic mode output is available and the investigator seeks to minimise noise. This allows the data inspector to make a copy of the XML data (in a forensically safe manner), make the changes to it as suggested by the output of the minimum change error handling strategy, and re-run the compiler on the modified data in panic mode. This should enable the data inspector

to deduce whether all compiler errors were in fact caused by the transaction(s) pointed out by the minimum change strategy output. Refer to Sections 5.3.4 and 5.4 for a detailed discussion on using the error output to make sense of potential data irregularities.

One can therefore summarise the process (as applicable to the detection of data irregularities) in two steps: 1) analysis, and 2) error handling (if applicable). As mentioned at the start of this section, the synthesis phase is not of interest and is therefore ignored.

In the next section, the knowledge gathered above is applied.

### 5.3.4 Applying the Compiler to Locate Data Irregularities

As mentioned in the introduction of this chapter, using an automated tool to detect data irregularities has the potential to ease the investigative burden tremendously. In the above section, the techniques of compilation and specifically error handling were used to aid the creation of such an automated tool. One is now faced with an important question: *how should such a tool be used?* This section intends to investigate how the investigator should apply the compiler to detect data irregularities and aid him or her in the completion of the investigation.

Before continuing with the application of the compiler, Figure 5.1 below provides an overview of the process established thus far.

The first (and most important) step for the investigator is to establish a rule set, customised for the specific data that one wishes the compiler to be executed on. As mentioned in Chapter 4, this rule set is known as a ‘grammar’ and forms the foundational component of a compiler. Rules may be of one of the following two types, and rules from both types should be included in the rule set:

- normal transactions (i.e. normal data for which no errors will be noted, as these transactions are not irregular); and
- error productions, namely patterns of transactions that deviate from the norm (i.e. data irregularities).

The second step of utilising an automated search tool is to execute the compiler. The compiler now tests for input that matches ‘normal patterns’ and discards these. Should the process finish whilst this step is in operation, the compiler will report that no data irregularities could be found.

During step three, the data inspector is alerted. This step is triggered in the case where any error productions (specific, pre-defined irregularities) were encountered, or where any data that did not fit the rules was encountered.

Finally, the data that resulted in an error is flagged and highlighted so that the data inspector can scrutinise it. At this point the work of the data inspector starts. The data inspector can now start to interpret the compiler output.

It should however be emphasised that the output from the compiler does not suggest the use of a binary approach whereby one can state with absolute certainty that there are concerns with the data or not. Instead, the compiler should be seen as another tool in the toolkit of the data inspector.

Such a tool may point in the direction of a final conclusion, much like the use of fingerprint detection in a crime scene. However, the absence of fingerprints does not lead one to conclude that no crime has occurred, and neither does the presence of fingerprints necessary lead one to

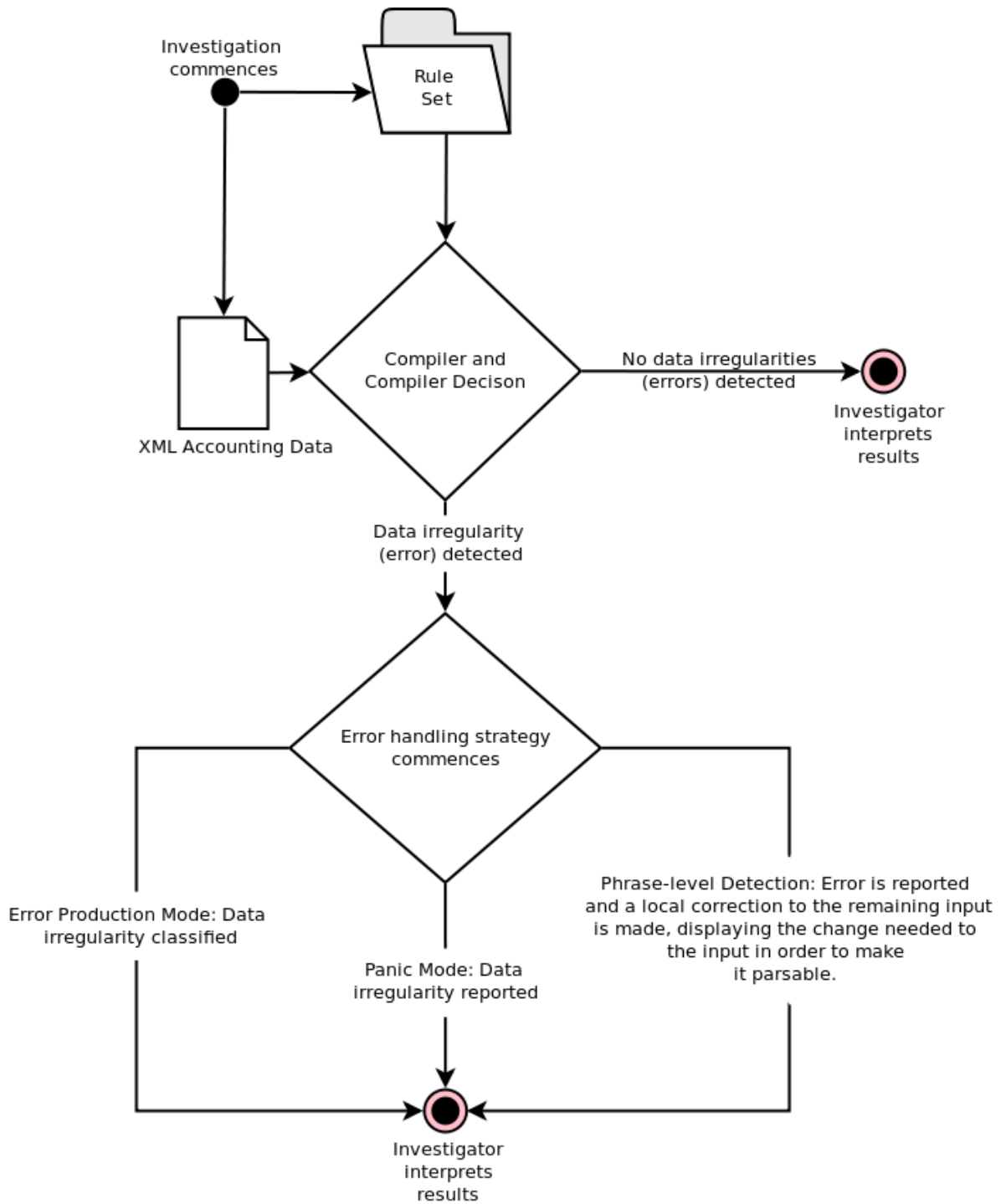


Figure 5.1: An overview of the process of detecting data irregularities using a compiler.

the criminal. It is merely a tool used by an investigator as part of a process of analysing the crime scene.

In the same way, the compiler output is a tool to be used as part of analysing the XML data for irregularities. The automated data irregularity detector is not meant as a silver bullet. It should rather be seen as an aid to the data inspector, assisting him or her to flag which data files

are most likely in need of further investigation. The purpose of this tool is merely to ease the workload of the data inspector (or forensic examiner, whichever the case may be) and to allow the inspector to easily determine where he or she should allocate time and effort, two resources which are in very short supply during any investigation.

Finally, this tool is only as good as the rule set specified for it. In order to derive real benefit from the automated data irregularity detector, the rule set should be as comprehensive and expanded as possible, whilst at the same time highly customised for the specific XML accounting data file that it will receive as input. It is therefore recommended that significant effort should be spent on deriving and creating a comprehensive rule set for use with this tool. This tool's success hinges fully on the quality of the rule set backing it up.

In using a compiler as an automated search tool to look for patterns of data irregularities in the XML accounting data, the data inspector has achieved two goals. Firstly, he or she saved a great deal of time, as the automated data irregularity detection is done much faster than a 'manual' search could have been performed. Secondly, the irregularity detection was done in an efficient and accurate manner, ensuring that deviations from the rule set will be highlighted (in the case of a full compiler run, where the execution was not interrupted by data that could not be compiled).

The next section discusses the practical application of the above methodology as well as the benefits derived by employing it.

## **5.4 Application of and Benefits Derived from the Use of a Compiler for Detecting Data Irregularities**

In the previous section, an automated tool for detecting data irregularities was proposed. Before the tool can be applied, however, it is important to look at how such an application would look in real life. More importantly, it should be determined whether real-world irregularities can be detected by this approach. In this section, the focus is placed on the use of a compiler to detect data irregularities, specifically, investigating which irregularities can be detected using this approach, and why these will be detected.

As the reader may recall from Chapter 3, XML accounting data possesses three characteristics that makes it particularly well-suited for detecting data irregularities. As these characteristics are key to the evaluation of the compiler's real-world irregularity detection abilities, and provide numerous hints as to which data irregularities should be detected, its three core characteristics will be briefly reviewed.

Firstly, XML accounting data has a rigid, predictable structure. Chances are that when the XML accounting data is modified outside of the software that generates it, the rigid and predictable structure may be disturbed, e.g. a tag may be left out or not closed properly, a tag may be misspelled, tags may be in the wrong order, etc.

Secondly, XML accounting data stores plenty of metadata and XML accounting data transactions are typically in date or sequence number order. A disturbance in this order may very well indicate the presence of a data irregularity.

Finally, XML accounting data consists of numerous XML-related cross-checks and check-sums. Furthermore, due to the nature of accounting data and its rules, several accounting-level cross-checks exist as well, such as the double entry system. The original purpose behind these

cross-checks was to detect accounting errors [165], but these same cross-checks also serve to highlight potential data irregularities.

Let us now continue to discuss the types of irregularities that can be detected by using compilers as per Definition 5.3.1. Many different XML accounting formats are available. For the sake of simplicity, a snippet of XML data in a generic format with normalised data will be used to illustrate the basic concepts utilised by most of these accounting formats. Refer to 5.4.1 for such a snippet of XML data. An XML data format, such as used in the provided snippet, would usually require a data type definition (DTD). However, for the sake of the example a DTD would not be required and is therefore not included.

**Example 5.4.1.** *Generic XML accounting data format:*

```
<Transaction>
  <ID> 101-1 </ID>
  <Account> Bank </Account>
  <Action> Credit </Action>
  <Amount> 25000 </Amount>
  <User> 012437 </User>
  <Date> 6/19/2011 8:25:02 AM </Date>
  <Hash> 1a88f9a8293e88c87ae1ae5f8bd63585 </Hash>
</Transaction>
```

Other types of entries (such as account balance entries) in this generic format have the same basic structure (i.e. ID, Account, Amount (if necessary), Date and Hash) with specific fields as required (i.e. a balance field for account balance entries). Other accounting views (such as Journal Views, Account Views and Balance Statements) are available and are generated from a composition of the basic transaction data.

As mentioned in Section 4.3 of Chapter 4, there are three main error types that are applicable to the process of detecting data irregularities, namely: Lexical, Syntactic and Semantic errors. In view of these errors, table 5.1 is presented, illustrating the comparison of the XML components and the data contained in the XML tags with the error types. This comparison (or juxtaposition) can then be used to classify different examples of data irregularities that could occur. Drawing up such a table assists in understanding how the different error types can be used to detect various data irregularities occurring in either the XML tags or the source data.

Table 5.1: A tabular illustration of the juxtaposition of XML tags and compiler error types.

	<b>Lexical</b>	<b>Syntactic</b>	<b>Semantic</b>
<b>XML Data</b>	1.1. Tag not opened or closed correctly, e.g. a missing '<' or '>'.	2.1. The XML schema is violated.	3.1. Tag is not correctly specified to match the content described by the tag, for example the tag attribute incorrectly specifies 24 hour time whilst the time is specified in AM/PM '<'Time format="HH:mm">' 12:30 PM '<'/Date'>'.
	1.2. Amounts that contain non-numeric characters.	2.2. A transaction entry has a missing or imbalanced tag for: <ul style="list-style-type: none"> <li>• Transaction or Balance</li> <li>• Hash</li> <li>• User</li> <li>• Date</li> <li>• Amount</li> <li>• Account</li> <li>• Etc.</li> </ul>	
	1.3. Reserved characters (< or >) used in transaction statement.	2.3. Tags that are not defined, e.g. a tag containing a spelling error on the tag name.	
		2.4. An entry matches one or more predefined rules specifying an incorrect transaction.	

*Continued on next page*

Table 5.1 – *Continued from previous page*

	Lexical	Syntactic	Semantic
<b>Data Errors (Errors in the Accounting Data)</b>	1.4. Irregularities are found in the formatting of the data, introduced by editing the machine generated data, e.g. numbers within tags are given a comma to indicate thousands, but the comma is omitted in certain numbers.	2.5. A violation of the hierarchical structure and/or order of the tags, e.g. an ID tag that exists in isolation (instead of belonging to a parent tag, such as a transaction), or a transaction tag without children.	3.2. Omission of part of a transaction, e.g. a transaction with a missing corresponding double entry.
	1.5. The data contained within XML tags is bad, e.g. a ‘;’ or ‘@’ character occurs in a number, or date with the month specified as larger than 12.	2.6. The allocation of optional tags that is not applicable to the tag object, e.g. listing a vehicle asset number together with a transaction for the purchase of furniture, instead of listing the asset details of the office for which the furniture was actually purchased.	3.3. Transaction ID Errors: <ul style="list-style-type: none"> <li>• ID skipped</li> <li>• ID repeated</li> </ul>
			3.4. Violation of transaction logic, e.g. purchase fulfilment comes before order.

From the above table it is clear that the compiler error types cover a wide variety of possible data irregularities. Additionally, there are definite advantages to using the error types together with knowledge about the XML content (both data and tags) in detecting data irregularities.

Furthermore, one can also add knowledge regarding error handling strategies to the above. Lexical errors will generally be handled by using the panic mode strategy, as these are severe enough to halt further processing of the XML data file and typically point to a breakdown in the generation of the XML data.

Syntactic errors, on the other hand, can generally be resolved using either the panic mode or the Phrase-level error correction strategy of error handling. Syntactic errors are also uniquely positioned to utilise the error production strategy of error handling. Error productions realise the full definition of a syntactic error and are therefore particularly useful in resolving syntactic errors.

Lastly, semantic errors can, by definition, not be handled using error productions (due to the fact that semantic errors are not errors or mistakes in the rules of the language itself). Instead, semantic errors are typically due to modification of the XML data as it implies that the meaning



of the XML data is incorrect. Typically, XML financial data generated by accounting software should be free from semantic errors. This is due to the fact that semantic meaning is considered during the creation of XML accounting data and that semantic logic rules are applied to transactions within the accounting software. Therefore it is considered highly unlikely that accounting software will generate XML data that is semantically incorrect (unless of course, the software has a specific bug).

Semantic errors therefore allow the data inspector to start hypothesising about the event leading to the data irregularity, based on the fact that the expected pattern of the XML accounting data has been violated.

Many methods exist for such analysis. Firstly, on a pure XML data level, one may typically look for instances where opening and closing tags do not match up, indicating that modification took place. It is highly unlikely that accounting software will generate data with mismatching tags, therefore it is considered likely that the data has been tampered with.

Secondly, on an accounting data level, various forms of trend analysis are typically applied to assist in locating semantic errors. Examples include:

- Considering the order of transactions — in a business operating on a cash-on-delivery basis, it is unlikely that payment would be received for goods prior to goods being delivered.
- Analysing the statistical occurrence of numbers — Benford's law is a type of digital analysis based on the statistical occurrence of digits and the distance between them. As a full discussion of this theory falls outside of the scope of this discussion, the interested reader is strongly recommended to read [50].
- Analysing time trends — By analysing the typical time frame between which transactions are posted, it may be useful to review transactions posted outside of this period as these may not be regular business transactions.

Semantic errors, therefore, is typically the result of unauthorised modification of data and checking for such errors is very useful in determining whether data irregularities occurred in a specific data set.

From Table 5.1, it is clear that several types of data irregularities can be detected using a compiler. These can easily be grouped [134], forming several classes or levels of data irregularities, ranging from the obvious and trivial, such as simple data modification to the more complex, such as modification of business processes. This classification is illustrated in the form of a pyramid in Figure 5.2, ranging from the simple and common at the bottom of the pyramid, to the less frequent and more complex at the top.

Different classes of data irregularities can be detected, such as: data modification, falsification of documents, business transaction errors and other data irregularities. Each of these will now be briefly explained and it will be illustrated how the compiler will detect each.

The first class of data irregularities that can be detected, data modifications, are irregularities that imply the possible occurrence of an XML tag related error that will be registered as a lexical error by the compiler.

Secondly, falsification of documents implies a possible error in the sequence numbers of the document upsetting the sequence of dates in the transaction data or another logical error that will be registered as a semantic error by the compiler.

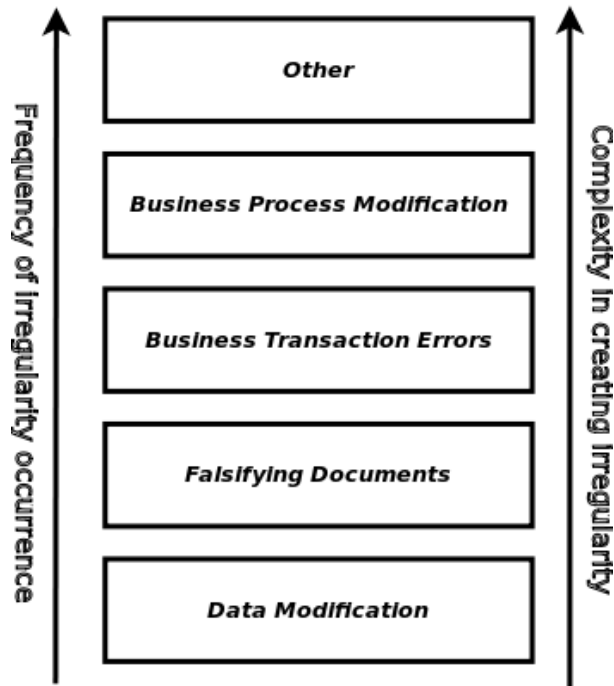


Figure 5.2: The classification levels of the data irregularities that can be detected using a compiler.

A third class of data irregularity is that of business transaction errors. A business transaction that is not reported or reported in duplicate will be detected by using the sequence numbers and logic regarding the ordering of transactions, which will trigger a semantic error in the compiler.

Fourthly, business process modification is yet another type of data irregularity. Such transactions could either violate the transaction order (triggering a semantic error) or could match a pre-defined error production for fraudulent transactions (triggering a syntactic error).

Finally, other data irregularities may be detected by statistical methods, such as Benford's law. Benford's law (also known as the first-digit law), applies to most large sources of numerical data and refers to the frequency distribution of the first digit of such data. In summary, Benford's law concludes that digits starting with a '1' should occur around 30% of the time, whilst larger digits occur in that position less frequently.

Detection methods using statistical analysis as will likely detect the deviations in caused by data modifications, and can be used to trigger a semantic error. This idea is discussed in greater detail in Section 8.4 on future work, forming part of the concluding chapter of this dissertation.

In Section 1.3.2, the problem statement mentioned not only determining whether data has been tampered with but also reconstruction of the events.

As can be seen in the above discussion, there are several facets of information available when applying the automated tool to detect data irregularities. However, these still do not provide the investigator with a coherent sequence of events which can be used in reconstructing the events. This next step in making sense of the compiler output uses the concept of creating hypotheses and testing and verification of such hypotheses. Using this strategy, the data inspector can start to reconstruct the events and interpret the results in a more meaningful fashion.

The scientific approach, including hypotheses formulation and testing, will be discussed in

the next chapter, Chapter 6. These observations will assist the investigator in further interpretation of the results and validation of theories about the events that were responsible for the data irregularities.

## 5.5 Conclusion

Normally, in most applications of XML, tampering would be of very little consequence. However, XML is increasingly used to mark up important information, such as business critical accounting information. In such cases, tampering is often malicious in nature and is viewed as a serious offence.

The only way to protect such important XML files from tampering is to regularly inspect them to ensure that they are free from data irregularities caused by tampering. Searching for data irregularities in an XML data file however, is exceedingly time-consuming and laborious.

The chapter proceeded by offering a solution for this problem in the derivation of an automated tool that can be used to search for such data irregularities (modified accounting data) in an XML file.

It was noted that changes to the XML data can be categorised in two ways — changes that impact meaning (semantic changes) or changes that do not impact meaning. Additionally, it was also remarked that not all data modifications are irregular — some changes (such as typographical errors or accounting errors) occur during normal use of accounting data and should therefore not be classified as data irregularities. Therefore it was concluded that, when looking for data irregularities, only semantic changes were of importance. Furthermore, it was concluded that true data irregularities only consist of changes that cannot be explained by normal use.

The chapter continued by observing that a particularly effective way of dealing with the general information overload is that of grouping information into trends or patterns. Applying this logic to the XML accounting data, it was noted that one can group unauthorised modifications into incident types. This is done by looking in detail at the specific individual changes constituting each incident, thereby specifying a pattern of data irregularities for each incident type. Should one therefore match a pattern of specific data irregularities, chances are that an incident warranting further investigation has occurred.

This makes the job of retrospectively establishing whether tampering has occurred much easier. One can even further develop this principle by taking specific incidents that are high-risk, and reverse-engineering these to find the data irregularities that would lead to such incidents.

Subsequently, the following question was answered: how can one find such data irregularities, using the principle of grouping these data irregularities in patterns? It was concluded that the hybrid approach of combining strict pattern matching, as well as identifying data that deviates from the norm, was the best approach to finding data irregularities.

Next, it was concluded that the best method of implementing such an approach would be that of automated searching. The motivation behind this approach is the fact that manual searching has been found to be far too laborious, time-consuming and error-prone.

Additionally, it was observed that the problem of automatically recognising patterns and acting on these patterns is well known. This specific problem is encountered and solved in a very elegant fashion in the discipline of compiler construction. This led to the proposal of a

basic compiler (or semantic analyser) as a solution to the problem of finding data irregularities in XML files.

Using a compiler as an automated search mechanism for the identification of data irregularities poses several advantages:

- The investigation time is drastically shortened, as the data inspector does not have to inspect the entire XML file manually.
- The investigator is given a detailed indication of whether or not the XML file at hand requires further investigation.
- There is very little chance of data irregularities not being detected. This is due to the automation of the process, eliminating the error-prone approach of human inspection of the XML data.

However, it should be clearly noted that an automated search mechanism does not negate the need for a data inspector. The automated search does not replace the manual inspection, but simply serves to assist the inspector with the process. The automated tool is only intended to give the inspector a time-efficient indication of whether further inspection of the XML data is required.

Furthermore, the success of the tool hinges completely on the quality of the rule set that was specified for it. It is therefore of great importance that the rule set should be as comprehensive and expanded as possible, whilst remaining highly customised for the specific XML data that it will be executed on.

# Chapter 6

## Moving from Evidence to Hypothesis

### 6.1 Introduction

The problem statement (see Section 1.3.2), posed the question of determining whether data has been tampered with and how to reconstruct such tampering. In this chapter, the basis for reconstruction of tampering events is established.

Building on the overview of digital forensics and digital forensic techniques in Chapter 2, this chapter focuses on establishing the criteria for scientific evidence within digital forensics. In doing so, an analysis of the scientific process is performed and the application of this process is investigated. The notion of a hypothesis is defined, and the process of validating and testing such hypotheses is discussed.

The concept of a hypothesis is then applied to the problem statement, whereby hypotheses formulation and testing are used as a vehicle to assist in reconstructing the events leading to data tampering. Finally, the chapter concludes by drawing attention to the need for evidentiary facts and notes the need for instrumentation. Instrumentation is defined as a measuring instrument [117], and will be applied to assist in recording evidence which is required to test the hypotheses formed in this chapter.

Digital forensic evidence is often perceived as evidence which is beyond reasonable doubt by courts and the general public [140]. However, digital forensic evidence is only as good as the rigour of the scientific process (if any) followed in extracting such evidence.

The purpose of this chapter is to establish the need for scientific approaches in digital forensics. Also, the chapter intends to provide a perspective on potential means for solidifying the scientific approach of digital forensics.

The main contributions of this chapter are:

1. Examining the scientific methods used in digital forensics in order to gain an understanding of the requirements of such methods.
2. Examining whether evidence will be applicable by means of applying the standards of admissibility.
3. Using the scientific method as a criterion for admissibility of evidence and applying the scientific method so that a set of hypotheses are established to aid in reconstructing the events leading to the tampered data.

4. Deriving further work using these hypotheses, by interpreting the evidence required and the application of scientific method to digital forensic evidence.

In essence, this chapter considers the definition of the term ‘science’ and questions whether digital forensics really conforms to scientific principles. As part of the discussion in this chapter, several criteria for scientific methods are derived, and a common set of scientific principles for application to digital forensics is distilled from these.

Next, the concept of a hypothesis is introduced, and sample hypotheses are presented. These are used to illustrate how hypotheses may be used to reconstruct the events leading to tampering, the detection of which is made possible by the resulting information from the automated tool proposed in Chapter 5. These hypotheses can also be leveraged to obtain deeper understanding of the evidence required to test such hypotheses, and the instrumentation required to capture such evidence. In turn, this leads to the foundation of the XML Accounting Trail Model, derived in the next chapter, Chapter 7.

Olivier [115] notes that the field of forensics “employs science in determining facts (where such facts are necessary to resolve and settle disputes) or determining the cause of an event”.

Saks and Faigman [140] further point out that the general public and courts generally perceive the various fields of forensic science as a single entity (a black box, if you will) which in its sum total is bound by the laws of science and is the result of the flawless application of such scientific principles. In short, evidence generated by any process termed ‘forensic’ in nature, is generally considered to be proven beyond reasonable doubt.

This belief however is in stark contrast to reality — there are several fields in forensics that can be termed ‘non-science forensic sciences’ [140]. Such fields, according to Saks and Faigman [140] “neither borrow from established science nor systematically test their hypotheses”. Saks and Faigman further continue “whereas in most scientific fields experience and observation are designated as the first steps of the scientific method, for many forensic fields they constitute the final stages of confirmation.”

Often, the ‘science’ of such fields is assumed true as long as their hypotheses and suppositions have not been tested. In turn, these hypotheses are deemed proven if they endure over time. Instead, as Karl Popper [128] notes, hypotheses should be rigorously tested and subjected to falsification.

Cole [42] concurs in noting that “the word and accoutrements of science were exploited to sell these fields to courts and the public”.

Several cases where the notion of ‘infallible forensic science’ has led to the severe miscarriage of justice have been published in the media in recent years [79, 41, 27].

Cohen reflects on one such story that took place in 2004 [157, 144, 41]. Brandon Mayfield, a lawyer, was arrested as a material witness by the FBI. He was held without trial for two weeks as authorities linked him to the terrorist train bombings in Madrid that took place earlier in 2004. The FBI used a fingerprint (collected from the crime scene) as their primary evidence to arrest Mayfield, stating that the fingerprint proved that Mayfield was present at the scene of the crime. During the court proceedings, however, it was proven that Mayfield had no connection with the bombings and that he had not left US soil for several years prior to the Madrid bombing. To make matters worse, the Spanish investigators then revealed that the collected fingerprints belonged to an Algerian citizen and not Mayfield.

In this case, the scientific rigour underpinning the use of forensic fingerprint evidence is open to question. The hypothesis that no two persons have the same fingerprint seems to

not have been sufficiently tested and evidence presented to the court made no mention of the potential error rate of the methods used.

In another story, published in the Sunshine Coast Daily [27] (an Australian newspaper), it is reported that Graham Stafford spent 15 years in prison after being convicted of murdering a 12 year old girl. He was found guilty, primarily based on blood and hair evidence, which turned out to have been contaminated in the laboratory investigating the evidence. His conviction has since been overturned and he has been released from prison.

In this particular case, the science behind the forensic method was sound, but the execution of the scientific process was flawed as it allowed the contamination of the evidence. Again, had the courts been made aware of the potential error rate involved with this type of evidence, the result from the case might have been different.

This is but two of many such stories [140] and a dedicated organisation, the Innocence Project [79], has been established to help other wrongfully convicted individuals prove their innocence.

In the next section, the scientific requirements for evidence and the processes to obtain evidence will be discussed.

## 6.2 Scientific Methods and Digital Forensics

In the introduction of this chapter it was noted that although methods of gathering evidence are usually perceived as being ‘scientific’, such methods often do not actually conform to the rigour demanded by scientific examination. In this section, the focus will be on determining how scientific a method is, and to derive a set of criteria against which the scientific rigour can be measured. This set of criteria will then be further applied to digital forensics in later sections.

The purpose of this discussion is to derive a common scientific approach from which additional scientific rigour may be introduced to the field of digital forensics. The application of such an approach to digital forensics will be discussed in Section 6.2.2.

The dilemma of court decisions based on a misrepresentation of forensic evidence was briefly mentioned in the introduction of this chapter. Despite these cases, forensic evidence is still often presented to a court of law with an implied authority, a notion that the facts derived through forensics are absolutely correct and completely above reasonable doubt. Absolute trust is placed in forensic evidence.

The reason for this trust is the generally accepted notion that the use of the forensic method implies scientific rigour in application of the method. In other words, forensic evidence must be scientifically correct because of the fact that it is forensic in nature.

Science is defined by the Oxford Dictionary [117] as “a branch of study which is concerned ... with observed facts systematically classified and more or less colligated (bound together) by being brought under general laws, and which includes trustworthy methods for the discovery of new truth within its own domain”.

The key question now becomes: *when is something scientific?* Looking at the above definition, it can be argued that an investigative method is scientific when it involves classification of observed facts, using trustworthy methods for discovery of new truths.

This definition however is vague. What is defined as a ‘trustworthy’ method? What is defined as an ‘observed fact’?

The Merriam-Webster dictionary's [101] definition of science is less vague: "knowledge or a system of knowledge covering general truths or the operation of general laws especially as obtained and tested through scientific method". The dictionary further defines scientific method as "principles and procedures for the systematic pursuit of knowledge involving the recognition and formulation of a problem, the collection of data through observation and experiment, and the formulation and testing of hypotheses".

Although the above definition is still somewhat vague, one can conclude from the above that science involves a 'problem statement' combined with 'data collection via observation and experimentation' and finally the 'formulation and testing of hypotheses'.

Still, this definition is open to interpretation and debate. In fact, the definition of science is a known philosophically contentious and controversial issue about which many papers have been written. However, this debate falls beyond the scope of this discussion. The interested reader is referred to [88, 111, 25, 40].

One of the more widely-accepted schools of thought regarding the definition of science [104] was provided by Karl Popper [128] in 1959. Popper's thoughts will therefore be used as a benchmark against which proposed requirements for scientific method will be measured throughout the remainder of this chapter.

Consider the following problem posed by Karl Popper [128, 139], known as the 'White Swan Problem', used to illustrate the limitations of reasoning by induction. Popper proposed a hypothesis stating that "all swans are white". Suppose now that this hypothesis is tested with a sample of 100 swans, all of which are white. This still does not confirm the hypothesis, as the possibility of finding a non-white swan always exists. In the case where a non-white swan is found, the hypothesis would however automatically be proven false. This is an example of a hypothesis that is "susceptible to rigorous test but immune to certain proof" [140].

In his landmark work, "The logic of scientific discovery", Popper [128] puts forward the key observation, based on the above, that it is not possible to prove that a theory (or hypothesis) is true for all possible circumstances, but that it is possible to disprove or falsify a hypothesis. This forms the basis of Popper's theory of falsification (that is, a scientist could and should attempt to falsify his or her hypotheses), an important consideration in creating scientific credibility with regards to forensic methodologies. In other words, even though a hypothesis could not be proven as absolutely true, it should be tested in a rigorous and thorough manner. Cohen [41] summarises the theory of falsification as "refutation can destroy a theory, but finite confirmations cannot 'prove' it — they can only confirm it".

One of Popper's key observations can therefore be summarised as 'critical testing'. Popper noted that "scientific testing should require an attempt to falsify a theory (set of hypotheses), in a rigorous and thorough manner". From this, one can therefore conclude that scientific testing should be reasonably expected to include observation and experimentation with the intent of disproving the proposed theory.

This concurs with the 'formulation and testing of hypotheses', proposed by the Merriam-Webster [101] definition.

Now that hypotheses formulation and testing has been emphasised as foundational to ensuring that an investigative measured can be classified as scientific, the next step would be the application of scientific method as a criteria to evaluate whether or not evidence should be admissible in a court of law. The next section focuses on this application.



## 6.2.1 Using Scientific Method as a Criterion for Admissibility of Evidence in a Court of Law

As mentioned initially, this section will aid in the decision of whether a proposed method by which evidence is obtained or derived is admissible in court. This will be accomplished by questioning the scientific principles employed by such a method.

Over the last century, courts were tasked with deciding the legitimacy and admissibility of various new methods employed in collecting and presenting evidence. Well-known examples of such new evidentiary methods include the use of polygraph tests, DNA evidence and fingerprint evidence. Soon, the question of whether or not a particular method of evidence collection was scientific became the hinge on which the admissibility of evidence would be decided [54].

One of the first recorded court cases to deal with the question of admissibility of evidence based on the scientific nature of the evidence collection method used, is the “Wells Harbour Case” [83]. In this case, regarding the silting up of the Wells harbour, a civil engineer testified as an expert witness regarding the events that took place using observations made using science. This case is remarkable because the engineer testified about events which he did not observe, instead using scientific principles as a foundation for the testimony presented. Such testimony is typically seen as hearsay and not allowed in court. However, in this case, the court allowed the use of the scientific evidence, deeming it as valid evidence and resulting in this case being noteworthy for the use of expert witnesses and scientific evidence.

Subsequent to the Wells Harbour Case, many types of evidence became acceptable based its apparent use of science [140]. Many practices which were not really scientific were accepted by courts under the guise of science. The guise of science was however challenged in the case of *Frye v. United States*, 293 F. 1013 [100], heard in 1923 in the United States. The main point of argument in this case was whether or not polygraph evidence should be admitted to the court. The court found that admissible evidence should be sufficiently established in such a manner that *it has gained general acceptance* in the particular field where it was developed. This has become known as the ‘Frye Standard’ and has become a guideline in a number of judgements involving the admissibility of evidence.

Judges however found the Frye Standard difficult to follow [100] due to the vagueness associated with general acceptance. In 1975, the Federal Rules of Evidence was published and many questioned whether the Frye Standard was rendered not applicable by the Federal Rules of Evidence.

Considering the Federal Rules of Evidence, and the questions surrounding the use of the Frye Standard, in *Daubert v. Merrell Dow Pharmaceuticals, Inc.* (1993) [171, 54, 158], the United States court derived four criteria for the admissibility of scientific evidence, based in part on Frye. This became known as the ‘Daubert Standard’.

The 1959 work of Popper (mentioned earlier), and specifically the criteria of falsification of hypotheses, formed the cornerstone of the Daubert Standard, as can be seen in the first point below.

The criteria of the Daubert Standard are listed below. Evidence should be [158]:

- empirically testable (i.e. it has been or can be tested and should be refutable and falsifiable);
- subject to review by peers as well as publication in a peer-reviewed environment;
- generally accepted by the relevant scientific community;

- governed by the existing standards and controls applicable to its operation (which should be regularly maintained); and
- subjected to and contrasted against the known or potential error rate.

Despite only being applicable in the United States, and being contested by some states in favour of the older Frye Standard, the Daubert Standard is a prominent example of a court ruling relating to evidence admissibility using scientific method as a criterion [100]. The argument of whether or not the Daubert Standard is superior to the Frye Standard falls outside the scope of this discussion and the interested reader is referred to Meaney [100], Lyons [97] and Cheng and Yoon [38].

It should be noted that although standards like the Daubert Standard or the Frye Standard are not applicable in every court around the world, these provide one with a yardstick by which to measure the validity of evidence. These standards serve as gatekeepers to ensure that at least some consideration is given to the scientific validity of evidence in preparation for court.

The Daubert Standard remains as one of the most widely used standards to determine the evidentiary admissibility. As it also considers the hypotheses formulation, falsification and testing advocated by Popper, this standard is deemed appropriate to benchmark whether or not a particular method can be viewed as scientific and admissible to a court of law. As such this standard will be referred to consistently throughout the remainder of the chapter.

In the next section, the criteria of the Daubert Standard will be applied to Digital Forensic Evidence.

## 6.2.2 Applying the Standards of Admissibility and Science to Digital Forensic Evidence

Previously, the need for scientific evidence was articulated. Popper's theory of falsification and the Daubert Standard were introduced to aid in determining whether evidence obtained is scientific and would be considered as admissible in a court of law. However, the question of whether digital forensic evidence conforms to these requirements remains. Furthermore, one can also question the criteria that should be employed to measure such evidence. The remainder of this section will address these questions.

Forensic disciplines often fail to conform to the criteria set out in the Daubert Standard [140]. Digital forensics, the focus of this dissertation, is no exception [30, 103]. Much has been argued in the digital forensics field about the final criterion in the Daubert Standard, that of a 'known or potential error rate' — a debate to which the likes of Carrier [31], Casey [35] (specifically the Casey Scale) and Cohen [41] contributed.

Too often, forensics is seen as a black box [140] — techniques not incorporated into formal study by university faculties, left instead to forensics laboratories and practised by police and prosecutors. Forensics is perceived as a dark art, yielding perfect answers and not subject to scrutiny by courts [140]. Saks [140], notes an example where a court deliberately misinterpreted the Daubert Standard so that fingerprint evidence could be admitted as it was unheard of to question the error rate of fingerprinting (*US v. Havvard* in 2000).

Garfinkel [58] expands on this idea and notes how digital forensics also suffers from the notion of being seen as perfect and not subject to scientific scrutiny. He remarks that digital forensic tools are often accepted based on vendor reputation instead of rigorous testing. Because of the established track record of vendors such as Encase [151] or AccessData [1], any

new tool released by them is assumed to have the same integrity as other, more established and reputable tools.

One can therefore argue that applying scientific benchmarking criteria like the Daubert Standard is vitally important in establishing the validity and trustworthiness of forensics methods used in courts of law. However, the application of such criteria and/or the requirements of scientific methods to digital forensics is not an easy task.

Going forward, the remainder of this section will focus on various ideas regarding how scientific method requirements can be applied to digital forensics.

In 2003, Carrier [32] remarked that the term ‘digital forensic science’ is often used, but that no distinction is made between which of the techniques employ scientific principles and which are mere engineering. He argued that for digital forensics to be scientific, it should be based on the formulation of hypotheses and the attempted falsification thereof as per Popper [128]. Carrier suggests the use of a scientific method based on observation, hypothesis, prediction and testing.

Carrier’s approach involves collecting and observing the relevant information, followed by forming hypotheses from the observed information and then subjecting these hypotheses to testing and falsification. Finally, Carrier’s approach involves making predictions and then testing these predictions against the evidence.

A fundamental observation by Carrier [32] is that “the entire investigation process can be described as a series of hypotheses”. Carrier [32] further makes the important observation of stating that statistics are frequently used to test scientific hypotheses and that a certainty value is often assigned to them. He proceeds to conclude that it is not clear how to calculate certainty values for some digital forensics hypotheses.

Carrier is arguing for the application of the last criterion of the Daubert Standard on digital forensics, namely subjecting digital forensics hypotheses to testing and contrasting them against a known or potential error rate.

In 2009, Garfinkel [58] notes that one of the key characteristics of science is the ability to perform repeatable and controlled experiments that produce consistent and repeatable results. As mentioned earlier, he notes that the results from widely used, commercially available tools are often accepted based on the reputation of the vendor, rather than the principle of repeatability. He further remarks that although an independent evaluation of such widely used, commercially available tools is possible, the work is drastically complicated by the lack of availability of standard test data.

One is therefore led to conclude that Garfinkel is making a case for the application of the first criterion of the Daubert Standard —empirical testability.

Cohen [41], in 2010, makes observation relevant to the debate regarding the application of the scientific method principles to digital forensics. In his work he lists four basic elements of a scientific methodology, applicable to most areas of science, and argues that these should also be applicable to forensic methodologies. The elements are listed below:

- studying the ories (past and current), the methods involved in the field and their experimental basis;
- identifying inconsistencies between current theories and repeatable experimental outcomes;
- hypothesising a new theory that explains refuted hypotheses and performing experiments to test the new theory (in essence, applying the concept of falsification); and

- publishing the results.

Cohen[41] therefore concurs with Carrier [32] on the need for hypothesis-based thinking in terms of applying scientific principles to digital forensics and also addresses two more criteria from the Daubert Standard, namely:

- subjecting work to review by peers and publishing it in a peer-reviewed environment; and
- ensuring that work is generally accepted by the relevant scientific community.

Cohen [41] subsequently agrees with the conclusions of Carrier in [32] and Gladychhev in [63] regarding the need for certainty values. He remarks that the concept of reliability is key to the legal implications of technical and/or scientific expertise, although little information is available on the application of reliability to the digital space.

In this section the work of Carrier, Gladychhev, Garfinkel and Cohen regarding the application of scientific methodology to digital forensics was contrasted and discussed. Note how the arguments proposed in their work correlate with the criteria of the Daubert Standard. One can therefore apply the Daubert Standard not only to admissibility of evidence, but also use the criteria as a means to introduce scientific benchmarking to digital forensics methodologies. More importantly though, one can also conclude that the Daubert Standard criteria largely overlap with the criteria for testing whether a scientific method has been employed.

In addition to the criteria of the Daubert Standard, both Carrier [32] and Cohen [41] emphasised a hypothesis-based approach. This implies the need to create hypotheses from observed information, make predictions and then test the hypotheses. These actions form the fundamental approach in reconstructing the events, as mentioned in the problem statement in Section 1.3.2. In the next section, this approach will be followed and the need for collection of specific data types will be introduced. Ultimately, this leads to the introduction of the XML Accounting Trail Model in Chapter 7.

### **6.3 Applying Scientific Method to the Detection of Data Irregularities**

In the introduction of this chapter, it was established that a proven scientific approach to digital forensic science is required. In the previous section, criteria for the application of a scientific methodology to digital forensics were discussed. It was observed that one of the key ways in which to introduce scientific rigour to a methodology is to create hypotheses based on observations and then test them, utilising falsification and known error rates.

In this section, the idea of analysing items will be introduced. This analysis will focus on the characteristics they exhibit as well as the grouping of items based on their characteristics. From these, hypotheses will be derived in order to better understand the evidence and to aid in reconstruction of the events. More importantly, this section will apply the hypothesis-based approach to the output obtained from the compiler (introduced in Chapter 5), resulting in a set of hypotheses. By using these hypotheses the need for supporting evidence and instrumentation (capturing and measuring the evidence) will be illustrated.

Olivier [115] notes that “most digital forensic investigators will be comfortable with characterising the examination process as a set of hypotheses that are tested and then rejected or

not rejected”. Olivier [115] also notes that decision problems are very similar to hypotheses. A decision problem is a logic problem with a yes or no answer (a Boolean answer) for a given set of parameters [23]. Olivier concludes by noting “that decision problems, just like hypotheses, do not prescribe how an examination should be conducted, but clearly delineates what may be offered as evidence. An ‘accepted’ hypothesis makes a truth claim — as does a decision problem that has been decided”.

Considering Olivier’s observations, and the arguments by Cohen and Carrier in the previous section, it can be concluded that not only does the use of a hypothesis-based approach have merit in establishing scientific rigour, but hypotheses may also be stated as decision problems. Whilst remaining true to the hypothesis-based approach advocated by Carrier [32] and Cohen [41], it has the additional benefit of deducing error rates and predicting time complexity [115]. Although these properties of decision problems fall outside the scope of this discussion, the use of decision problems may be useful in addressing the error-rate concern noted by Carrier [32], Gladyshev [63] and Cohen [41]. The reader interested in decision problems is referred to Olivier [115] for a discussion of these advantages of decision problems.

For the remainder of this chapter, I will continue to refer to the use of hypotheses, although the hypotheses offered will be phrased as decision problems.

Whilst hypotheses are of key importance going forward another consideration also plays a major role, namely the reconstruction of past events in order to determine the nature of tampering as mentioned in the problem statement (Section 1.3.2).

Carrier and Spafford [33] outlines a process for digital event reconstruction using an object that was the effect of an event (effect artefact), and to conduct a backward search of cause artefacts that could have changed the characteristics of the effect artefact.

Applying this process to the evidence produced by the compiler in the previous chapter, a data irregularity becomes an effect artefact, for which cause artefacts should be found. One of the foundational approaches to find cause artefacts, is that of asking basic investigative questions in order to reconstruct events [116]. These include:

- What?
- Why?
- When?
- Who?
- Where?
- How?

Above, both hypotheses and investigative questions have been introduced as part of the process of applying the scientific method to the detection of data irregularities. Prior to continuing with the remainder of the chapter, one additional element needs to be introduced in order to ensure a solid foundational understanding.

Epistemology, the philosophy that studies the nature and foundation of knowledge, [49], provides yet another way in which scientific rigour can be introduced to digital forensic evidence. Cohen [41], in discussing the epistemology of digital forensics, derives the concept of ‘the physics of digital information’. Cohen observed that digital forensic evidence displays

common, albeit not universal, properties that are unique to digital information. He further noted that such properties are similar to the properties displayed by matter, and hence coined the name ‘physics’ of digital information to describe the study of these properties.

These common properties, observed by applying the concept of epistemology to digital forensics, provide important insight into the characteristics of digital evidence. Furthermore, these properties assist with the formation of hypotheses relevant to digital evidence and will play an important role as a basis for the further derivation of hypotheses in the remainder of this chapter.

A selection of the attributes of the physics of digital information, pertinent to the remainder of the discussion, is listed below:

- Digital information exhibits finite granularity, i.e. it can be broken down to the point of the granules of the information (the bit-related nature of digital information) and no further.
- Exact copies can be made of digital information on a bit-level without altering the original in any way.
- An exact copy of digital information bits may be made and removed without removing the original, whereas in the physical world, removing an item renders that item unavailable.
- The bits that make up digital information move very quickly (but finitely quickly) from place to place.
- Digital Forensic Evidence (DFE) is created by artificial means — the bits that make up DFE are generated by automated mechanisms in an artificial way based on how such mechanisms are designed, implemented and operated.

Considering these attributes, one notices that all of these relate to the bit-level characteristics of digital information. Cohen views these bits as a basic building block of digital information [41], analogous to the electrons, protons and neutrons of the physical world. Most digital forensic practitioners should agree — at a most basic level, digital forensics can be broken down as a study of ‘ones’ and ‘zeros’.

One may therefore refer to the attributes above as the bit-level attributes of digital information.

Cohen’s physics of digital information does not however consider the attributes of grouping larger collections of bits (already representing useful data), instead focusing exclusively on attributes of data from a single bit perspective. Investigating the attributes displayed by such groupings is an important part of the remainder of this discussion, providing not only context and direction for the approach taken in the XML Accounting Trail Model in the next chapter (see Chapter 7), but also forming an inherent basis for future hypotheses based on output from a compiler detecting data irregularities, as introduced in Chapter 5.

Going forward, it is important to define these collections of bits in the context of the work performed in this dissertation. Considering the work performed in the previous chapter, regarding using a compiler to detect data irregularities, one could view each of the irregularities as a data bit, as an irregularity is a finite concept that cannot be meaningfully broken down in smaller parts. Furthermore, these data irregularities can be grouped together based on specific characteristics displayed by the irregularities, e.g. the type of error that triggered the irregularity, or the corrective steps that would be necessary to eliminate the irregularity.

However, further grouping is also possible. Considering the notion of investigative questions introduced in the beginning of the section, in conjunction with Carrier and Spafford's concept of backward searches to determine the cause events for data irregularities (effect events) even further groupings become possible. It becomes possible to group data irregularities using characteristics that are not immediately apparent ('deeper' characteristics). Examples include the type of action that caused them (the "why" investigative question), the date and time when the irregularity occurred (the "when" investigative question), the person by whom the data irregularities were introduced (the "who" investigative question) and so forth.

In the remainder of the section, these further 'deeper' characteristics ideas above will be engaged and the evidence collected in Chapter 5 will be analysed by means of the use of hypotheses based on these 'deeper' characteristics.

### **6.3.1 A Set of Hypotheses Based on the Evidence Gathered Using Compilers to Detect Data Irregularities**

As mentioned above, the purpose of this section is to apply the concept of developing hypotheses based on the evidence gathered by using a compiler to detect data irregularities. Previously, it was demonstrated that data irregularities can be grouped based on characteristics, both obvious characteristics and characteristics that are not immediately apparent. The latter group of characteristics, referred to as 'deeper' characteristics throughout the remainder of the section, are derived by reconstructing the events using investigative questions. In doing so, a measure of scientific rigour will be introduced to the model, enabling the model to be subjected to testing and falsification as required by the Daubert Standard [171] and Popper [128]. Furthermore, the hypotheses established in this section will be used to point out the need for additional evidence which can only be derived by instrumentation (setting up measuring devices to gather information) [117].

In order to formulate hypotheses, it is helpful to consider the various components of evidence that are available. As was noted in the work of Osterburg and Ward [116], the use of investigative questions greatly aids the event reconstruction process. The reconstruction process is central to the work performed in this dissertation as seen in Section 1.3.2. Applying these investigative questions to the data irregularity would lead to the following:

- What constitutes the data irregularity?
- When did the data irregularity occur?
- Who modified the data?
- Why was the data modified?
- How was the data modified?

Note that the above questions are illustrative only, and many more variations of these questions can be used. However, the concept of hypothesis-forming remains the same and therefore a more exhaustive example is not required.

Considering the questions above, one notices that each of these questions centers around a core theme. Table 6.1 lists the mes for each of the questions above.

Table 6.1: Core themes for investigative questions applied to the evidence gathered using a compiler to detect data irregularities.

Investigative Question	Core Theme
What constitutes the data irregularity?	Data Content
When did the data irregularity occur?	Time
Who modified the data?	User Identity
Why was the data modified?	Cause of Event
How was the data modified?	Method

In the introduction of this section, it was noted that Olivier [115] suggested the use of decision problems to phrase hypotheses. Going forward, the core themes will be used to formulate sample decision problems or hypotheses, in order to illustrate how detail evidence can be used by an investigator to start performing event reconstruction.

Table 6.2 illustrates the process of hypothesis-formation based on the core themes listed in Table 6.1.

Note that the term ‘the books’ is used to refer to the accounting data in the tables below. The intention of Table 6.2 is to illustrate the power of hypotheses used in conjunction with available evidence. Also note that Table 6.2 does not constitute complete list of hypotheses relating to the available evidence, but is used rather to illustrate the process of hypothesis-formation.

Table 6.2: An illustrative set of decision problems (or hypotheses), ordered by core theme, drawn from the evidence gathered using a compiler to detect data irregularities.

Core Theme	Decision Problem
User Identity	Has user x changed any entries?
	Has user x logged in before?
	Was user x able to access y?
	Do the entries modified by user x conform to pattern y?
Time	Were the books edited at time x?
	Was entry x modified before or after entry y?
	Were all changes made to the books between dates x and y?
	Which changes to the books were made outside of times x and y (e.g. outside of office hours)?
Data Content	Were changes with characteristic x made to entries at regular intervals?
	Did entry sequence P occur in the books?
	Was entry x changed?
	Was entry x changed y number of times?

*Continued on next page*



Table 6.2 – *Continued from previous page*

Core Theme	Decision Problem
Cause of Event	Were books by mistake whilst saving changes in the data file?
	Were the books edited in order to commit crime according to a certain modus operandi, e.g. salami attack (refer to Chapter 5)?
Method	Were books edited by directly modifying the XML data file?
	Were the books edited using accounting software?

When considering the sample hypotheses listed in Table 6.2, one notes that only a single core theme has been used in the formulation of each hypothesis. It is however possible to combine two or more core themes, yielding new and more intricate hypotheses. For example, combining time and user identification results in a hypothesis such as ‘were the books edited at time x by user y?’.

In combining these evidence sources, the number of hypotheses greatly increases, and as such these hypotheses are not included here. However, it should also be noted that the precision and detail of the hypotheses are greatly enhanced as well, enabling the digital forensics investigator to more accurately reconstruct the events leading to a data irregularity, as evidence reconstruction typically involves a combination of investigative question types [116].

In the next section, the applications of a set of hypotheses, as derived above, will be discussed, coupled with applying the next steps of the scientific method, namely testing the hypotheses and attempting falsification of the hypotheses.

### 6.3.2 Applying Scientific Rigour to the Set of Hypotheses

In the introduction, it was noted that forensic methods are not necessarily synonymous with adhering to scientific rigour. It was further noted that many challenges and gaps exist with regards to the application of scientific rigour to digital forensics. Finally, it was also noted that applying scientific rigour to digital forensics was possible, and the idea of using hypotheses in order to aid with the process was proposed.

In the previous section, scientific rigour was introduced by means of creating hypotheses from the data gathered by using a compiler to detect data irregularities. However, when considering these hypotheses and their core themes, it becomes that clear a whole lot of evidentiary data is missing.

A typical XML financial data file only contains financial transactions reflecting the current state of the books [12, 59]. Typically no meta-data exists from which core themes can be derived, unless explicitly recorded by the software package generating the XML financial data. Examples include the time at which a data irregularity was introduced, by whom the data irregularity was used and the content of what was modified, to name a few of the core themes that cannot be addressed. The interested reader is referred to Section 7.2.1 where the notion of attempting to extract evidence from an XML file is discussed in more detail.

In essence then, the evidence required to test and verify the hypotheses, ensuring that the appropriate scientific rigour is involved, is absent. Therefore it is concluded that the evidence

provided by data irregularities themselves is insufficient to fully address the problem statement in Section 1.3.2.

Using the analogy of forensic pathology from the introduction of Chapter 5, one might say that it is not possible to establish an accurate cause of death as the entire body of evidence is not present.

How then should this shortcoming be rectified? What can be done to ensure that adequate evidence is available to ensure that reconstruction of the events leading the data irregularity is possible?

This problem is very similar to a problem that has been experienced for several years in the aircraft industry. How can one reconstruct the events leading to an aircraft crash in order to ensure that such an event does not re-occur [28]?

Bibel [19] notes that forensic investigators seeking to identify the cause of an aircraft crash and to reconstruct the events leading up to the crash have long struggled with only having available the aircraft wreckage remaining after the crash. This process is termed “obtaining probable cause of an accident” [19]. For many years investigative teams tasked with obtaining probable cause failed to adequately prove their hypotheses regarding the cause of accident, as evidence was not available to test and falsify these hypotheses.

In 1960, flight 538 from Trans Australia Airlines was involved in a crash, killing all 29 passengers on board [172]. In the aftermath of the accident, a Board of Accident Inquiry was established by the Australian government, tasked with investigating the accident. The inquiry however failed to determine the particular cause of the incident.

The Board of Accident Inquiry realised the need for more detailed evidence and was the first to mandate that all aircraft larger than a particular size be equipped with instrumentation to record flight details, also known as ‘flight data recorders’ [19, 172]. Soon, other government bodies followed, such as the 1964 requirement by what is known today as the United States’ Federal Aviation Authority that all turbine and piston aircraft with at least four engines should have flight data recorders [19].

Flight data recorders, better known by the colloquial term ‘black boxes’, have been in use for many years, originally by military institutions to gather test flight data of military aircraft, and also independently developed by a mechanical engineering professor from the University of Minnesota at a request from the Civil Aeronautics Board in 1943 [2].

Flight data recorders are designed to record instructions by the electronic systems responsible for aircraft manoeuvring, tracking and flight operations, such as altimeter (height) reading, speed, inputs to ailerons and trim, engine parameters and other such information [19]. Typically, the flight data recorders are also coupled with cockpit voice recorders, recording the conversations of the aircraft pilots responsible for flying the aircraft.

Similarly to this approach, in order to ensure that sufficient evidence is available to determine probable cause of aircraft accidents, additional instrumentation is required in order to record the required evidence such that the afore-mentioned hypotheses can be tested. This idea is developed into a model to record forensic evidence on a continuous basis (forming the basis of forensic readiness, refer to Section 2.3.3) allowing for the evidence of events leading to data irregularities to be available when required. This model, together with its associated requirements, is introduced in Chapter 7.

## 6.4 Conclusion

The purpose of this chapter was to determine which attributes and techniques should be used to determine whether an idea or notion is ‘scientific’, and applying these to digital forensic methods of evidence gathering.

Forensic science is often seen as infallible and by definition, scientific in nature. In the introduction of this chapter, this notion was questioned and several cases where forensic science was proved to be flawed were cited. It was concluded that forensics is not scientific by default, but instead that specific criteria exists in order to ensure the scientific rigour of evidence gathering methods.

The chapter proceeded to discuss the definition of the term ‘scientific’ and proposed various possible definitions of the term. Popper’s Philosophy of Science, specifically his theory of falsification, was introduced as one of the cornerstones of the scientific method.

Next, it was argued that the particular question of scientific criteria was also addressed by various courts of law. It was pointed out that various courts attempted to define whether or not a method was admissible in a court of law, which led to the introduction of the Daubert Standard. Also, it was established that the Daubert Standard incorporated falsification as one of its requirements.

The discussion then continued by noting how the Daubert Standard [171] and Popper’s theory of falsification [128] could be applied to digital forensics. A comparative study of various arguments regarding the process of testing whether or not digital forensics is scientific in nature was presented and it was concluded that the outcome complements the Daubert Standard. It also noted a strong emphasis on the use of hypotheses and the subsequent testing of these hypotheses [32, 41].

Continuing from the above, the question of event reconstruction as per the problem statement in Section 1.3.2 was introduced. Drawing from the above observations by Carrier [32] and Cohen [41], it was noted that various hypotheses can be proposed based on central themes derived from asking basic investigative questions and applying these to the evidence presented by executing a compiler XML financial data in order to detect data irregularities. These hypotheses were classified based on these central themes and presented to the reader as a practical example of how hypotheses should formed.

Finally, it was concluded that testing of these hypotheses was not possible as the evidence required to falsify the hypotheses was not available. It was noted that the lack of evidence was analogous to the lack of evidence in aircraft investigations that led to the introduction of instrumentation to record flight information on a continuous basis so that evidence is available in the event of an accident. Drawing on this example, it was noted that a need exists for similar instrumentation regarding data irregularities and evidence surrounding time, user identification, data content, etc. It was then concluded that additional work is required to record such evidence. This concern is addressed in the next chapter, introducing the ‘XML Accounting Trail Model’ (see Chapter 7).



# Chapter 7

## XML Accounting Trail Model

### 7.1 Introduction

In the Introduction to this dissertation, the problem statement was constructed (see Section 1.3.2) as follows: “XML financial data is susceptible to tampering due to the human-readability property required by the XML data specification. Upon presentation of a set of XML financial data, how can one determine whether data has been tampered with, and reconstruct the past events so that the nature of tampering can be determined?”.

Considering the problem statement, one notes that Chapter 5 related exclusively to the first part of the problem statement — providing a manner in which one could determine whether data has been tampered with upon presentation of a set of XML financial data. However, the process of reconstruction of past events in order to determine the nature of tampering has not yet been addressed.

This then, is the aim of this chapter, as summarised by Section 6.3.2 of the previous chapter. This chapter intends to focus on the process of gathering evidence for use in event reconstruction by means of instrumentation. Much like the flight data recorders in an aircraft accident, this chapter proposes the use of instrumentation by means of a model to constantly gather evidence. This evidence can then be used to assist in event reconstruction, should data irregularities occur.

As explained previously in Chapter 3, XML data files have several defining properties. One of these properties, the property of human readability, is particularly relevant for the discussion in chapter.

XML files require both human and machine readability as this ensures that data formats cannot become deprecated as the data will always be interpretable by a human. Furthermore, XML is self-describing, meaning that a human can interpret the data without any previous knowledge. This allows a user to open the data in any text editor and to perform easy data editing without tying the data to a specific vendor or piece of software in order to be able to edit it [62].

Human readability, despite its practicality and being a requirement of XML, poses a large drawback. Being human readable, XML documents are vulnerable to modification by unauthorised individuals in a trivial manner. As described above, a user can simply open the data file in a text editor, modify it, and save the changes. Whilst posing the advantage of being editable without requiring special software, this also enables individuals to easily make unauthorised changes to the data in the manner described above.

This chapter starts by deriving the evidence required to be retained by the XML Accounting

Trail Model. The chapter then discusses how such evidence could potentially be obtained, and concludes that additional software in the form of ‘proxy’ between the XML data file and the editor of the data file is required. Subsequently, the chapter analyses the process of recording of evidence using such software and the necessary components of this software. Finally, the requirements and details of the XML Accounting Trail Model are derived from these components and evidence requirements, allowing the chapter to conclude by constructing the model and providing an overview of the model.

## 7.2 Developing the XML Accounting Trail Model

In the introduction, it was noted that evidence needed to be recorded in order for it to be available when needed for event reconstruction. This section focuses on the XML data, with the intent of determining which evidence should be retained in order for the proposed model to be of use. Once a complete list of requirements or specifications for the model is constructed, the remainder of the chapter will reverse-engineer the requirements and derive the model specifications needed to comply with these requirements.

However, before proceeding, it is important to define the ‘threat’ (a use case illustrating how unauthorised data modification may take place).

*An instance of unauthorised or malicious modification of XML data. This can be achieved by either directly editing the XML file (using a text editor) or editing the data in the financial accounting program and updating the XML file.*

In the problem statement, one of the key requirements was the reconstruction of past events that led to the data irregularity. This poses the question: in the event where the threat (as defined above) occurs, how can one reconstruct the events leading to the data irregularity?

Considering the work of Osterburg and Ward [116] noted in Chapter 6, one concludes that event reconstruction relies heavily on the use of the five investigative questions, namely ‘What?’, ‘When?’, ‘Where?’, ‘Why?’ and ‘How?’. In the discussion below, these questions will be used to derive the evidence required to reconstruct the events leading to a data irregularity being detected as per the problem statement.

In trying to reconstruct these events, the first important consideration is that of ‘When?’. Chisum and Turvey [39] emphasises the importance of the ‘When?’ by remarking that the establishment of an event timeline is critical for successful event reconstruction.

A second consideration is that of determining the exact detail the events that took place [63]. This corresponds to ‘What?’ and such information can only be obtained by determining the exact nature of the data modifications made prior to the data irregularity being detected.

In the previous paragraphs, the need for a timeline has been established. Furthermore, it was demonstrated that specific events details are required and that these events can then be plotted on the timeline. However, a third piece of evidence is required in order to successfully reconstruct the events as per the requirement posed as part of the problem statement. This last piece of evidence that is being required is that of ‘Who?’. It is key to determine who was responsible for making the modifications at a specific point in time, as actions need to be attributable to a specific person or entity in order for event reconstruction to successfully take place [39]. However, it should be noted that full attribution is not possible using only digital

forensic information [41]. Therefore, determining the user account responsible for making the changes would be sufficient for the purposes of event reconstruction in this chapter.

At this point, one notices that two of the investigative questions, namely ‘How?’ and ‘Why?’, remain unanswered. In the context of the work performed in this dissertation, the ‘How?’ has already been answered by the threat definition, which was drawn from the problem statement noting the XML weakness posed by human readability.

As for the ‘Why?’, this question would be necessary in establishing the motive for the changes made to data, but would not be necessary in reconstructing the actual events leading to the data irregularity. As such, this question is deemed out of scope of the work performed in this chapter and will not be considered.

In summary, one can therefore conclude that three evidentiary facts would, at minimum, be required to successfully reconstruct the events. These are defined as the minimum evidence set, defined below.

**Definition 7.2.1.** *Minimum set of evidence required:*

- *Evidence showing the details of the data modifications;*
- *Evidence stating the date and time of the modification; and*
- *Evidence showing who modified the data.*

## 7.2.1 Obtaining the Evidence

In the previous section, the minimum set of evidence required to successfully reconstruct the events leading to the data irregularity was derived. Now that the evidentiary requirements are established, the next step is that of obtaining such evidence in a manner that will allow the evidence to be admissible in court, as per Chapter 6.

In this section, it will be demonstrated that the required evidence is not available by default (from either the XML file or other sources of evidence) and that a dedicated, additional software proxy is required to gather evidence.

Consider the orem below, labelled theorem 7.2.1.

**Theorem 7.2.1.** *It is possible to use the XML data file itself as a source to obtain the evidence required.*

As explained in Chapter 3, the XML data file contains two types of data, namely: source data and markup tags. Two options now exist for obtaining the evidence.

Most XML accounting formats do not retain a historic view of the data as this is cumbersome and not typically required [47]. Therefore, using this data, one can only view the current state of the financial information. Also, the source data contains no traces of modification date or time or data regarding who performed the modification. In other words, no meta information can be extracted from the XML data. One is therefore led to conclude that none of the evidentiary facts required by the minimum evidence set can be extracted from the source data.

Even if the XML accounting data did contain the necessary evidence, one cannot rely on such evidence as this evidence too may have been modified in the event of direct modification of the XML file using a text editor. It is therefore possible to disprove theorem 7.2.1 and conclude that the XML data file by itself is not a sufficient source of evidence. Obtaining the

evidentiary facts required by the minimum set of evidence has to be obtained from an evidence source other than the XML data file.

However, the need for additional sources of evidence does not yet allow one to prove the need for the use of an additional software program deployed as a proxy in order to collect the required evidence. Instead, other sources of evidence seem to be readily available, such as default applications, viewers and tools in the system environment. Casey [35] mentions four sources of digital evidence available on most systems:

- File systems and file related data;
- Evidence that can be retrieved using data recovery tools (slack space, deleted files, etc.);
- Log files; and
- Evidence residing in the operating system internals, e.g. the kernel.

Although not necessarily an exhaustive list, these sources of evidence should provide a reasonable indication of whether system information which is available by default could be used to obtain the evidence required. Consider therefore another theorem as defined below.

**Theorem 7.2.2.** *It is possible to use system information, which is available by default, to obtain the evidence required.*

Evaluating the correctness of theorem 7.2.2 involves deciding whether any one or more of the evidence sources noted by Casey above can satisfy the minimum set of evidence required.

The first evidence source mentioned by Casey is that of file system related data. It is possible to obtain the date and time of last modification as well as the username of the person who modified the file [147]. However, the specific details of the exact modifications are not available. Furthermore, it is only possible to tell the date and time of the most recent modification, meaning the complete history of modifications is lost. As a result, file system related data by itself does not provide the evidence required to allow reconstruction of the events.

The second source of evidence mentioned by Casey is data that has been revealed through the use of data recovery techniques, such as the recovery of deleted files. The recovery of deleted files poses two drawbacks [35] that makes this approach less suitable for our purposes — 1) it is only able to recover a complete copy of the original file if the disk space which originally contained the file is not disturbed (i.e. overwritten); and 2) with subsequent overwriting of the disk space, it becomes extremely difficult to recover the original data.

Although it may be possible to obtain the details of a previous version of the XML accounting data file using data recovery, the full recovery of the file is by no means guaranteed as it may have been overwritten. Also, there is no way to determine which previous version of the file has been recovered as only the file content can be recovered, the meta information such as file date and last accessed time is typically not recoverable [35]. Therefore, a recovered version may be the previous one, or another, much older version.

This method of recovering deleted data is therefore not reliable enough to provide accurate data of the exact modifications that were made prior to the data irregularity being discovered and can therefore not be used as a source of evidence in reconstructing the events.

According to Casey [35], the third possible source of evidence on the operating system is that of log files. Typically, the default log files only record system events [147]. Unless specifically configured, such logs do not log events specific to the XML accounting data. Instead, it



contains operating system log data which may or may not be relevant to the facts that must be proven. Even in the most favourable case, where custom logging was created specifically to trigger upon modification of the XML data, the evidence still only enables partial extraction of the minimum evidentiary facts. It is only possible to establish that a specific user modified the XML data at a specific time. Although better than any of the previous sources, it still lacks the critical information regarding the detail and extent of the modification.

The final source of evidence according to Casey, is that of the operating system internals. Operating system internals typically refer to using operating system constructs such as the kernel, file system and other building blocks enabling the functioning of the operating system [147]. However, similar to log files, unless specifically modified and enabled, these internals do not retain any evidence relating to time of activity, user activities or authentication information. One therefore concludes the same regarding operating system internals as one would regarding logging, namely that the minimum set of evidence cannot be obtained from operating system internals.

Therefore, it is possible to conclude that Theorem 7.2.2 has been disproven. No combination of one or more sources of system information could provide the minimum set of evidence required to reconstruct the events in a successful manner. In summary, evidence reconstruction is not possible using only information that is available by default. Instead, special, additional evidentiary sources which are required.

This implies that a dedicated, additional tool should be deployed to record the necessary evidence so that such evidence is available when event reconstruction needs to take place. The next section focuses on the requirements for such an additional tool.

## **Real-Time Logging of Evidence**

In the previous section, it was noted event reconstruction requires evidence that is not available by default. Instead, a dedicated tool is required to record such evidence. Recording such evidence can be achieved in two ways, namely: 1) record events as they occur (real-time) or 2) attempt to recover evidence on a post-event basis.

As established previously when investigating the use of recovered data as a source of evidence, a post-event approach is not feasible as the detail of the modifications will be lost if not captured in real time. Instead, a real-time system is required to capture the evidence as events occur.

However, the recording information in real-time is quite complex [170]. Real-time recording of evidence adds significantly complexity to the requirements for a software tool to capture the minimum set of evidence and keep the evidence safe from tampering. In this section, these additional requirements will now be investigated.

Due to the complexity of capturing real-time information, the requirements necessary to perform real-time operations have become the subject of dedicated field of study, called real-time computing [170]. This led to a well-known, standard solution for recording data in real-time, used in systems such air traffic control and command and control systems. Ionescu and Aurell [45] define a real-time system as “a computational process that has to respond to internal or external stimuli in determined periods of time”.

Williams [170] concludes that one could implement a real-time system in one of two ways, namely: 1) using a dedicated proxy or 2) using interrupts. However, the use of interrupts will not be considered due to its complexity, and the fact that interrupts may be lost or ignored,

causing the event to not be recorded [170]. Considering the conclusion by Williams, it is therefore necessary to implement a real-time system using a dedicated proxy. Utilising a dedicated proxy requires that access to the XML file needs to be controlled via a single point of access. In essence, modification to the file should be possible if, and only if, the modification occurs by means of the dedicated proxy.

However, this idea of a ‘dedicated proxy’ is not a novel one. Pfleeger & Pfleeger [124] discuss the use and implementation of such ‘dedicated proxies’ or ‘reference monitors’ at length. Going forward, due to the recognition of the term ‘reference monitor’ in IT security and the body of knowledge available regarding reference monitors, the term ‘reference monitor’ will be used instead of ‘dedicated proxy’.

Pfleeger and Pfleeger define a reference monitor as “a portion of code that controls accessibility of objects” [124]. They further observe that each reference monitor should comply with three requirements in order to be effective: 1) it should be *tamper-proof*; 2) it should *always be invoked* when access to protected information is required; and 3) it should *be small enough* to be subjected to thorough analysis and testing, in order to ensure that all components are functioning correctly and can be trusted.

Considering the importance of a reference monitor and the fact that the requirements for the additional software tool to record evidence in a real-time manner is currently collected, the next section will focus on the use and protection of a reference monitor.

## Use of a Reference Monitor

In the previous section, it was concluded that the use of a reference monitor is the only way to log real-time evidence regarding changes. However, in the event that the reference monitor is bypassed, events will not be logged in a complete and accurate manner. As a result, the evidence gathered will not be complete and event reconstruction will not be possible. Therefore, it is key to consider Pfleeger and Pfleeger’s [124] requirements regarding always invoking the reference monitor.

Considering the fact that modification of the XML accounting data file is initiated by the user, it is not trivial to ensure that the reference monitor is always invoked. Furthermore, by the very nature of XML, it can be modified without the reference monitor being involved. The key challenge therefore becomes protecting the XML data from modification by any means other than the reference monitor (forcing the user to invoke the reference monitor in the event that modification is required). Also, such protection should remain compliant with them XML standard, in order to remain classified as ‘XML’ data. In essence, the XML file therefore needs to be protected from tampering.

In order to tamper-proof the XML data file, it is necessary to first define that which would be considered as tamper-proofing. The reference monitor shall be considered tamper-proof if: 1) it is sufficiently tamper-resistant as to dissuade tampering and 2) the XML files should remain readable to all users but the content is protected from modification using a means other than the reference monitor. The latter of these requirements is of prime importance in order to maintain compatibility with the XML standards.

Given physical access to a system and unlimited time, virtually all systems are susceptible to tampering [60]. Tamper-proofing, in the context of this dissertation, is therefore defined as “being sufficiently tamper-resistant as to dissuade tampering”.

Now that the requirements and scope for the tamper-proofing scheme have been defined,

is possible start work on tamper-proofing the XML file. Tamper-proofing is a well-known challenge in the sphere of both software and web-security, and several solutions have been offered by the community involved in these areas [43, 60, 96, 141]. However, requiring tamper-protection of a file in such a manner that editing is not possible but the read-ability of the document is kept intact is a well-known problem.

Email tamper-protection has the exact same requirements — content should be retained in a manner so that it is readable to all but cannot be modified after it was sent. This is achieved by means of the use of digital signatures and has become an accepted industry standard [159, 57].

Digital signatures provide a way to guarantee that the content of a document or email is pristine and not tampered with. Furthermore, it allows the sender of the document or email to bind his or her identity to the document or email and claim ownership of it. Using digital signatures allows one to prove that the owner of the signature wrote the document or email and that no-one changed the content of the document or email en-route to the receiver[57].

Similarly to the use of tamper-protection in email, for the purposes of the work performed in this dissertation, XML data needs to be protected from tampering as well. However, XML data poses an additional challenge, namely compliance with the XML standard. Due to the fact that the XML standard [26] mandates user-readability, it would be counter-intuitive to merely encrypt the XML data, as it would cease to be user-readable and therefore cease to be compliant with the XML standard. As such, the strategy for email tamper-protection cannot simply be duplicated in order to tamper-protect XML data.

In this case, the email data may be substituted with the XML document data and one can simply attach a digital signature to the document with the reference monitor as author. This may be added as an additional XML node, thereby leaving the remainder of the document and its XML structure intact. In doing this, the typical encryption of the entire document [159] is circumvented, instead only encrypting the hash and adding it to the document as an additional data element. By encrypting the hash value with the author's private key, the hash value cannot be merely substituted by another in the event that the XML data was changed by not invoking the reference monitor. This approach allows for tamper-proofing whilst maintaining readability, satisfying the above requirement regarding ensuring that the reference monitor cannot be bypassed.

Now that it has been established that the only way of obtaining the evidence necessary to enable event reconstruction is by means of a reference monitor that is always invoked, it becomes necessary to consider how the reference monitor should operate in order to collect all the required evidence. The next section focuses on the process of collecting the evidence by the reference monitor.

## 7.2.2 Collecting the Evidence

As mentioned in the introduction, the purpose of this chapter is to propose a model that addresses the requirement of instrumentation by enabling the collection of required evidence for event reconstruction regarding data irregularities. In Section 7.2, it was noted that a minimum set of evidence is required to enable event reconstruction. This minimum set of evidence can be summarised as follows:

- What was modified?
- When was it modified?

- Who modified it?

Considering that the reference monitor needs to collect evidence to satisfy the above requirements, this section will be dedicated to deriving the requirements necessary for the reference monitor in order to be able to collect the evidence. This will be achieved by discussing each of questions making up the minimum set of evidence and determining the method required to collect the evidence needed for each of the questions.

By the end of this chapter, these mechanisms, together with the requirements for obtaining the evidence will be integrated to form the building blocks of the XML Accounting Trail model.

### **Collecting Evidence Showing ‘What was modified?’**

The purpose of the following is to discuss the actual mechanisms used to collect the required evidence. The first type of evidence that needs to be collected is evidence that addresses the question of *what was modified*.

As noted previously, a reference monitor is required to log evidence in real time. Given the existence of a reference monitor, and XML data that changes regularly, it becomes necessary to store details of the change history so that it is possible to see what was changed with each modification of the XML file.

The problem of saving the change history between file versions is a known problem for which a standard solution exists. This problem is noted in the area of source code development [126] and the solution is termed ‘version control’. Pilato et al [126] define version control as “the procedure in managing changes made to information”. They further state that version control is particularly suited to “people using computers to manage information that changes often”. Version control, therefore, is particularly suited to examining the history of exactly how data changed through time.

From the above, one concludes that version control would be exceptionally well-suited as a way of managing the content of the XML data file and obtaining the history of all modifications committed during the lifetime of the XML data file.

Version control works on a similar principle to incremental backups [126]. An initial copy of the file is stored when it is first checked in, with modifications stored as subsequent differential files or deltas. One can therefore easily roll back or forward between versions, as the relevant deltas are simply applied to the initial snapshot of the file in the appropriate order. This approach is very handy for investigators, as it is flexible enough to allow them to view subsequent modifications in both a time-line and user-by-user manner.

The next section will now focus on collecting evidence showing when the XML file was modified.

### **Collecting Evidence Showing ‘When was it modified?’**

The second evidentiary fact for which a method to collect evidence needs to be derived is that of *when was it modified*. The primary means of doing this is by adding a timestamp to each action being logged.

Timestamps originate from rubber stamps used in offices to stamp the current date and time on paper documents when processed or to indicate the time of receipt (e.g. letters in a post office) [169]. Timestamps are one of the cornerstones of a digital investigation [168] and

are typically used in forensics to create a time-line in order to assist with event reconstruction [168].

Timestamps however have one weakness, namely that in order for an accurate timeline to be established, the time used to create the timestamps needs to be correct. Computer clocks are prone to drift [148], in other words, running too fast or too slow. If clock drift is present in evidence, the timestamp information can still be used for an investigation, as it still provides a sequence of events and allows for some event reconstruction.

Hosmer [76] notes that timestamps can also be affected by intentional or unintentional modification by a user on the system. The user may adjust the time of the clock forward or backward in a deliberate fashion, or inadvertently enable a feature such as daylight savings time. In such a case, timestamp evidence obtained cannot be used to reconstruct the timeline of events. Therefore, the use of a timestamp obtained from a local clock source is to be avoided as far as possible and another, ‘correct’ time, needs to be used in timestamps.

Hosmer also notes the need for a “secure and auditable timestamp” that is retrieved from a trusted time source in [76]. He concludes that timestamps should be accurate, authenticated and have integrity. It is therefore of key importance that the reference monitor use only the correct time that can be shown as accurate, authenticated and having integrity in logging timestamps.

One concludes that some means of obtaining such a ‘correct’ time is therefore needed. Some process is therefore necessary in order to describe the process of ensuring the security and auditability of a time stamp. In Figure 7.1, an algorithm is proposed to achieve the outcome of obtaining the ‘correct’ time. The discussion below will step through the detail of the algorithm and explain its steps in detail.

In step 1 of Figure 7.1, an attempt is made to connect to a non-local trusted time-source. Such a time-source should provide a verification that the timestamp provided is secure, for example, providing a secure timestamp via either the ANSI ASC X9.95 or RFC 3161 standards for requesting a trusted timestamp [76]. Due to the fact that such a service provider may be temporarily not available, it is advisable to have multiple trusted time-source providers (TTSPs). This process is addressed in steps 2 to 4.

In the case where a TTSP is available, step 5 in Figure 7.1 is performed and a trusted, secure timestamp is obtained.

However, the external network connection allowing traffic to non-local destinations may not be available (e.g. the Internet Service Provider may be down). In such a case, an alternate source for a timestamp is required, and the timestamp will no longer be secure and trusted. One such alternate source which is at least more trustworthy than the local clock on the system, would be that of a local Network Time Protocol (NTP) server providing time information to all local servers. A typical example of such an NTP server driven time distribution is that of a NTP-synchronised domain controller providing time to all member servers in a Windows domain.

To cater for such a case, the loop induced in steps 2 to 4 will be terminated in the event that all entries in the list of TTSPs prove to be unavailable. Steps 6 and 7 in Figure 7.1 then provide an opportunity to search for a local NTP server from which a timestamp can be obtained.

Should the timestamp be successfully obtained (step 8), a log entry will be created stating that a secure, trusted timestamp could not be obtained and that a local NTP timestamp was used instead. However, it is possible that no local NTP server has been configured, which would cause steps 6 and 7 in Figure 7.1 to fail. If no TTSP can be obtained and there is no local NTP server available, one is left with no other option than to rely on the local system clock for

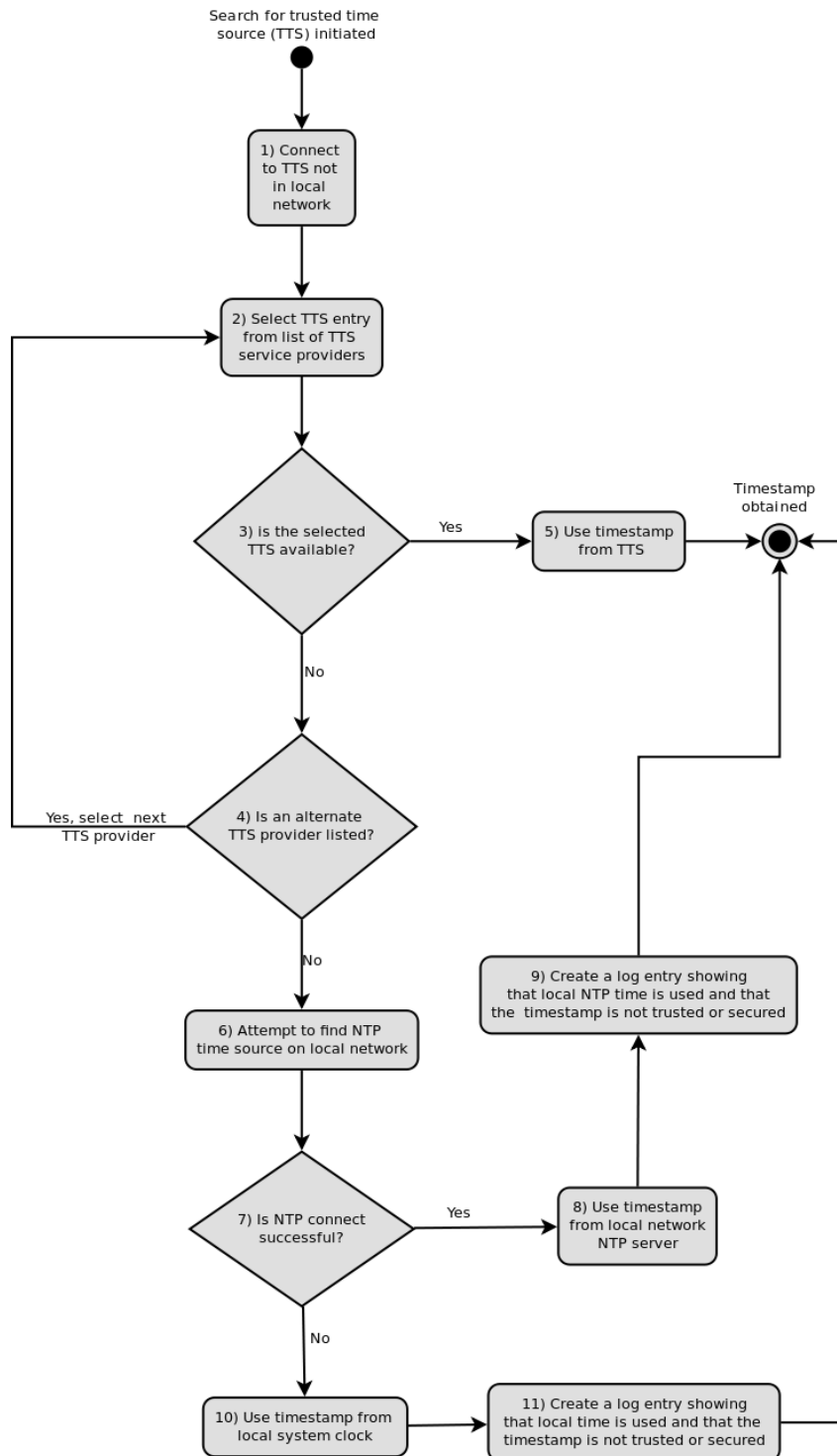


Figure 7.1: An algorithm showing how to obtain and use a trusted, secure timestamp.

obtaining a timestamp.

Steps 10 and 11 show how this process is followed. It is important to note that again, a log entry will be made stating that the timestamp has been obtained from the local system clock and that the timestamp is not secure or trusted.

Finally, it important to determine how often this algorithm should be followed. To a certain

extent, this configuration would be determined by the administrator deploying the reference monitor, but at a minimum, a new timestamp should be obtained each time the XML file is required to be modified. In essence, the reference monitor should act as a transparent proxy, allowing any file modification after recording the required evidence.

Now that the process of obtaining evidence regarding the timeline of events has been discussed, one more piece of evidence is still required. As it stands the reference monitor is only able to collect evidence regarding what was modified and when it was modified, but no information as to who modified the data is currently available. In the next section, a method for collecting such evidence is proposed.

### **Collecting Evidence Showing ‘Who modified it?’**

As mentioned at the start of this section, it is necessary to determine how the reference monitor should act in order to record the evidence required to reconstruct the events leading to the discovery of the data irregularity. Previously, it was determined that the reference monitor needs to make use of version control in order to record the evidence of what was changed, as well as employing timestamps in order to record evidence of when changes were made.

The final piece of evidence which needs to be obtained is that of *who modified it*. In this section, the focus will be on the techniques that should be employed by the reference monitor in order to record evidence of the user responsible for making modifications to the XML accounting data.

In order for evidence to be useful, actions have to be attributable to a specific person or user [41]. Attribution, the act of ascribing a deed or action to a specific individual [101], is therefore of vital importance in digital forensics. Cohen [41] however notes a very important fact, namely that digital forensics cannot provide absolute attribution tying a specific person to digital actions. Instead, the username responsible for committing specific actions can be traced, but a separate investigation, outside of the domain of digital forensics, is required in order to tie back the username to a specific individual. This process falls outside of the scope of this discussion, and as such the discussion will be limited to the attribution of actions to a specific username.

When considering the threat definition, one notes that the details of the user responsible for performing the modifications are required in order to be able to attribute the actions. In order to modify the file, the user has to have a current session (in other words, be logged in to the system) [147]. Obtaining the details of the current user, however is not trivial.

Authentication, the process of accurately identifying a user requesting access to a system and verifying that he or she is whomever he or she claims to be [41], is typically used to establish a user’s identity. Authentication, is typically accomplished by requesting the following from a user:

- unique information that is *known* only by the user;
- something unique the user *has*; or
- something unique the user *is*.

The authentication process is typically performed upon user login by the operating system (OS). Many applications rely on the authentication performed by the operating system (in essence, allowing for single sign-on) [124]. However, it is possible to circumvent the process

of authenticating to the OS. For example, the OS may be configured to bypass authentication, completing the login process without ever challenging the user to prove his or her identity. Also, the user may have stepped away from their computer, and neglected to lock their session.

As a result, it becomes quite complex to perform attribution, as actions performed on a user's account may not have been performed by the user owning the account. Again, one sees how the problem of attribution falls outside of the realm of pure digital forensics and instead requires additional evidence sources such as employee access card location data or camera footage, to name some examples.

However, given the risk of single sign-on, or the OS password being compromised, one concludes that the reference monitor should not rely on the authentication performed by the OS. Instead, although attribution will still not be possible, the likelihood of proper authentication is greatly increased if the reference monitor performs its own authentication. The exact method of such authentication is not relevant for this discussion, as long as the user is challenged to prove his or her identity to the reference monitor whenever a file edit commences. Furthermore, the reference monitor should retain all such user authentication information, together with all other relevant information.

In this section, the behaviours required of the reference monitor to be able to successfully collect the minimum evidence required have been noted. Now that it has been established that event reconstruction requires the use of a reference monitor to collect the evidence, the next step is that of protecting the collected evidence in such a manner that this evidence is not also susceptible to tampering. The next section therefore focuses on storing and protecting the collected evidence.

### **7.2.3 Storing and Protecting the Collected Evidence**

The purpose of this chapter is to develop a model to enable the reconstruction of events relating to unauthorised modification of an XML accounting data file. The model is tasked to collect evidence and store it in such a manner it can be recalled in order to successfully reconstruct the events leading to the data irregularity being discovered.

In the previous section, the strategy for collection of evidence was discussed, noting which evidence should be stored and why. However, in much the same way as XML data, the stored evidence is also vulnerable to unauthorised modification and tampering. Such evidence would not allow for correct reconstruction of events as per the problem statement in Section 1.3.2. This section is therefore dedicated to discussing how the evidence should be stored and protected.

The first important consideration is determining what the evidence is comprised of, so that a suitable storage and evidence protection strategy may be implemented. Evidence regarding user actions (such as modifying a file) is typically written to a log file [11]. Babbin et al [11] note that log files contain entries with two basic information fields, namely: the event detail field and the timestamp field.

Evidence stored by the reference monitor is therefore particularly suited to being stored in a log file. As mentioned before, the log file will therefore contain the following event details (fields):

- timestamp information;
- the specific change made to the file; and



- the username of the user who was authenticated to the reference monitor at the time of the change being made.

Now that the format and content of the evidence has been secured, the next step in providing protection against tampering is storing the evidence in a secure manner. The evidentiary log can be stored in two ways: 1) as part of the main XML data file by using a dedicated schema forming part of the document or 2) separately from the XML document in a dedicated file.

Storing the evidence within the XML file has several disadvantages and is therefore not suitable. Firstly, it increases the size of the XML data file unnecessarily. Secondly, storing the evidence within the XML data file allows any user with access to the XML data file to access and potentially modify the stored evidence (if the evidence is not considered part of the document that is hashed when signed).

Instead, the evidence log should be stored in a dedicated file outside of the XML data. This evidence log should be secured in such a manner that tampering with the evidence is not possible. However, unlike the XML accounting data, the log evidence is not required to be stored in human readable form. As per Garfinkel [60], a simple file encryption scheme would sufficiently deter tampering.

In the next section, the requirements for the XML Accounting Trail are reviewed and an overview of the operation of the model is provided.

### **7.3 Assembling the Pieces — An Overview of the XML Accounting Trail Model**

Up until this point, this chapter focused on the requirements for a model to enable reconstruction of the events leading up to a data irregularity being discovered. Requirements were derived for the types of evidence required, the process of collecting such evidence was established and a mechanism to protect the collected evidence was put in place.

In this section all these building blocks will be assembled in order to create a model responsible collecting the required evidence regarding modifications of the XML accounting data. This model is called the XML Accounting Trail model. The section will then conclude by discussing the operation of such a model.

As noted in Section 7.2.1, real-time logging of evidence is required. In addition, the real-time logging should be performed by means of a dedicated reference monitor, acting as a proxy for all editing operations on the XML accounting file.

Furthermore, evidence pertaining to the following should be collected and stored securely by the reference monitor:

- What was modified?
- When was it modified?
- Who modified it?

The structure of the XML Accounting Trail Model will now be described using the analogy of building a house.

The first step in building a house is establishing a firm foundation, on which the house may stand. This foundation, on which the entire model rests, is that of the reference monitor,

acting as a third party tool to gather information in real-time. The reference monitor forms the basis for the gathering of information and is the single point of access through which all edit operations of the XML file must take place. It, together with the modules it interacts with, is responsible for collecting all the evidence and storing it safely.

Building upon the foundation of the reference monitor, the walls of the house are comprised of a version control system (VCS). The VCS is responsible for providing a historical view of the file under investigation. Specifically, it collects detail on what was modified, coupled with a timestamp data showing the history of the modifications and how the XML accounting data changes over time. Therefore, the version control system also collects and stores the timestamp of each modification.

The third component of the model, responsible for interacting with the VCS, is the authentication mechanism. This mechanism can be seen as the ‘door’ of the house which controls access to entire model. It is responsible for collecting evidence regarding who modified the XML accounting data. The VCS therefore becomes the core component responsible for storing and re-assembling all evidence with regards to the modification of the XML accounting data.

Finally, every house needs a roof to protect the occupants from the elements. In the same way, the roof of tamper-protection protects the XML accounting data file from modification by a means other than the reference monitor. The tamper-protection is comprised of a digital signature belonging to the reference monitor and a hash of the contents of the XML file, appended to the XML accounting data file. This process is described in detail in Section 7.2.3.

The basic components of the model and how they fit together is illustrated in Figure 7.2.

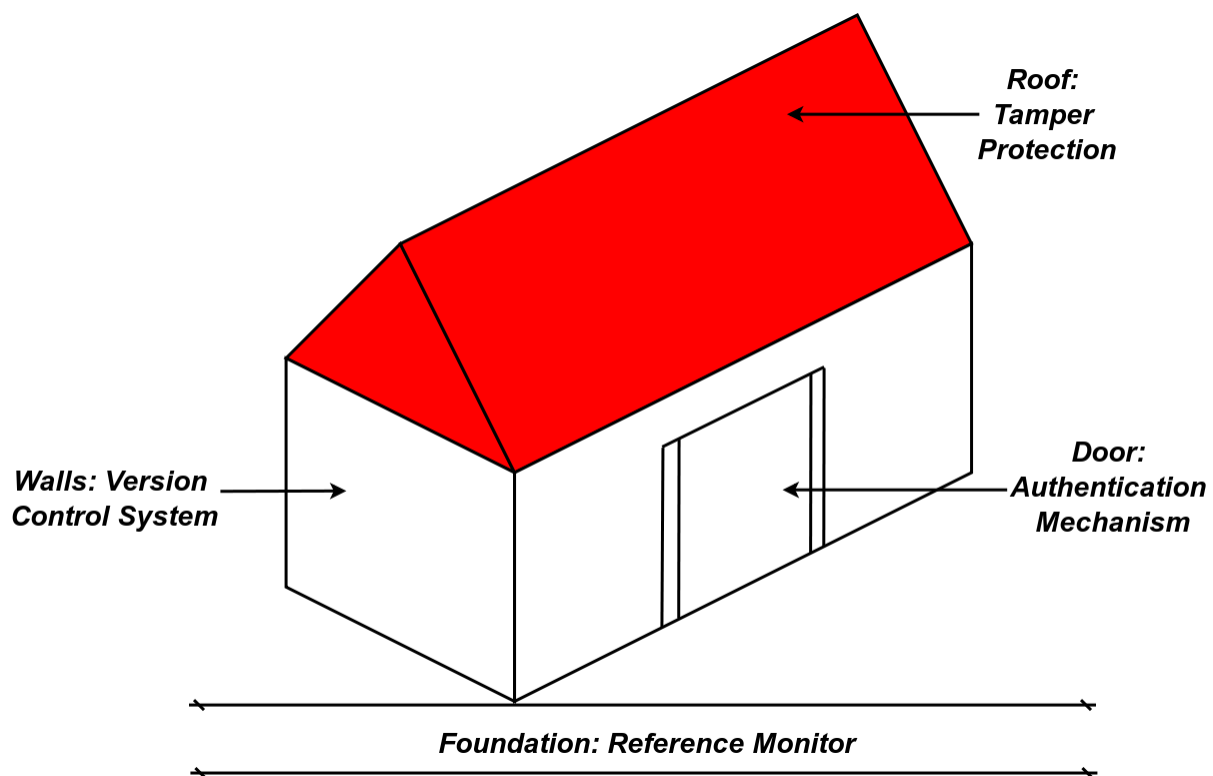


Figure 7.2: The basic components that combine to form the XML Accounting Trail model.

Now that the construction of the model is established, it is necessary to provide an overview

of the operation of the XML Accounting Trail model. An overview of the operation of the model is provided in Figure 7.3.

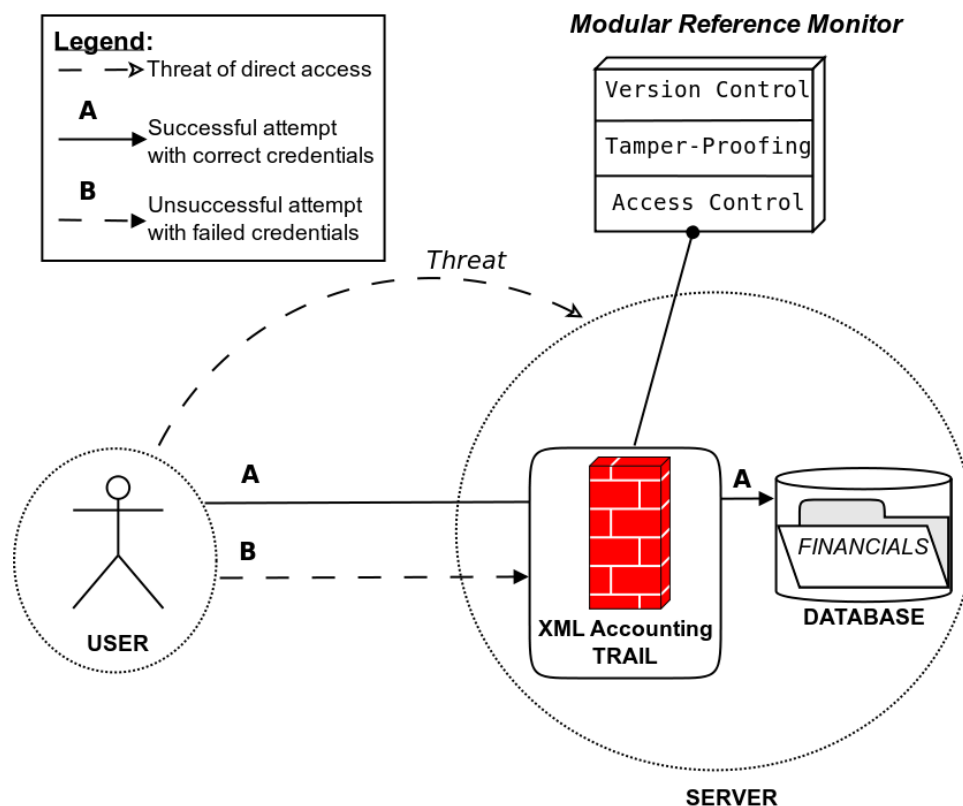


Figure 7.3: The operation of the XML Accounting Trail Model.

The model is invoked when a user attempts to open the XML accounting file by means of an authorised editing program. This program sends a request to the reference monitor. The reference monitor responds by requesting authorisation from the user.

If the incorrect authentication is supplied, the modification request is rejected (Action B in Figure 7.3). In the case where the correct authentication is supplied, the reference monitor continues to open the file (Action A in Figure 7.3). The following step is for the reference monitor to verify the integrity of the XML accounting file. Using its public key, the reference monitor decrypts the digital signature and obtains the hash file of the XML accounting document. It proceeds to re-hash the file, and compare it with the hash obtained from the digital signature. If the values match, access is allowed for the user to view and edit the file. Should these values not match or should the digital signature be absent, the appropriate person(s) are alerted to external tampering of the file by means of the pre-set alert triggers built into the reference monitor. The reference monitor provides the functionality for the system administrator to roll back to the last authorised copy of the XML accounting file and continue with the editing operation should the functionality be required.

Should the user then modify the file and save it, the version control system calculates the changes between the new version (as saved by the user) and the previous version. Next, the reference monitor obtains the current timestamp, as described in Section 7.2.2. The changes

between the versions are stored, along with the timestamp and user information, and the file is saved. Upon writing the new file, the reference monitor creates a new hash of the file, combines it with the private key of the reference monitor and applies the new signature to the XML accounting file.

In the case where the user attempts to directly modify the file (Action labelled ‘Threat’ in Figure 7.3), the change in the content of the file will cause the file to no longer match the hash contained in the digital signature. This cannot be circumvented, because even in the case where the digital signature is removed, the reference monitor will trigger an alert when the file is opened again.

This process of restricting access via the reference monitor is illustrated in Figure 7.4.

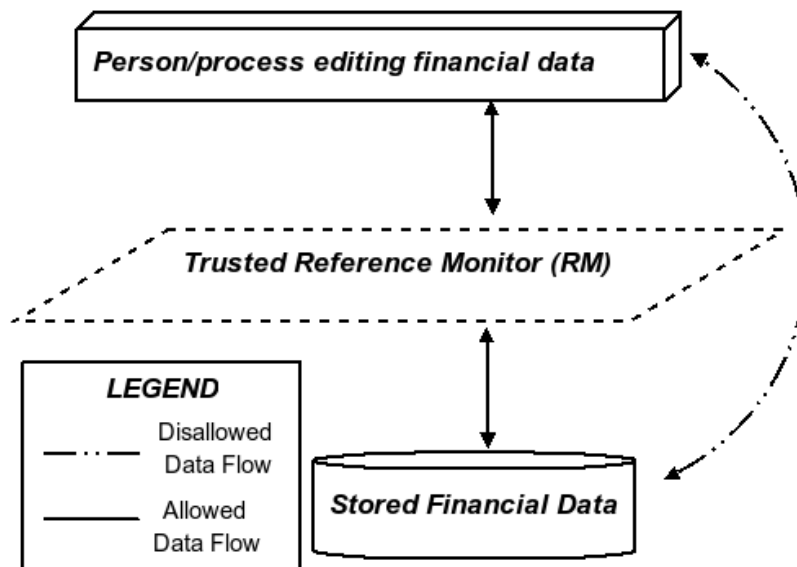


Figure 7.4: The role of the reference monitor in the XML Accounting Trail Model.

## 7.4 Conclusion

The problem statement in Section 1.3.2 mentions the reconstruction of events leading to data irregularities. As mentioned in Section 6.3.2, merely pointing out the data irregularities is not sufficient in addressing the evidence needed to reconstruct the events. Instead, instrumentation is required to specifically record the relevant evidence so that it is available in order to assist with reconstruction when necessary.

This chapter proposed the XML Accounting Trail Model as a means of addressing the above need. The discussion was initiated by a brief overview of the XML language and its main characteristics in order to remind the reader of the basic functioning of XML. This basic functioning was then used to outline the risks posed by the human readability requirement of XML and illustrating the ease with which unauthorised modification of the XML content (especially in sensitive environments, like accounting data) can be committed.

Subsequently, the evidence required for event reconstruction was examined and it was noted that a minimum set of evidence pertaining to the detail of what was changed, the time the changes were made and the identity of the user making the changes was required. Next, the process of obtaining evidence was discussed. It was noted that real-time logging of evidence was required and it was determined that the use of third party software capturing data in real-time by means of a dedicated proxy, referred to as a ‘reference monitor’, would be best suited to capturing XML modification data.

Despite the use of a reference monitor, it was observed that XML data remains vulnerable to direct modification resulting in the bypassing of the reference monitor. The combination of a digital signature and a watermark was proposed to ensure the integrity of the XML accounting data and to enforce the reference monitor as a single point of access to the data.

Subsequently, the actual process of collecting the evidence was discussed. This was broken down into three sections, based on the evidentiary requirements established in Section 7.2, namely:

- Collecting evidence showing ‘What was modified?’;
- Collecting evidence showing ‘When was it modified?’; and
- Collecting evidence showing ‘Who modified it?’.

From the above, it was concluded that a version control system was suitable for collecting evidence regarding what was modified. It was further noted that a reliable timestamp should be obtained in order to determine when was it modified, and an algorithm to obtain such a timestamp was proposed. Finally, the concept of attribution was introduced in order to determine who modified the data, and it was noted that an additional authentication mechanism would be required to establish the identity of the user responsible for modifying the file.

After establishing which evidence to collect, it was observed that the evidence should be stored in such a way that it is protected from tampering. A section discussing the storage and protection of the evidence was therefore introduced and it was found that the evidence should be stored as a log file, external to the XML accounting data and should be encrypted to prevent tampering.

The chapter concluded by combining all the building blocks above and creating the XML Accounting Trail model. The final section of the chapter provided an overview of the model itself, discussing the different elements of the model and the interaction of these elements to attain the goal of minimising the risk of unauthorised modification of XML data files. The operation of the model was also discussed, in addition to the practical application of the ways in which forensic readiness and tamper-proofing of the XML accounting data were established.



# Chapter 8

## Conclusion

XML has become ubiquitous for the purposes of sharing data. This is especially true in the world of business, where XML is increasingly used to store and share financial and accounting data. When considering financial data stored in XML format, it is important to note that XML is required to store marked up data in a human-readable, text-based format. XML data is therefore easy to make sense of and interpret.

However, together with human readability also comes a serious weakness. Due to its human readability, XML files can easily be modified by manipulating the file content. This is as a result of the data being stored in plain-text as well as the ease with which the file structure can be understood.

Financial information is typically very sensitive in nature and is usually most at risk of being misappropriated for financial gain. Traditionally, a person with access to the financial accounting system is able to post entries that may lead to personal financial gain, commonly known as fraud. Such accounting systems are typically well protected against such actions by various controls enforcing accounting rules, checks and balances, such as double-entry accounting, approval levels for transactions larger than a specific monetary threshold, and so on.

However, in the case of XML financial data, although these controls may be operational within the financial accounting software system, all controls enforcing accounting rules, checks and balances can be circumvented by directly accessing an XML financial data file and directly modifying its contents.

### 8.1 Problem

Detecting whether XML financial data has been modified in an unauthorised manner is extremely complex. In the case of modification in a manner other than with the accounting software responsible for updating and maintaining, the modification will typically not be detected. Only in the extreme case where the modification was severe enough to violate an XML rule, causing the accounting software package not to be able to read the file, will direct modification to the XML data file be detected.

In the event of unauthorised modification using the financial accounting package responsible for creating the XML file (in other words direct modification by editing the XML file using a text editor did not occur), such modifications will typically not be detected. The only way in which such a modification may be detected is if the modification affects a particular external

control, such as a monthly reconciliation of bank details from those recorded the system against the verified bank details of vendors.

This introduces one of the key problems with XML financial data. Determining whether an XML financial data file has been modified in an unauthorised manner is extremely complex. Furthermore, testing for such unauthorised modifications by means of a manual reconciliation of the file content is an extremely complex and time consuming operation, which is not practically feasible. In essence, one is searching for a very specific piece of information, which may or may not exist, in a large body of data. This problem is known as the large dataset problem.

Furthermore, in the event that direct modification did take place, it is of high importance to determine the extent of the modification and the details surrounding the actions in order to quantify the potential misstatement and enforce adequate protection against such a modification in future.

In essence the problem can be stated as the following question. *“Upon presentation of a set of XML financial data, how can one determine whether data has been tampered with, and reconstruct the past events so that the nature of tampering can be determined?”*

The next section will discuss the method employed in addressing the above problem.

## 8.2 Method and Results

In the previous section, it was noted that the problem involved a means to detect data tampering after it has occurred. Also, the problem statement involves a means to attempt the reconstruction of the events leading up to the final data irregularity which was detected.

To solve the problem of detecting potential tampering with XML files containing financial data, the work in this text proposed an approach which operates in an after-the-fact fashion. It was suggested that a grammar should be created to describe the financial XML data contained in the file which is being checked for potential tampering. The grammar should include both the required structures as well as specific targets for known modification indicators and/or patterns consistent with fraud. A compiler could then be used to analyse the XML financial data file based on the rules specified in the grammar, allowing one to determine whether the file is normal and does not need any further investigation (the data file conforms to the rules applicable and no signs of tampering is present in the data file) or whether the data file is consistent with known patterns of tampering and/or fraud.

It was found that the analysis stage of compiler theory is exceptionally well-suited to matching patterns in large data sets, which is the exact challenge faced when trying to look for data irregularities in an XML financial data file. Due to the rigour and structure of XML, it was possible to define rules describing the correct composition of such a file, and using the rules of accounting and experience in financial investigations, it was also possible for an investigator to set up grammar rules checking for the application of accounting rules as well as checking for known traces of specific types of fraud. Furthermore, it was found that by making use of the analysis phase presented in compiler theory, that a compiler is able to parse such grammars and determine whether any of the patterns requiring further scrutiny exists in any given file.

A benefit of this approach is providing the investigator with more information at his or her disposal, enabling the investigator to more efficiently triage and more accurately decide whether a given XML file containing financial data is worthy of further investigation. Also, using known patterns of data irregularities associated with fraud, the investigator can prioritise



which evidence to process and investigate first.

This is coupled with a further, less direct benefit namely a reduction in cost. This is due to the investigator being able to more easily determine which evidence should be investigated first, leading to a higher investigation success rate and reducing investigation time. Also, due to the fact that an investigator can now triage with more confidence, investigation time is further reduced, leading to additional cost benefit.

Although this approach was able to assist an investigator in determining whether XML financial data has been tampered with, reconstruction of the events leading up to the detection of the data tampering is not possible using this approach in isolation. This is a direct result of the available information not being sufficient to reconstruct the events leading up to the discovery of data tampering.

Therefore, in order to address the event reconstruction aspect of the problem, the work in this text also constructed a model in order to collect the evidence necessary for event reconstruction. The XML Accounting Trail model is composed of a reference monitor (or proxy) which is responsible for intercepting all write requests to the XML data file containing financial data. It then updates the file, whilst also recording additional information using various other means, such as version control. The following additional evidence is retained: the username of the user making the change, the date stamp and time stamp of the moment the change was made and a difference file highlighting the difference between the updated file and version prior to the update.

As a result, it was found that it is possible to secure XML files from being directly modified and their content being usable without violating the human readability requirement. This was due to the use of digital signatures and the accounting data file hashes in order to ensure that only the reference monitor can modify the file.

Also, because of the additional information being captured, forensic readiness principles are applied when the model is deployed, enabling an investigation to take place with minimal additional investigation required as historic evidence of all XML financial data file modifications already exists.

The XML Accounting Trail model poses a number of benefits. Firstly, by using the model, it is possible to have a historical view of all changes made to the data. Also, it is possible to isolate a specific data irregularity and remove it from the body of data.

A second benefit is that of drastically reducing investigation time and thereby, investigation cost. By having evidence on hand regarding the specific modifications that were made, the time and dates of these modifications and the user account responsible for making the changes, the basic investigative questions can be answered without any further investigation.

The final benefit of the XML Accounting Trail model is that of decreasing the likelihood of inducing data irregularities. Due to the XML file becoming tainted if modified outside the reference monitor, the incentive for direct modification of the file is significantly reduced. Furthermore, knowing that evidence about every modification is collected by the reference monitor should deter a large number of attempts to modify data in an unauthorised manner.

The next section reflects on the potential shortcomings and areas of concern around the solutions proposed above.

## 8.3 Self Evaluation

Despite the author's best efforts, areas in research always exist where answers may not be clear or the proposed solution only leads to more questions. Therefore, it is of prime importance to step back and reflect on the work suggested so that shortcomings and areas for future work can be identified.

It is possible to mathematically prove that a compiler will compile input only if it matches the grammar that specifies the legal input [5, 103, 145].

Using the above argument, according to set theory, a compiler should only be able to compile input that matches the set of allowable tokens and productions, thereby ensuring that input which successfully compiled is lexically, syntactically and semantically correct.

Considering this argument, the proposed approach for detecting data irregularities using compilers shows great promise. Although the proposed approach seems to be mathematically provable, the approach has not been tested extensively in a real-world application. Mention must be made of the prototype that was constructed by Kotze and Olivier [91] during the early research phases of this dissertation. However, this prototype merely served to prove the concept of detecting irregularities using the analysis stage of a compiler and only extended to a very simple definition of XML accounting data. It is therefore a key consideration that future work should apply this approach in a real-world scenario in order to determine whether the proposed approach is feasible.

Furthermore, the successful detection of data irregularities is very much predicated on the quality of the grammar that defines the input to the compiler. If the investigator does not possess the knowledge to correctly specify a grammar that deals with the complete set of allowable input for a specific XML accounting data file, the compiler will not be able to correctly triage the input. Instead, the extent of the triage and certainty of whether a specific pattern of data irregularities exist in an XML accounting data file, will be limited by an investigator's ability to correctly specify and use a grammar for the XML file analysed.

In scenarios where first responders are not well-trained or not able to successfully apply the concept of a grammar, the detection of data irregularities using a compiler will be stunted and the benefits derived from such an application would be severely limited. This concern needs to be addressed by future work and will be noted in the "Open Problems" section below.

Considering the XML Accounting Trail model and the use of this model to reconstruct data modification events leading to the discovery of a data irregularity using a compiler, one notes that several weaknesses also exist with the application of this model.

The successful use of the model operates on the foundational assumption that the reference monitor should always be invoked whenever a change is made to the XML accounting data. This creates a potential bottle-neck which may carry a severe performance penalty if the XML Accounting Trail model is not implemented to be as small and efficient as possible.

Furthermore, the model operates on the assumption that the recipient of XML accounting data would, on receipt of the data, verify the content of the data and the signature to ensure that data tampering did not occur. In the event that this step is not followed, or in the event that verification is not possible (e.g. the web server hosting the public key of the reference monitor used is down), the XML data file again becomes vulnerable to tampering, negating any benefits posed by the XML Accounting Trail model.

Finally, the possibility of compromise of the reference monitor should be considered. In the event that the reference monitor becomes compromised, the entire XML Accounting Trail

model becomes invalid as direct modification to the XML data once again becomes possible. Compromise of the reference monitor is possible by either a direct hacking attempt which may be able to extract the private key from the executable comprising the reference monitor, or an attacker may simply clone the environment executing the reference monitor.

These considerations have not been addressed in the work presented in this dissertation and have been left for future work, listed in the “Open Problems” section below.

## 8.4 Open Problems

Whilst documenting the proposed approach for detecting data irregularities using compiler theory and considering the XML Accounting Trail model and how it allows for reconstruction of events, new avenues of thought were introduced regarding additional work that can be completed in order to further strengthen the work proposed in this dissertation. The purpose of this section is therefore to emphasise these avenues for further exploration which were not directly addressed in the work performed in the preceding chapters.

The first item for further consideration is that of grammar development. One criticism that can be levelled at the use of compilers to detect data irregularities, is that of the use of grammars. The quality of grammars used to detect data irregularities as well as the expertise required by investigators to draw up such grammars may provide cumbersome. Future work around the creation of template grammars for different implementations of XML accounting data files would lessen the responsibility of the investigator as he or she would not need to derive their own grammars. Furthermore, if the templates could also include patterns for data irregularities known to indicate misstatement, investigators could also use such templates to prioritise which files should be investigated first. Investigators will then be able to re-use proven grammars and do not need to be experts at grammar construction in order to triage evidence involving XML accounting data.

A second item for consideration is that of improving and expanding the use of errors and error handling strategies in detecting and classifying data irregularities. Specifically, the use of semantic errors can be improved and expanded. Potential exists in implementing strategies to analyse trends in errors, and use such trends to predict malicious intent behind data modifications. Semantic error testing can be extended to test for trends, for example: testing for large numbers of very small transactions (the salami attack) or testing for statistical significance of numbers using methods like Benford analysis.

Also, further work can be done on the use of error correcting strategies like global correction. If global correction could be implemented in such a way that it could tell the data inspector which changes had led to a particular error being noted in the input, the data inspector could start with evidence reconstruction in the absence of additional sources of evidence as suggested by the XML Accounting Trail model.

Thirdly, the security of the XML Accounting Trail model can be vastly improved. The use of anti-forensic and anti-hacking techniques could be investigated in order to protect the extraction of the private key from the XML Accounting Trail reference monitor. Should this key be compromised, an attacker would be able to bypass the evidence collection capabilities of the XML Accounting Trail completely.

Also, further work could be done in order to set up a secure server which XML Accounting Trail reference monitors could use to authenticate against on a regular basis, using a unique

key. That way, if a reference monitor is somehow replicated, it would be able to detect this fact and cease to operate.

Finally, now that the foundational work has been completed, further proof of concept and/or real world implementations of both the XML Accounting Trail model and the proposed approach of detecting data irregularities by means of compiler theory will prove very helpful. These will provide detail as to the real-world challenges of the proposed work as well as potential additional shortcomings which were not obvious from the more theoretical approach followed in this dissertation.

# Glossary of Abbreviations

<b>ACFE</b>	Association of Certified Fraud Examiners
<b>AM</b>	Ante meridiem (before noon)
<b>ANSI</b>	American National Standards Institute
<b>BASIC</b>	Beginner's All-purpose Symbolic Instruction Code
<b>BI</b>	Business Intelligence
<b>BISON</b>	GNU Project parser generator
<b>CAS</b>	Continuous Audit System
<b>CEO</b>	Chief Executive Officer
<b>COBIT</b>	Control Objectives for Information and related Technology
<b>COBOL</b>	Common Business Oriented Language
<b>DFE</b>	Digital Forensic Evidence
<b>DNA</b>	Deoxyribonucleic Acid
<b>DTD</b>	Data Type Definition
<b>ERP</b>	Enterprise Resource Planning
<b>FBI</b>	Federal Bureau of Investigation
<b>FIX</b>	Financial Information Exchange Protocol
<b>FLEX</b>	Fast Lexical Analyser Generator
<b>FORTRAN</b>	Formula Translation Language
<b>FTK</b>	Forensic Tool Kit
<b>FTP</b>	File Transfer Protocol
<b>GL</b>	General Ledger
<b>GNU</b>	GNU's Not Unix
<b>HIPAA</b>	Health Insurance Portability and Accountability Act
<b>HMAC</b>	Hashes and Message Authentication Codes
<b>HR</b>	Human Resources
<b>HTML</b>	Hypertext Markup Language
<b>IBM</b>	International Business Machines
<b>ICT</b>	Information and Communications Technology
<b>ID</b>	Identification
<b>IDIP</b>	Integrated Digital Investigation Process
<b>IFX</b>	Interactive Financial Exchange
<b>IP</b>	Internet Protocol
<b>IR</b>	Intermediate Representation
<b>ISACA</b>	Information Systems Audit and Control Association
<b>ISP</b>	Internet Service Provider

*Continued on next page*

*Continued from previous page*

<b>IT</b>	Information Technology
<b>LISP</b>	LISt Processing (Programming Language)
<b>MDDL</b>	Market Data Definition Language
<b>NAT</b>	Network Address Translation
<b>NTP</b>	Network Time Protocol
<b>OASIS</b>	Organization for the Advancement of Structured Information Standards
<b>OFC</b>	Open Financial Connectivity format
<b>OFX</b>	Open Financial Exchange
<b>OS</b>	Operating System
<b>PC</b>	Personal Computer
<b>PDA</b>	Personal Digital Assistant
<b>PL</b>	Processing Language
<b>RFC</b>	Request for Comment
<b>SAP</b>	Systemanalyse und Programmentwicklung (System Analysis and Program Development) — A company known for its ERP products.
<b>SCSIM</b>	Scientific Crime Scene Investigation Model
<b>SEC</b>	Securities and Exchange Commission
<b>SGML</b>	Standard Generalised Markup Language
<b>SIM</b>	Subscriber Identity Module
<b>SSL</b>	Secure Sockets Layer
<b>SWIFT</b>	Society for Worldwide Interbank Financial Telecommunication
<b>TTSP</b>	Trusted Time-source Provider
<b>UBL</b>	Universal Business Language
<b>UNIVAC</b>	UNIVersal Automatic Computer
<b>UNIX</b>	UNIplicated Information and Computing Service — a computer operating system.
<b>US</b>	United States
<b>USD</b>	United States Dollar
<b>VCS</b>	Version Control System
<b>XBRL</b>	Extensible Business Reporting Language
<b>XML</b>	Extensible Markup Language
<b>YACC</b>	Yet Another Compiler Compiler

# Bibliography

- [1] AccessData Corporation. FTK Mobile Phone Examiner. Online, May 2010. Accessed on 27 April 2010, <http://www.accessdata.com/mobilephoneexaminer.html>.
- [2] U. S. G. Accountability, editor. *Aviation Safety: Efforts to Implement Flight Operational Quality Assurance Programs*. BiblioGov, 1st edition, Jan. 2011.
- [3] N. Aggarwal, Q. Dai, and E. A. Walden. Do Markets Prefer Open or Proprietary Standards for XML Standardization? An Event Study. *International Journal of Electronic Commerce* , 11(1), 2006.
- [4] A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers: Principles, Techniques, and Tools*. Addison Wesley, 2nd edition, Aug. 2006.
- [5] A. V. Aho and J. D. Ullman. *The theory of parsing, translation, and compiling*. Prentice-Hall, 1972.
- [6] W. Albrecht and C. Albrecht. *Fraud Examination*. Cengage Learning, 4th edition, Feb. 2011.
- [7] M. Anandarajan, A. Anandarajan, and C. A. Srinivasan. *Business Intelligence Techniques: A Perspective from Accounting and Finance*. Springer, 1st edition, 2004.
- [8] F. Arciniegas. *XML Developer's Guide*. McGraw-Hill Companies, 1st edition, Dec. 2001.
- [9] Association of Certified Fraud Examiners. Report to the Nations on Occupational Fraud and Abuse (2012). Online, 2012.  
[http://www.acfe.com/uploadedFiles/ACFE\\_Website/Content/rtnn/2012-report-to-nations.pdf](http://www.acfe.com/uploadedFiles/ACFE_Website/Content/rtnn/2012-report-to-nations.pdf).
- [10] P. K. Austin and J. Sallabank, editors. *The Cambridge Handbook of Endangered Languages*. Cambridge University Press, 1st edition, Apr. 2011.
- [11] J. Babbin et al. *Security Log Management: Identifying Patterns in the Chaos*. Syngress Publishing, Inc, 1st edition, 2006.
- [12] D. K. Barry. Finance XML. Online, 2012. Available at  
[http://www.service-architecture.com/xml/articles/finance\\_xml.html](http://www.service-architecture.com/xml/articles/finance_xml.html).
- [13] V. Baryamureeba and F. Tushabe. The Enhanced Digital Investigation Process Model. *Digital Forensics Research Workshop*, Aug. 2004.
- [14] B. Berg. *Criminal Investigation*. McGraw-Hill, 4th edition, July 2007.
- [15] P. Bergman and S. Berman-Barrett. *The Criminal Law Handbook: Know Your Rights, Survive the System*. Nolo, 1st edition, Sept. 2009.
- [16] T. Berners-Lee, J. Hendler, and O. Lasilla. The Semantic Web. *Scientific American*, May 2001.
- [17] T. Berners-Lee, N. Shadbolt, and W. Hall. The Semantic Web Revisited. *IEEE Intelligent Systems*, pages 96–101, May 2006.
- [18] R. Bhaskar. State and local law enforcement is not ready for a cyber Katrina. *Communications of the ACM*, 49(2):pages 81–83, Feb. 2006.
- [19] G. Bibel. *"Beyond the Black Box"*. Johns Hopkins University Press, 1st edition, Dec. 2007.
- [20] H. Bidgoli, editor. *The Internet Encyclopaedia*. Wiley and Sons, 2nd edition, Nov. 2003.
- [21] H. Bidgoli. *Handbook of Information Security*, volume three. Wiley and Sons, 1st edition, Dec. 2005.
- [22] E. Bonsón. The role of XBRL in Europe. *The International Journal of Digital Accounting Research*, 1(2):pages 101–110, 2001.

- [23] E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Springer, Oct. 2001.
- [24] J. Bosak and T. McGrath. OASIS. Online, 2012. Available at [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=ubl](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ubl).
- [25] R. Boyd, P. Gasper, and J. D. Trout. *The Philosophy of Science*. A Bradford Book, 1st edition, June 1991.
- [26] T. Bray, J. Paoli, and C. M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0. Technical report, W3C, Feb. 1998. Available at <http://www.w3.org/TR/REC-xml>.
- [27] A. Brown. Quashed conviction gives new life, Dec. 2009. Available at <http://www.sunshinecoastdaily.com.au/story/2009/12/28/life-begins-again-after-conviction-quashed/>.
- [28] J. F. Brown, K. S. Obenski, and T. R. Osborn. *Forensic Engineering Reconstruction of Accidents*. Charles C Thomas Pub Ltd, 2nd edition, Jan. 2003.
- [29] R. D. Burnett, M. Friedman, and U. S. Murthy. Financial Reports: Why You Need XBRL. *The Journal of Corporate Accounting & Finance*, pages 33–40, July 2006.
- [30] B. Carrier. Open Source Digital Forensic Tools: The Legal Argument. Technical report, @stake Technical Report, 2002.
- [31] B. Carrier and E. H. Spafford. Getting physical with the Digital Investigation Process. *International Journal of Digital Evidence*, 2(2), 2003.
- [32] B. D. Carrier. *A Hypothesis-based Approach to Digital Forensic Investigations*. PhD thesis, Purdue University, May 2006.
- [33] B. D. Carrier and E. H. Spafford. Denying Event Reconstruction of Digital Crime Scenes. *Journal of Forensic Science*, 49(6), Nov. 2004.
- [34] E. Casey. Error, Uncertainty, and Loss in Digital Evidence. *International Journal of Digital Evidence*, 1(2), 2002.
- [35] E. Casey. *Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet*. Academic Press, 2nd edition, 2004.
- [36] N. P. Chapman. *LR Parsing: Theory and Practice*. Cambridge University Press, Jan. 1988.
- [37] A. N. K. Chen, R. C. LaBrie, and B. B. M. Shao. An XML Adoption Framework for Electronic Business. *Journal of Electronic Commerce Research*, 4(1), 2003.
- [38] E. K. Cheng and A. H. Yoon. Does Frye or Daubert Matter? A study of scientific admissibility standards. *Virginia Law Review*, 91(2), Apr. 2005.
- [39] W. J. Chisum and B. E. Turvey. *Crime Reconstruction*. Academic Press, 2nd edition, Aug. 2011.
- [40] D. M. Clarke. *Descartes' Philosophy of Science (Studies in intellectual history)*. Manchester University Press, Oct. 1982.
- [41] F. Cohen. *Digital Forensic Evidence Examination*. Fred Cohen & Associates, 2nd edition, 2010.
- [42] S. A. Cole. It's the testimony, stupid. *The National Law Journal*, 30(22), Dec. 2010.
- [43] C. S. Collberg. Watermarking, Tamper-Proofing, and Obfuscation - Tools for Software Protection. *IEEE Transactions on Software Engineering*, 28(8):pages 735–746, Aug. 2002.
- [44] K. D. Cooper and L. Torczon. *Engineering a Compiler*. Elsevier Science, 2004.
- [45] A. Corell. *Real-Time Systems: Modelling, Design, and Applications*. World Scientific Publishing Company, Mar. 2007.
- [46] V. V. Das. *Compiler Design Using FLEX and YACC*. Prentice-Hall, Aug. 2007.
- [47] R. Debreceeny and G. L. Gray. The production and use of semantically rich accounting reports on the Internet: XML and XBRL. *International Journal of Accounting Information Systems*, 2:pages 47–74, 2001.
- [48] A. Deshmukh. *Digital Accounting: The Effects of the Internet and ERP on Accounting*. Idea Group Publishing, 1st edition, Dec. 2005.
- [49] Dictionary.com. Dictionary.com. Online, 2013. <http://dictionary.reference.com/browse/epistemology>.



- [50] C. Durtschi, W. Hillison, and C. Pacini. The Effective Use of Benford's Law to Assist in Detecting Fraud in Accounting Data. *Journal of Forensic Accounting*, V:pages 17–34, 2004.
- [51] L. Dykes and E. Tittel. *XML For Dummies*. For Dummies, 4th edition, May 2005.
- [52] A. Edmunds and A. Morris. The problem of information overload in business organisations: a review of the literature. *International Journal of Information Management*, 20:pages 17–28, 2000.
- [53] European Council. Council Regulation (EC) No 2157/2001 of 8 October 2001 on the Statute for a European company (SE). Technical report, EU Council, Oct. 2001. Available at <http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32001R2157:EN:NOT>.
- [54] M. Farrell. Daubert v. Merrell Dow Pharmaceuticals, Inc.: Epistemology and Legal Process. *Cardozo Law Review* 1994, 15(6-7):pages 2183–2217, Apr. 1994.
- [55] J. Feather. *The Information Society: A Study of Continuity and Change*. Library Association Publications Ltd, Oct. 2004.
- [56] Free On-Line Dictionary of Computing Terms. Free On-Line Dictionary of Computing. Online, Mar. 2001. Available at <http://foldoc.doc.ic.ac.uk/foldoc/Dictionary.gz>.
- [57] S. Garfinkel. *PGP: Pretty Good Privacy*. O'Reilly and Associates, 1st edition, Dec. 1994.
- [58] S. Garfinkel, P. Farrell, V. Roussev, and G. Dinolt. Bringing science to digital forensics with standardized forensic corpora. *Digital Investigation*, 6, 2009.
- [59] S. Garfinkel and J. J. Migletz. New XML-Based Files: Implications for Forensics. *Security & Privacy*, 2009.
- [60] S. Garfinkel and G. Spafford. *Web Security, Privacy and Commerce*. O'Reilly and Associates, 2nd edition, Jan. 2002.
- [61] I. Gartner. Gartner Reveals Five Business Intelligence Predictions for 2009 and Beyond. Online, Jan. 2009. Located at <http://www.gartner.com/newsroom/id/856714>.
- [62] V. Geroimenko. *Dictionary of XML technologies and the Semantic Web*, volume 1. Springer Verlag, 2004.
- [63] P. Gladychyev. *Formalising Event Reconstruction in Digital Investigations*. PhD thesis, University College Dublin, Aug. 2004.
- [64] R. J. Glushko, J. M. Tanenbaum, and B. Meltzer. An XML Framework for Agent-based E-Commerce. *Communications of the ACM*, 42(3):pages 106–114, Mar. 1999.
- [65] D. Gollman. *Computer Security*. Wiley and Sons, 2nd edition, 2005.
- [66] Government of the United States of America. Health Insurance Portability and Accountability Act of 1996. Online, Aug. 1996. <http://aspe.hhs.gov/admnsimp/pl104191.htm>.
- [67] Government of the United States of America. Gramm-Leach-Bliley Act. Online, Nov. 1999. Available at <http://www.ftc.gov/privacy/glbact/glbsub1.htm>.
- [68] Government of the United States of America. Public Law 107 - 204 - Sarbanes-Oxley Act of 2002. Online, 2002. Available at <http://www.gpo.gov/fdsys/pkg/PLAW-107publ204/content-detail.html>.
- [69] Government of the United States of America. The Federal Information Security Management Act. Online, 2002. Available at <http://csrc.nist.gov/drivers/documents/FISMA-final.pdf>.
- [70] M.-A. Grado-Caffaro and M. Grado-Caffaro. The Challenges that XML Faces. *IEEE Computer*, pages 15–18, Oct. 2001.
- [71] M. A. Hardy and A. Bryman. *Handbook of Data Analysis*. Sage Publications Ltd., May 2004.
- [72] Her Majesty's Government. Companies Act 2006. Online, Nov. 2006. Available at <http://www.legislation.gov.uk/ukpga/2006/46/contents>.
- [73] S. Hill. *Corporate Fraud: Prevention and Detection*. Bloomsbury Professional Ltd., 2nd edition, 2010.
- [74] C. Hoffman, C. Kurt, and R. J. Koreto. The XML Files. *Journal of Accountancy*, 187(5):pages 71–77, May 1999.

- [75] D. Hollander and C. M. Sperberg-McQueen. Happy Birthday, XML! Online, Feb. 2003. Available at <http://www.w3.org/2003/02/xml-at-5.html>.
- [76] C. Hosmer. Proving the integrity of digital evidence with time. *International Journal of Digital Evidence*, 1(1), 2002.
- [77] IFX Forum Inc. Interactive Financial eXchange Forum. Technical report, IFX Forum, 2012. Available at <http://www.ifxforum.org/standards/>.
- [78] Information Systems and Control Association Regulatory Board. IS Auditing Guideline, Computer Forensics, Document G28. Technical report, Information Systems and Control Association Regulatory Board, July 2004. Available at <http://faculty.ksu.edu.sa/73811/Documents/Auditing/COMPUTER%20FORENSIC.pdf>.
- [79] Innocence Project. Unreliable or Improper Forensic Science. Online, Aug. 2012. Available at <http://www.innocenceproject.org/understand/Unreliable-Limited-Science.php>.
- [80] O. G. Kakde. *Comprehensive Compiler Design*. Laxmi Publications, 1st edition, Dec. 2005.
- [81] H. Kaptein, H. Prakken, and B. Verheij. *Legal Evidence and Proof (Applied Legal Philosophy)*. Ashgate, 1st edition, Feb. 2013.
- [82] Kaspersky Lab ZAO. Kaspersky Anti-Virus for Linux File Server. Online, 2011. Available at <http://www.kaspersky.com/anti-virus-linux-file-server>.
- [83] A. V. D. M. Kayem and C. Meinel. *Theories and intricacies of information security problems*. Universitätsverlag Potsdam, 1st edition, mar 2013.
- [84] A. Keane. *The Modern Law of Evidence*. Oxford University Press, 7th edition, May 2008.
- [85] R. W. Keyes. The Impact of Moore's Law. *Solid State Circuits*, Sept. 1996.
- [86] K. A. A. Khan, C. Buisman, and C. Gosnell. *Principles of Evidence in International Criminal Justice*. Oxford University Press, 1st edition, Jan. 2010.
- [87] A. Kirkpatrick. The XML Files. *The Computer Bulletin*, pages 27–29, May 1998.
- [88] P. Kitcher. Kant's Philosophy of Science. *Midwest Studies In Philosophy*, 8(1):pages 387–407, Sept. 1983.
- [89] B. Knight. *Simpson's Forensic Medicine*. Arnold Publishers, 11th edition, 1997.
- [90] M. Kohn, J. Eloff, and M. Olivier. Framework for a Digital Forensic Investigation. In *Proceedings of the ISSA 2006 from Insight to Foresight Conference*, 2006.
- [91] D. Kotze and M. S. Olivier. Detecting XML Data Irregularities by Means of Lexical Analysis and Parsing. In J. Demergis, editor, *Proceedings of the 9th European Conference on Information Warfare and Security*, pages 151–159. Academic Conference Limited, Jan. 2010.
- [92] W. Kruse and J. G. Heiser. *Computer Forensics: Incident Response Essentials*. Addison Wesley, Oct. 2001.
- [93] S. S. Laurent. *XML - A Primer*. M&T Books, 2nd edition, 1999.
- [94] H. C. Lee, T. Palmbach, and M. T. Miller. *Henry Lee's Crime Scene Handbook*. Academic Press, 1st edition, July 2001.
- [95] D. Loshin. *Business Intelligence: The Savvy Manager's Guide (The Morgan Kaufmann Series on Business Intelligence)*. Morgan Kaufmann, 1st edition, 2009.
- [96] C.-S. Lu, editor. *Multimedia Security: Steganography and Digital Watermarking Techniques for Protection of Intellectual Property*. Idea Group Publishers, 1st edition, July 2004.
- [97] T. Lyons. Frye, Daubert and where do we go from here. *Rhode Island Bar Journal*, 1997.
- [98] R. MacMillan. XBRL financial reporting faces hurdles. Online, June 2009. Available at <http://mobile.reuters.com/article/technologyNews/idUSTRE55M4IM20090623>.
- [99] R. McKemmish. What is Forensic Computing? *Trends & issues in crime and criminal justice*, June 1999.
- [100] J. R. Meaney. From Frye to Daubert: Is a Pattern Unfolding. *Jurimetrics Journal*, 35:pages 191, 1995.
- [101] Merriam-Webster. The Free Merriam-Webster Dictionary. Online, Feb. 2013. <http://www.merriam-webster.com/>.

- [102] A. R. Meyer and M. J. Fischer. Economy of description by automata, grammars, and formal systems. *IEEE Annual Symposium on Foundations of Computer Science*, 54, 1971.
- [103] M. Meyers and M. Rogers. Computer Forensics: Meeting the Challenges of Scientific Evidence. *Research Advances in Digital Forensics*, 2005.
- [104] J. Mitchell. Quantitative science and the definition of measurement in psychology. *British Journal of Psychology*, 88(3):pages 355–358, Apr. 2011.
- [105] T. Moore. The Economics of Digital Forensics. *Fifth Workshop on the Economics of Information Security*, June 2006.
- [106] S. Muchnick. *Advanced Compiler Design and Implementation*. Morgan Kaufmann, 1st edition, Aug. 1997.
- [107] U. S. Murthy and S. M. Groomer. A continuous auditing web services model for XML-based accounting systems. *International Journal of Accounting Information Systems*, 5:pages 139–163, 2004.
- [108] National Institute of Justice. Electronic Crime Scene Investigation: A Guide for Law Enforcement. Online, July 2001. <http://www.ncjrs.gov/pdffiles1/nij/187736.pdf>.
- [109] National White Collar Crime Centre, Bureau of Justice Assistance, and Federal Bureau of Investigation. 2013 Internet Crime Report. Technical report, Federal Bureau of Investigation, 2013. Available at [http://www.ic3.gov/media/annualreport/2013\\_IC3Report.pdf](http://www.ic3.gov/media/annualreport/2013_IC3Report.pdf).
- [110] M. Nicola, I. Kogan, and B. Schiefer. *An XML Transaction Processing Benchmark*. ACM, June 2007.
- [111] P. H. Nidditch. *The philosophy of science*. Oxford readings in philosophy. Oxford University Press, 1981.
- [112] S. Niranen. *Open Information Management: Applications of Interconnectivity and Collaboration*. Information Science Reference, 1st edition, May 2009.
- [113] Office of the President of South Africa. Companies Act No 61 of 1973. Online, 2006. Available at [http://www.acts.co.za/company/companies\\_act.htm#companies\\_ac.htm](http://www.acts.co.za/company/companies_act.htm#companies_ac.htm).
- [114] M. S. Olivier. *Information Technology Research*. Van Schaik, 2nd edition, 2004.
- [115] M. S. Olivier. On Complex Crimes and Digital Forensics. In H. S. Venter, M. Coetzee, and M. Look, editors, *2012 Information Security for South Africa*. IEEE, 2012.
- [116] J. W. Osterburg and R. H. Ward. *Criminal Investigation: A Method for Reconstructing the Past*. Anderson Publishing, 6th edition, June 2010.
- [117] Oxford University Press, editor. *Oxford English Dictionary*. Oxford University Press, Mar. 2007. Available at <http://dictionary.oed.com>.
- [118] G. Palmer. A Road Map for Digital Forensic Research. Technical Report DTR-T0010-01, DFRWS, 2001.
- [119] G. Palmer. Forensic Analysis in the Digital World. *International Journal of Digital Evidence*, 1(1), 2002.
- [120] G. Pangalos and V. Katos. Information Assurance and Forensic Readiness. In *Next Generation Society Technological and Legal Issues: Third International Conference, e-Democracy 2009*, pages 181–187. Springer, Mar. 2010.
- [121] Paraben Corporation. Paraben Forensic Tools Catalogue. Technical report, Paraben Forensic Corporation, May 2010. Available at [http://www.paraben.com/catalog/product\\_info.php?products\\_id=484](http://www.paraben.com/catalog/product_info.php?products_id=484).
- [122] J. J. Parsons and D. Oja. *New Perspectives on Computer Concepts 2012: Comprehensive*. Course Technology, 14th edition, Feb. 2011.
- [123] Payment Card Industry Security Standards Council. Payment Card Industry Data Security Standard. Technical report, Payment Card Industry Security Standards Council, Oct. 2008. Available at [https://www.pcisecuritystandards.org/security\\_standards/download.html?id=pci\\_dss\\_v1-2.pdf](https://www.pcisecuritystandards.org/security_standards/download.html?id=pci_dss_v1-2.pdf).
- [124] C. P. Pfleeger and S. L. Pfleeger. *Security in Computing*. Prentice Hall, 3rd edition, 2003.

- [125] G. Picot. *Handbook of International Mergers and Acquisitions*. Palgrave Macmillan, Oct. 2002.
- [126] C. M. Pilato, B. Collins-Sussman, and B. W. Fitzpatrick. *Version Control with Subversion*. O'Reilly and Associates, 1st edition, June 2004.
- [127] R. Pinsker and S. Li. Costs and Benefits of XBRL Adoption: Early Evidence. *Communications of the ACM*, 51:pages 47–50, Mar. 2008.
- [128] K. Popper. *The Logic of Scientific Discovery*. Basic Books, 1959.
- [129] M. Power. *The Audit Society - Rituals of Verification*. Oxford University Press, 2nd edition, Oct. 1999.
- [130] S. Quinn. Examining the state of preparedness of Information Technology management in New Zealand for events that may require forensic analysis. *Digital Investigation*, 2:pages 276–280, 2005.
- [131] E. T. Ray. *Learning XML*. O'Reilly and Associates, 1st edition, Sept. 2003.
- [132] J. Reason. Human error: Models and Management. *British Medical Journal*, 320:pages 768–770, Mar. 2000.
- [133] M. Reith, C. Carr, and G. Gunsch. An Examination of Digital Forensic Models. *International Journal of Digital Evidence*, 1(3), 2002.
- [134] A. Riahi-Belkaoui and R. D. Picur. Understanding Fraud in the Accounting Environment. *Managerial Finance*, 26(11):pages 33–41, 2000.
- [135] G. Robb. *White-Collar Crime in Modern England: Financial Fraud and Business Morality, 1845-1929*. Cambridge University Press, 1st edition, July 2002.
- [136] R. Rowlingson. A Ten Step Process for Forensic Readiness. *International Journal of Digital Evidence*, 2(3), Winter 2004.
- [137] J. Roy and A. Ramanujan. XML: Data's Universal Language. *IT Pro*, pages 32–36, May 2000.
- [138] V. O. Safonov. *Trustworthy Compilers*. Wiley and Sons, 1st edition, Mar. 2010.
- [139] M. J. Saks. Merlin and Solomon: Lessons from the law's formative encounters with forensic identification science. *Hastings Law Journal*, 49, 1998.
- [140] M. J. Saks and D. L. Faigman. Failed Forensics: How Forensic Science Lost its way and how it Might yet Find it. *Review of Law and Social Science*, 4:pages 149–171, 2008.
- [141] S. Sanderson. *Pro ASP.NET MVC Framework*. Apress, 2nd edition, Apr. 2009.
- [142] B. Schneier. *Secrets and Lies: Digital Security in a Networked World*. Wiley and Sons, 2000.
- [143] B. Schneier. A New Secure Hash Standard. Online, Feb. 2007. Available at [http://www.schneier.com/blog/archives/2007/02/a\\_new\\_secure\\_ha.html](http://www.schneier.com/blog/archives/2007/02/a_new_secure_ha.html).
- [144] H. Schuster and T. Frieden. Lawyer wrongly arrested in bombings: 'We lived in 1984'. Online, Nov. 2006. <http://edition.cnn.com/2006/LAW/11/29/mayfield.suit/index.html>.
- [145] S. Sippu and E. Soisalon-Soininen. *Parsing Theory: Volume II LR(k) and LL(k) Parsing*. Springer, 1st edition, Nov. 1990.
- [146] M. Solon and P. Harper. Preparing evidence for court. *Digital Investigation*, 1:pages 279–283, 2004.
- [147] W. Stallings. *Operating Systems: Internals and Design Principles*. Prentice Hall, 6th edition, Apr. 2008.
- [148] W. Stallings. *Computer Organization and Architecture: Designing for Performance*. Prentice Hall, 8th edition, Apr. 2009.
- [149] R. Standefer. *Enterprise XML Clearly Explained*. Morgan Kaufmann, 1st edition, Jan. 2001.
- [150] P. Stephenson. A Comprehensive Approach to Digital Forensic Investigation. *Information Security Technical Report*, 8(2):pages 42–547, 2003.
- [151] P. Stephenson. EnCase Software. SC Magazine, Online, Nov. 2006. Available at <http://www.scmagazineus.com/encase-forensic/review/878/>.
- [152] D. W. Straub and R. J. Welke. Coping with Systems Risk: Security planning models for management decision making. *MIS Quarterly*, 22(4):pages 441–469, 1998.

- [153] Y. Su and S. Y. Yan. *Principles of Compilers: A New Approach to Compilers Including the Algebraic Method*. Springer, 1st edition, Nov. 2011.
- [154] J. Tan. Forensic Readiness. Technical report, AtStake, July 2001. [http://www.arcert.gov.ar/webs/textos/forensic\\_readiness.pdf](http://www.arcert.gov.ar/webs/textos/forensic_readiness.pdf).
- [155] C. Taylor, B. Endicott-Popovsky, and D. A. Frincke. Specifying digital forensics: A forensics policy approach. *Digital Investigation*, 4S:pages 101–104, 2007.
- [156] TechMediaNetwork.com. Top 10 reviews: BitDefender Total Security 2012. Online, 2011. Available at <http://internet-security-suite-review.toptenreviews.com/premium-security-suites/bitdefender-total-security-review.html>.
- [157] T. A. Tizon, S. Rotella, and R. B. Schmitt. Critics Galvanized by Oregon Lawyer’s Case. Online, May 2004. <http://articles.latimes.com/2004/may/22/nation/na-mayfield22>.
- [158] U.S. Supreme Court. *Daubert v. Merrell Dow Pharmaceuticals, Inc*, volume 509 U.S. 579 of *Certiorari to the United States Court of Appeals for the Ninth Circuit*. Government of the United States of America, 1993.
- [159] J. R. Vacca. *Identity Theft*. Prentice Hall, 1st edition, Sept. 2002.
- [160] C. G. van der Merwe and J. E. du Plessis. *Introduction To the Law of South Africa (Introduction to the Laws of Series)*. Kluwer Law International, 1st edition, June 2004.
- [161] S. Vanstone, P. van Oorschot, and A. Menezes. *Handbook of Applied Cryptography*. CRC Press, 1st edition, 1997.
- [162] S. J. Vaughan-Nicols. XML Raises Concerns as It Gains Prominence. *Computer*, pages 14–16, May 2003.
- [163] W3C. Homepage of the W3C Organisation. Online, 2012. Available at [www.w3.org](http://www.w3.org).
- [164] C. Walter. Kryder’s Law. *Scientific American*, Aug. 2005.
- [165] C. S. Warren, J. M. Reeve, and J. Duchac. *Financial and Managerial Accounting*. Cengage Learning, 10th edition, 2009.
- [166] F. Webster. *Theories of the Information Society*. Routledge, 1st edition, Nov. 1995.
- [167] E. Wilde and R. J. Glushko. XML Fever. *ACM Queue*, pages 48–53, Oct. 2008.
- [168] S. Willassen. Hypothesis-based Investigation of Digital Timestamps. *Advances in Digital Forensics*, 4:pages 75–86, 2008.
- [169] S. Y. Willassen. *Methods for Enhancement of Timestamp Evidence in Digital Investigations*. PhD thesis, Norwegian University of Science and Technology Faculty of Information Technology, Mathematics and Electrical Engineering Department of Telematics, Jan. 2008. <http://ntu.diva-portal.org/smash/get/diva2:124235/FULLTEXT01>.
- [170] R. Williams. *Real-Time Systems Development*. Butterworth-Heinemann, 1st edition, Dec. 2005.
- [171] H. B. Wolfe. Computer Forensics. *Computers and Security*, 22(1):pages 26–28, Feb. 2003.
- [172] L. Writer. *The Australian Book of Disasters*. Murdoch Books, 1st edition, dec 2011.
- [173] L. Yona. *Financial accounting for executive MBA’s*. AuthorHouse, 1st edition, Jan. 2013.