

Marcus Rogers
Kathryn C. Seigfried-Spellar (Eds.)



114

Digital Forensics and Cyber Crime

4th International Conference, ICDF2C 2012
Lafayette, IN, USA, October 2012
Revised Selected Papers



 Springer

Lecture Notes of the Institute
for Computer Sciences, Social Informatics
and Telecommunications Engineering

114

Editorial Board

Ozgur Akan

Middle East Technical University, Ankara, Turkey

Paolo Bellavista

University of Bologna, Italy

Jiannong Cao

Hong Kong Polytechnic University, Hong Kong

Falko Dressler

University of Erlangen, Germany

Domenico Ferrari

Università Cattolica Piacenza, Italy

Mario Gerla

UCLA, USA

Hisashi Kobayashi

Princeton University, USA

Sergio Palazzo

University of Catania, Italy

Sartaj Sahni

University of Florida, USA

Xuemin (Sherman) Shen

University of Waterloo, Canada

Mircea Stan

University of Virginia, USA

Jia Xiaohua

City University of Hong Kong, Hong Kong

Albert Zomaya

University of Sydney, Australia

Geoffrey Coulson

Lancaster University, UK

Marcus Rogers
Kathryn C. Seigfried-Spellar (Eds.)

Digital Forensics and Cyber Crime

4th International Conference, ICDF2C 2012
Lafayette, IN, USA, October 25-26, 2012
Revised Selected Papers

 Springer

Volume Editors

Marcus Rogers
Purdue University, Center for Education
and Research in Information Assurance and Security
Lafayette, IN, 47906, USA
E-mail: rogersmk@purdue.edu

Kathryn C. Seigfried-Spellark
The University of Alabama
Tuscaloosa, AL, 35487, USA
E-mail: spellark@seattleu.edu

ISSN 1867-8211

e-ISSN 1867-822X

ISBN 978-3-642-39890-2

e-ISBN 978-3-642-39891-9

DOI 10.1007/978-3-642-39891-9

Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013944669

CR Subject Classification (1998): K.4.1, K.4.4, K.6.5, K.5, C.5.3, C.3, J.1

© ICST Institute for Computer Science, Social Informatics and Telecommunications Engineering 2013

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

As we continue our journey into the twenty-first century, our reliance on technology and cyber infrastructure has increased at an incredible pace. This fact is not lost on those who wish to use technology for criminal activities. If we believe the current statistics that are available, the frequency and prevalence of cybercrime have reached epidemic proportions. The estimated loss as a result of cyber criminal activity is hundreds of millions of dollars annually. Cybercrime is undoubtedly the new transnational criminal activity with no boundaries or borders. As such, it is important that the international community (industry, government, law enforcement, and academia) come together in order to share ideas and possible solutions to help us deal with this global, increasing risk.

This volume contains papers presented at the 4th Annual International ICST Conference on Digital Forensics and Cybercrime (ICDF2C 2012), held October 25–26 2012, in Lafayette, Indiana, USA. ICDF2C is unique in that it brings together the various constituents in the digital forensic field (e.g., academia, government, law enforcement, business). The focus of the conference is on the applied nature of digital forensics and it provides an opportunity for researchers and practitioners to interact and discuss the various challenges and potential solutions.

The 20 papers presented in this volume represent an excellent cross section of the current work in the field of digital forensics. The topics covered range from behavior and law, to mobile devices, malware and new developments. The papers went through a double-blind peer review process. Three reviewers selected from the Technical Program Committee reviewed each paper.

We would like to thank those individuals who volunteered their time to be on the Technical Program Committee, as well as all of the people who volunteered to be Chairs of the various sections. The success of the conference was in large part due to the efforts of the staff at EAI, with a special thank you for the tireless efforts of Erica Polini, who kept us all on track.

Marcus Rogers
Kathryn C. Seigfried-Spellar

Legal Chair

Mike Losavio University of Louisville, USA

Behavioral Sciences Chair

Kate Seigfried-Spellar Seattle University, USA

Steering Committee

Pavel Gladyshev University College, Dublin
Marcus Rogers Purdue University, USA
Ibrahim Baggili Zayed University, United Arab Emirates
Sanjay Goel University at Albany, State University of
New York, USA

Technical Program Committee

Ibrahim Baggili Zayed University, United Arab Emirates
Felix Balado University College Dublin, Ireland
Florian Buchholz James Madison University, USA
Glenn Dardick Longwood University, USA
Katrin Franke Gjøvik University College, Norway
Felix Freiling University of Mannheim, Germany
Sandra Frings Fraunhofer IAO, Germany
Zeno Geradts Netherlands Forensic Institute,
The Netherlands
Pavel Gladyshev University College Dublin, Ireland
Sanjay Goel University at Albany, State University of
New York, USA
Andrew Harbison Grant Thornton, Ireland
Michael Harris Ernst & Young, Ireland
Andrew Jones Khalifa University, United Arab Emirates
Anthony Keane Blanchardstown Institute of Technology,
Ireland
Tahar Kechadi University College Dublin, Ireland
Chang-Tsun Li University of Warwick, UK
Juha Lampinen National Bureau of Investigation, Finland
Vivienne Mee Rits Computer Forensics, Ireland
Nasir Memon Polytechnic Institute of New York University,
USA
Stig Mjølhus Norwegian University of Science and
Technology, Norway
George Mohay Queensland University of Technology, Australia
Bruce Nikkel UBS, Switzerland
Slim Rekhis Institute of Technology in Communications,
Tunisia

Marcus Rogers	Purdue University, USA
John Sheppard	Waterford Institute of Technology, Ireland
Eugene Spafford	CERIAS - Purdue University, USA
Bhadran V.K.	Resource Centre for Cyber Forensics, India
Vinod Bhattathiripad	Farouq Institute of Management Studies, India
Oscar Vermaas	Netherlands National Police Force, The Netherlands
Svein Willassen	Norwegian University of Science and Technology, Norway
Gregg Gunsch	Defiance College, USA
Tim Wedge	Defiance College, USA
James Lyle	NIST, USA
Nicole Beebe	University of Texas San Antonio, USA

Table of Contents

Cloud Investigations

Cloud Computing Reference Architecture and Its Forensic Implications: A Preliminary Analysis	1
<i>Keyun Ruan and Joe Carthy</i>	
Cloud Forensic Maturity Model	22
<i>Keyun Ruan and Joe Carthy</i>	
Identifying Remnants of Evidence in the Cloud	42
<i>Jeremy Koppen, Gerald Gent, Kevin Bryan, Lisa DiPippo, Jillian Kramer, Marquita Moreland, and Victor Fay-Wolfe</i>	

Malware

On Improving Authorship Attribution of Source Code	58
<i>Matthew F. Tennyson</i>	

Behavioral

Towards Automated Malware Behavioral Analysis and Profiling for Digital Forensic Investigation Purposes	66
<i>Ahmed F. Shosha, Joshua I. James, Alan Hannaway, Chen-Ching Liu, and Pavel Gladyshev</i>	
Measuring the Preference of Image Content for Self-reported Consumers of Child Pornography	81
<i>Kathryn C. Seigfried-Spellar</i>	
Cybercrime, Censorship, Perception and Bypassing Controls: An Exploratory Study	91
<i>Ibrahim Baggili, Moza Al Shamlan, Bedoor Al Jabri, and Ayesha Al Zaabi</i>	

Law

When Should Virtual Cybercrime Be Brought under the Scope of the Criminal Law?	109
<i>Litska Strikwerda</i>	

New Developments in Digital Forensics

Research Trends in Digital Forensic Science: An Empirical Analysis of Published Research	144
<i>Ibrahim Baggili, Afrah BaAbdallah, Deena Al-Safi, and Andrew Marrington</i>	
Face Recognition Based on Wavelet Transform and Adaptive Local Binary Pattern	158
<i>Abdallah Mohamed and Roman V. Yampolskiy</i>	
Similarity Preserving Hashing: Eligible Properties and a New Algorithm MRSH-v2	167
<i>Frank Breitinger and Harald Baier</i>	
Investigating File Encrypted Material Using NTFS \$logfile	183
<i>Niall McGrath and Pavel Gladyshev</i>	
Finding Data in DNA: Computer Forensic Investigations of Living Organisms	204
<i>Marc B. Beck, Eric C. Rouchka, and Roman V. Yampolskiy</i>	
On the Completeness of Reconstructed Data for Database Forensics	220
<i>Oluwasola Mary Adedayo and Martin S. Olivier</i>	

Mobile Device Forensics

BlackBerry PlayBook Backup Forensic Analysis	239
<i>Mohamed Al Marzougy, Ibrahim Baggili, and Andrew Marrington</i>	
ANTS ROAD: A New Tool for SQLite Data Recovery on Android Devices	253
<i>Lamine M. Aouad, Tahar M. Kechadi, and Roberto Di Russo</i>	
Evaluating and Comparing Tools for Mobile Device Forensics Using Quantitative Analysis	264
<i>Shahzad Saleem, Oliver Popov, and Oheneba Kwame Appiah-Kubi</i>	
Detection of Masqueraded Wireless Access Using 802.11 MAC Layer Fingerprints	283
<i>Christer Idland, Thomas Jelle, and Stig F. Mjølunes</i>	

Cybercrime Investigations

BREDOLAB: Shopping in the Cybercrime Underworld	302
<i>Daan de Graaf, Ahmed F. Shosha, and Pavel Gladyshev</i>	
A Review and Comparative Study of Digital Forensic Investigation Models	314
<i>Kwaku Kyei, Pavol Zavarisky, Dale Lindskog, and Ron Ruhl</i>	
Author Index	329

Cloud Computing Reference Architecture and Its Forensic Implications: A Preliminary Analysis

Keyun Ruan and Joe Carthy

Center for Cybersecurity and Cybercrime Investigation
University College Dublin
{keyun.ruan, joe.carthy}@ucd.ie

Abstract. In this paper, researchers provide a preliminary analysis on the forensic implications of cloud computing reference architecture, on the segregation of duties of cloud actors in cloud investigations, forensic artifacts on all layers of cloud system stack, cloud actors interaction scenarios in cloud investigations, and forensic implications of all cloud deployment models. The analysis serves as feedback and input for integrating forensic considerations into cloud standardization processes from early stage, and specifies requirements and directions for further standardization efforts.

Keywords: Cloud Forensics, NIST, Cloud Computing, Standardization, Digital Investigation, Digital forensics.

1 Introduction

In late 2011, NIST released its final definition of cloud computing after 15 versions of working definitions, and it is defined as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model offers three types of service models, i.e., Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS), and four types of deployment models, i.e., private cloud, community cloud, public cloud and hybrid cloud (Mell and Grance 2011).

As an extension to the NIST cloud computing definition, a NIST Cloud Computing Reference Architecture (Liu et al. 2011) has been released as a generic high-level conceptual model for discussing the requirements that are the basis for discussing the characteristics, uses, and standards for cloud computing (Hogan et al 2011). The NIST Cloud Computing Standards Roadmap (Hogan et al 2011) has also been released after surveying the existing standards landscape for security, portability, and interoperability standards/models/studies/use cases, etc. relevant to cloud computing in order to identify standards gaps and standardization priorities. However, little has been mentioned on the forensic implications and standardization gap in these documents.

Several researchers have identified various challenges posed by cloud adoption to digital investigation (Spyridopoulos and Katos 2011, Birk and Wegener 2011, Biggs and Vidalis 2009, Ruan et al. 2011A). In 2011, hackers rented Amazon servers and launched the second-largest online data breach in U.S. history (Galante et al. 2011). The need for digital investigation in cloud environments is only going to rise as cloud adoption emerges. According to survey results based on 156 forensic experts and practitioners worldwide (Ruan et al. 2011B), more than half of the respondents agree that “establishment of a foundation of standards and policies for forensics that will evolve together with the technology” is an opportunity for cloud forensics, 88.89% of the respondents agree or strongly agree that “designing forensic architecture for the Cloud” is a valuable research direction for cloud forensics.

Digital forensics has historically been an “after-after-thought” whereas security has been an “after-thought” whenever new technologies emerge. This could be one of the reasons why today cybercrime causes an annual loss of 750 billion Euros in Europe alone, according to new statistics released by Interpol (Cheslow 2012). On the other hand, the field of digital forensics lacks consensus in fundamental aspects of its activities in terms of methodology and procedures (Cohen 2011). There is no single framework that can be used as a general guideline for investigating all incidents cases (Selamat et al 2008), and a comprehensive model of cybercrime investigation is important for standardizing terminology, defining requirements, and supporting the development of new techniques and tools for investigations (Ciardhuáin 2004).

Cloud computing is expected to reach maturity in another decade (CSA 2012, Thomason 2010), as one of most significant paradigm shifts in computing history, it is an unique timing for digital forensics to be pro-actively integrated in cloud architectural design and standard acceleration. As a first step, in this paper researchers analyze the forensic implications based on the high-level conceptual cloud computing reference architecture and specify requirements for the future standardization efforts. The analysis is independent from any specific jurisdiction or specific service offering.

2 Cloud Actors and Segregation of Duties

As shown in Fig 1, Liu et al. (2011) defines five major cloud actors: cloud consumer, cloud provider, cloud carrier, cloud auditor and cloud broker. Each actor is an entity (a person or an organization) that participates in a transaction or process and/or performs tasks in cloud computing. In this paper researchers discuss two types of digital investigation, i.e. internal investigation happens within the cloud environment among cloud actors for security and incident response purposes, and external investigations initiated by external parties such as law enforcement for civil or criminal investigation. In this section, segregation of duties of each cloud actor regarding digital investigation is analyzed.

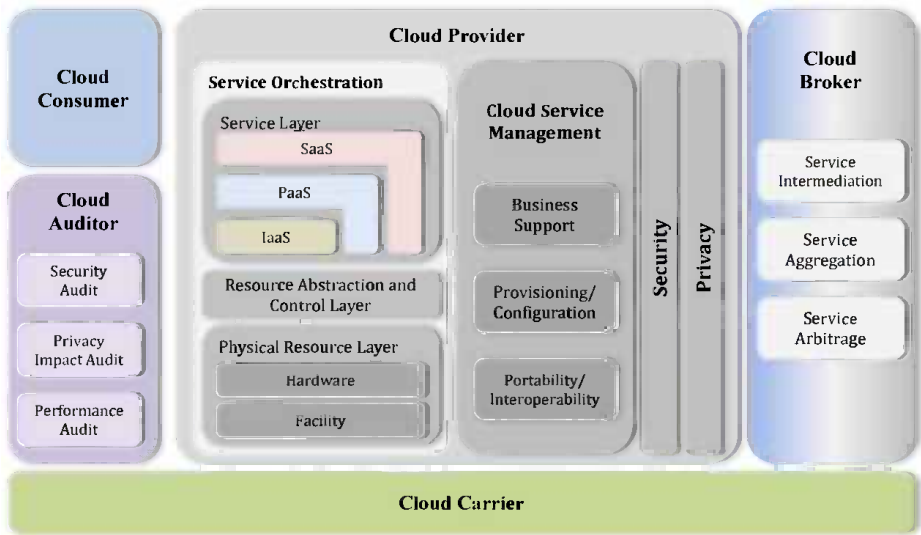


Fig. 1. NIST Cloud Conceptual Reference Model (Liu et al. 2011)

2.1 Cloud Provider and Cloud Consumer

According to NIST definition in Liu et al. 2011, the cloud provider is a person, an organization; it is the entity responsible for making a service available to interested parties through different cloud offerings. A cloud provider acquires and manages the computing infrastructure required for providing the service, runs the cloud software that provides the service, and makes arrangement to deliver the cloud services to the cloud consumers through network access. A cloud provider's activities can be described in five major areas, i.e., service deployment, service orchestration, cloud service management, security, and privacy.

As data is being migrated to cloud providers, so is evidence. Service provider will inevitably be expected to become evidence provider. Increasingly, both consumer and law enforcement will acquire access to evidence and demand forensic support from cloud providers. However, interfaces for such access and requirement of such support are still largely undefined.

As defined in Liu et al. 2011, the cloud consumer represents a person or organization that maintains a business relationship with, and uses the service from a cloud provider. A cloud consumer browses the service catalog from a cloud provider, requests the appropriate service, sets up service contracts with the cloud provider, and uses the service. The cloud consumer is the principle stakeholder for the cloud computing service.

As the principle stakeholder for cloud computing service, the consumer is responsible to demand visibility and control, be aware of its own risks from cloud migration, and make sure that appropriate security controls are implemented. However, guidelines on assessing forensic risks and concerns are still largely missing for consumers.

The segregation of duties between cloud provider and cloud consumers regarding forensic investigations is very complex and needs to be further clarified. With the absence of such clarification, in this section researchers utilize the Cloud Security Alliance (CSA) Cloud Control Matrix (CCM) v1.2 which rests on other industry-accepted security standards, regulations and controls frameworks such as HITRUST CSF, ISO 27001/27002, ISACA COBIT, PCI, HIPPA, NIST and SAS 70 as a starting point, identify controls that are closely related to forensic process, then group them into a) sole provider responsibility and b) provider and consumer shared responsibility.

The current forensic-related responsibilities expected solely from the provider are as follows:

1) Data ownership and stewardship (related control DG-01): all data should be designated with stewardship with assigned responsibilities defined, documented and communicated by the cloud provider. Such designation and documentation can be used for identification of evidence ownership and chain of custody in a forensic investigation.

2) Data retention and disposal (related control DG-04 DG-05): cloud provider must ensure backup and redundancy mechanisms are in place for data retention and storage; testing recovery of backups must be implemented at planned intervals by the cloud provider; cloud provider must secure disposal and complete removal of data from all storage media, ensuring data is not recoverable by any computer forensic means. Redundant storage is a source for forensic investigation. Event reconstruction and evidence recovery can be made possible through restoring back-ups. However, evidence will not be recoverable if the provider has physically destroyed all storage where evidence might reside.

3) Facility Security (related control FS-03 FS-04 FS-05 FS-06): authorization and access control to physical facility security should be ensured and reinforced by the cloud provider. As a result, cloud provider is responsible of providing access logs to physical storage.

4) Clock Synchronization (related control SA-12): an external accurate, externally agreed upon, time source should be used by the cloud provider to synchronize the system clocks of all relevant information processing systems within the organization or explicitly defined security domain to facilitate tracing and reconstitution of activity timelines. Clock synchronization is critical for analysis of event sequence and event reconstruction in a forensic investigation.

5) Audit log and intrusion detection (related control SA-14): audit logs recording privileged user access activities, authorized and unauthorized access attempts, system exceptions, and information security events shall be retained, complying with applicable policies and regulations. Audit logs shall be reviewed at least daily and file integrity (host) and network intrusion detection (IDS) tools implemented to help facilitate timely detection, investigation by root cause analysis and response to incidents. Physical and logical user access to audit logs shall be restricted to authorized personnel. In cases of an investigation, provider should be responsible of providing such audit logs.

The current forensic-related responsibilities expected to be shared between provider and consumer are as follows:

1) Audit (related control CO-01 CO-02 CO-03): audit planning, independent audit, third-party audits should be carried about by both provider and consumer on data duplication, access and data boundary limitations. Forensic related terms need to be defined and included in audit planning, independent audit, and third-party audits from both provider and consumer side.

2) Regulatory mapping (related control CO-05): information system elements (data, objects, applications, infrastructure and hardware) may be assigned a legislative domain and jurisdiction to facilitate statutory, regulatory and contractual requirements for compliance mapping. This mapping is of significant value in determining the legislative domain of digital evidence.

3) Data classification/labeling/handling (related control DG-02 DG-03): data and objects containing data shall be assigned a classification based on data type, jurisdiction of origin, jurisdiction domiciled, legal constrains, contractual constrains, and sensitivity that can be useful in a forensic investigation.

4) Asset management (related control FS-08): a complete inventory of critical assets shall be maintained with ownership defined and documented. In a forensic investigation, such documentation can be of great value and needs to be provided by both provider and consumer.

5) Authentication and Authorization (related control IS-07, IS-08, SA-02, SA-07): granting and revoking normal and privileged access to applications, databases, systems, databases, server, network, and sensitive data should be restricted and approved. Multi-factor authentication is required for all remote user access. In a forensic investigation, authentication and authorization logs and records to critical assets under investigation need to be provided by both provider and consumer.

6) Incidence management (related control IS-22 IS-25): policies and procedures should be established to triage security related events and ensure timely and thorough incident management. Mechanism shall be put in place to monitor and quantify the type, volumes and costs of information security incidents. Mechanisms to trigger post-incident investigations need to be included in the incidence management procedures.

7) Legal preparation (related control IS-24): in the event a follow-up action concerning a person or organization after an information security incident requires legal action proper forensic procedures including chain of custody shall be required for collection, retention, and presentation of evidence to support potential legal action subject to the relevant jurisdiction.

8) Data integrity and segmentation (related control SA-03 SA-05 SA-09): system interfaces, jurisdictions, or with a third party shared service provider to prevent improper disclose, alteration or destruction. The preservation of evidence integrity and segmentation is also a shared responsibility between provider and consumer.

Despite forensic-related controls are specified in CCM here and there, a separate set of controls explicitly covers the whole forensic process specifying a list of forensic capabilities is needed to further clarify, analyze and enforce the segregation of duties regarding forensic investigations among all cloud actors, especially between

provider and consumer at current stage. To develop such a set of controls, researchers suggest the following steps:

- 1) Identification of a list of forensic capabilities include
 - a) Investigative capabilities: a mapping of various existing forensic process models to cloud environment to cover core forensic phases.
 - b) Pre-investigative capabilities: defining a set of capabilities that are needed for pro-active forensic readiness in cloud environment, such as identity management, encryption management, interoperability management capabilities.
 - c) Supportive capabilities: certain capabilities are needed throughout the whole forensic process but are not core forensic phases, such as evidence management, case management, multi-jurisdiction, multi-tenancy capabilities.
 - d) Interfacing capabilities: the split of control in cloud environment implies the need for access and exchange forensic data, thus interfacing capabilities need to be defined between cloud actors, especially cloud provider and consumer when it comes to internal investigation. Interfacing capabilities also need to be defined for external investigation when law enforcement is involved.
- 2) An in-depth analysis of segregation of duties between provider and consumer against the list of forensic capabilities, in which requirement of forensic support from provider side can be better understood and demanded from the consumer through contractual negotiation.
- 3) Identification of a list of forensic capabilities that can be integrated and provided as a service through standard interfaces, so that providers can start integrating these services at early stage while cloud technology matures.

2.2 Cloud Broker

A cloud broker is an entity that manages the use, performance and delivery of cloud services and negotiates relationships between cloud providers and cloud consumers. As cloud computing evolves, the integration of cloud services can be too complex for cloud consumers to manage. As shown in Fig 2 below, a cloud consumer may request cloud services from a cloud broker, instead of contacting a cloud provider directly, and in this case the actual cloud providers are invisible to the cloud consumer and the cloud consumer interacts directly with the cloud broker. The cloud broker may create a new service by combining multiple services or by enhancing an existing service. In general, a cloud broker can provide services in service intermediation, service aggregation and service arbitrage (Liu et al. 2011)



Fig. 2. NIST Usage Scenario for Cloud Broker (Liu et al. 2011)

According to Gartner (Cearley and Smith 2012), cloud brokerage is expected to accelerate over the next three years and will facilitate cloud consumption. Cloud broker can play the following role in a forensic investigation:

- 1) Aggregate forensic capabilities of multiple providers and deliver to consumer while actual providers are hidden from the consumer
- 2) Facilitate investigation by adding an extra layer of forensic support, for example in areas of evidence segregation and interfacing law enforcement.

2.3 Cloud Carrier

A cloud carrier acts as an intermediary that provides connectivity and transport of cloud services between cloud consumers and cloud providers. Cloud carriers provide access to consumers through network, telecommunication and other access devices. As shown in Fig 3 below, the cloud provider arranges for two unique Service Level Agreements (SLAs), one with a cloud carrier (e.g., SLA2) and one with a cloud consumer (e.g., SLA1). A cloud provider may request dedicated and encrypted connections to ensure the cloud services are consumed at a consistent level according to the contractual obligations with the cloud consumers. In this case, the provider may specify its requirements on capability, flexibility and functionality in SLA2 in order to provide essential requirements in SLA1. (Liu et al. 2011)

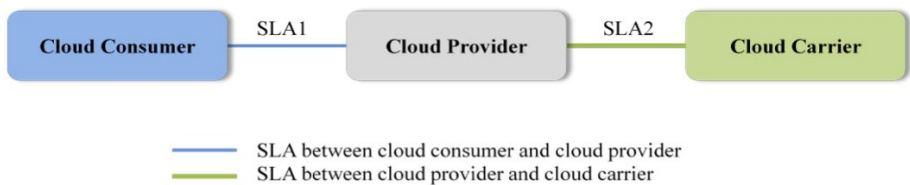


Fig. 3. NIST Usage Scenario for Cloud Carrier (Liu et al. 2011)

Carriers are not likely to be directly involved in a forensic investigation, however they can still play a critical role in providing pre-investigative and supportive capabilities, such as evidence transport, chain of custody, and inter-cloud forensic capabilities.

2.4 Cloud Auditor

A cloud auditor is a party that can perform an independent examination of cloud service controls with the intent to express an opinion thereon. Audits are performed to verify conformance to standards through review of objective evidence. A cloud auditor can evaluate the services provided by a cloud provider in terms of security controls, privacy impact, performance, etc. The audit may involve interactions with both cloud consumer and cloud provider, as shown in Fig 4 below (Liu et al. 2011)

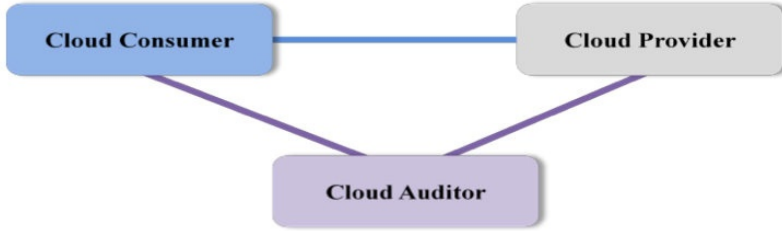


Fig. 4. NIST Usage Scenario for Cloud Auditor (Liu et al. 2011)

Forensic capabilities and segregation of duties among cloud actors in delivering these capabilities to facilitate both internal and external cloud investigations need to be reflected into auditable regulatory or contractual language. Currently these terms are missing. A set of key terms for the Service Level Agreement (SLA) between the cloud provider and cloud consumer are identified and recommended by Ruan et al. (2012).

3 Forensic Artifacts in Cloud Environment

A generic stack diagram is defined in NIST Reference Architecture (Liu et al. 2011) to represent the grouping of three types of system components for delivering cloud services, i.e., Physical Resource Layer, Resource Abstraction Layer, and Service Layer, as shown in Fig 5. Similar to traditional computer system stack, a list of forensic artifacts and its order of volatility need to be identified and specified for the cloud system stack.

3.1 Physical Layer

The Physical Resource Layer includes hardware computing resources such as computers (CPU and memory), networks (routers, firewalls, switches, network links and interfaces) and storage components (hard disks) and other physical computing infrastructure elements, as well as facility resources such as heating, ventilation, and air conditioning (HVAC), power, communications, and other aspects of the physical plant (Liu et al. 2011).

This layer consists of physical storage and is under control of the cloud provider. It is often geographically distant from the consumer and the law enforcement. Forensic artifacts for the hardware layer include hard disks, network logs, router logs, etc. This layer also includes data center artifacts such as access records, facility logs, activity logs, interior and exterior camera footage, biometrics records, visitor records, organization chart and contact information, etc. Gaining access to actual physical data center and carry out on-site investigation can be too costly or even impossible in most cases. Forensic artifacts on this layer often have to be acquired through remote forensics, or provided by provider.

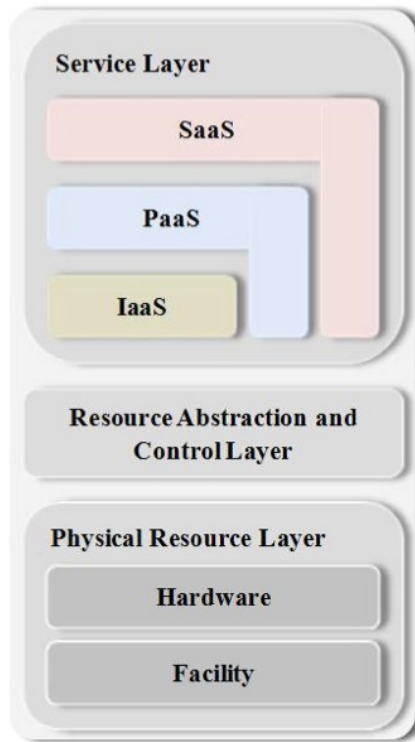


Fig. 5. Cloud System Environment (Liu et al. 2011)

3.2 Abstraction Layer

The Resource Abstraction and Control Layer contains the system components that Cloud Providers use to provide and manage access to the physical computing resources through software abstraction. Resource abstraction components typically include software elements such as hypervisors, virtual machines, virtual data storage, and other computing resource abstractions (Liu et al. 2011).

This layer is under control of the cloud provider and hidden from the consumer. However this layer is extremely critical for addressing multi-tenant issues around evidence segregation, and locating the actual physical computing resources (hard disk storage, etc.) from virtual resources in the Service Layer. Forensic artifacts on this layer include hypervisor event logs, virtual images, etc. Barrett (2012) provides a comprehensive overview of virtual forensics in cloud environments.

3.3 Service Layer

The Service Layer is where Cloud Providers define *interfaces* for Cloud Consumers to access the computing services. Access interface of each of the three service models

are provided in this layer. It is possible, though not necessary, that SaaS applications can be built on top of PaaS components and PaaS components can be built on top of IaaS components. (Liu et al. 2011)

The Service Layer is where the segregation of duties between the provider and the consumer comes in, and the segregation is where the interface is. Forensic artifacts reside from the service interface above can and need to be collected by the consumer. Forensic artifacts reside from the service interface below (including Resource Abstraction and Control Layer and Physical Resource Layer) can and need to be collected by the provider. As discussed earlier, a set of standardized forensic interfaces need to be defined and integrated into different service layer corresponding to forensic capabilities required from both provider and consumer side.

3.3.1 OS Layer (IaaS)

The IaaS interface layer can also be called OS (Operating System) Layer, as this layer of interface provides interfaces to access operating system and drivers, and is hidden from SaaS consumers and PaaS consumers. An IaaS cloud allows on or multiple guest OS's to run virtualized on a single physical host. Generally, consumers have broad freedom to choose which OS to be hosted among all the OS's that could be supported by the Cloud Provider. The IaaS consumers should assume full responsibility for the guest OS's, while the IaaS provider controls the host OS (Liu et al. 2011).

Forensic artifacts on this layer are similar to forensic artifacts in virtual OSs, which include virtual operating system event logs, configuration logs, audit logs, registry, anti-virus/anti-spyware application logs, intrusion detection system logs, virtual network logs, etc.

3.3.2 Middleware Layer (PaaS)

The PaaS interface layer can also be called Middleware Layer, as this layer of interface provides software building blocks (e.g., libraries, database, and Java virtual machine) for developing application software in the cloud. The middleware is used by PaaS consumers, installed/managed/maintained by IaaS consumers or PaaS providers, and hidden from SaaS consumers (Liu et al. 2011).

Forensic artifacts on this layer are similar to forensic artifacts in traditional (integrated) development environment, which include source code, performance logs, debugging logs, access logs, account information, etc.

3.3.3 Application Layer (SaaS)

The SaaS interface layer can also be called Application Layer, as this layer of interface includes software applications targeted at end users or programs. The applications are used by SaaS consumers, or installed/managed/maintained by PaaS consumers, IaaS consumers and SaaS providers (Liu et al. 2011).

Forensic artifacts on this layer are similar to forensic artifacts in traditional software applications, which include application logs, authentication and authorization logs,

account information, etc. The only difference is the software is hosted remotely from the consumer via the browser (or other thin-client or thick-client) thus thin-client/thick-client forensic data collection will play a major role in forensic data collection on this layer from the consumer side.

3.4 Forensic Acquisition in the Cloud

Based on the analysis above, researchers conclude that forensic acquisition in the cloud has to resort to a hybrid approach of remote, live, virtual, network, thin-client, thick-client, large-scale forensic acquisition due to the nature of forensic artifacts in cloud environments. A list of pro-active forensic artifacts needs to be identified across the cloud system stack to ensure forensic readiness. The identification of pro-active forensic artifacts must evolve closely with the developments of cloud SIEM solutions. A list of re-active forensic artifacts needs to be identified across cloud system stack with order of volatility for post-incident forensic evidence collection. Some of the e-discovery methodologies can be borrowed in identifying and collecting re-active forensic artifacts, such as creating a “data map” for these artifacts (Gonsowski 2012).

4 Cloud Actors Interactions

There are various ways for cloud actors to interact in cloud investigations. In this section, researchers introduce three main organizational interaction scenarios for cloud investigations based on the analysis of the forensic implications of the three main usage scenarios described in Liu et al. (2011). These interaction scenarios are detailed views of the organizational dimension described in Ruan et al. (2011) and are analyzed under the aspects of 1) Service level agreements 2) Internal and external investigation and 3) Forensic artifacts.

4.1 Scenario 1

Fig 6 depicts the simplest scenario for cloud actors' interaction. In a service offering, there is a single relation between the cloud consumer and the cloud provider, the cloud provider may or may not provide services through a cloud carrier.

The consumer signs a SLA (SLA1) with the provider. The provider signs a separate SLA (SLA2) with the carrier when the relation between provider and carrier exist. A cloud auditor may be involved to audit SLA(s). Forensic segregation of duties, requirements and implementations need to be defined and audited through the SLA(s).

An internal investigation happens between the provider and consumer shared systems. An external investigation is initiated by law enforcement towards the consumer, provider or system shared by provider and consumer. Provider, or consumer, or may resort to external assistance in enhancing forensic capabilities in facing internal or external investigations.

Forensic artifacts are scattered between provider and consumer systems.

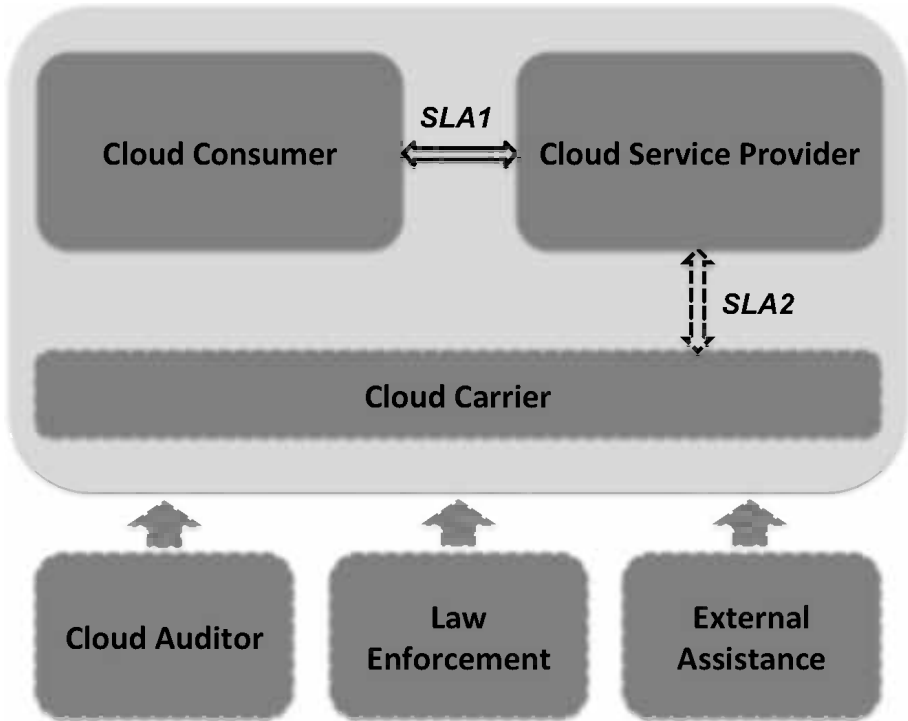


Fig. 6. Cloud Actors Interaction Scenario 1

4.2 Scenario 2

In the scenario shown in Fig 7, the cloud broker is acting as a cloud provider to the cloud consumer. The actual provider(s) are invisible to cloud consumer.

The consumer signs SLA A with broker. The broker signs a range of SLAs (SLA B1, SLA B2, SLA B3, ...) with multiple providers (Cloud Service Provider 1, Cloud Service Provider 2, Cloud Service Provider 3) respectively, and may sign a separate SLA C with a cloud carrier when services are delivered through a carrier. A cloud auditor may be involved to audit SLA(s). Forensic segregation of duties, requirements and implementations need to be defined and audited through the SLA(s).

An internal investigation happens within the shared cloud environment among cloud consumer, broker and provider(s). An external investigation is initiated by law enforcement towards cloud consumer, one or multiple providers, or broker, or cloud resources shared by consumer, broker, and provider(s).

Forensic artifacts are scattered across consumer, provider(s) and broker systems.

Computing resources of one or more of these actors in the shared cloud system and might, and very likely will involve all of the cloud actors in the investigative process as forensic artifacts are scattered across the shared system.

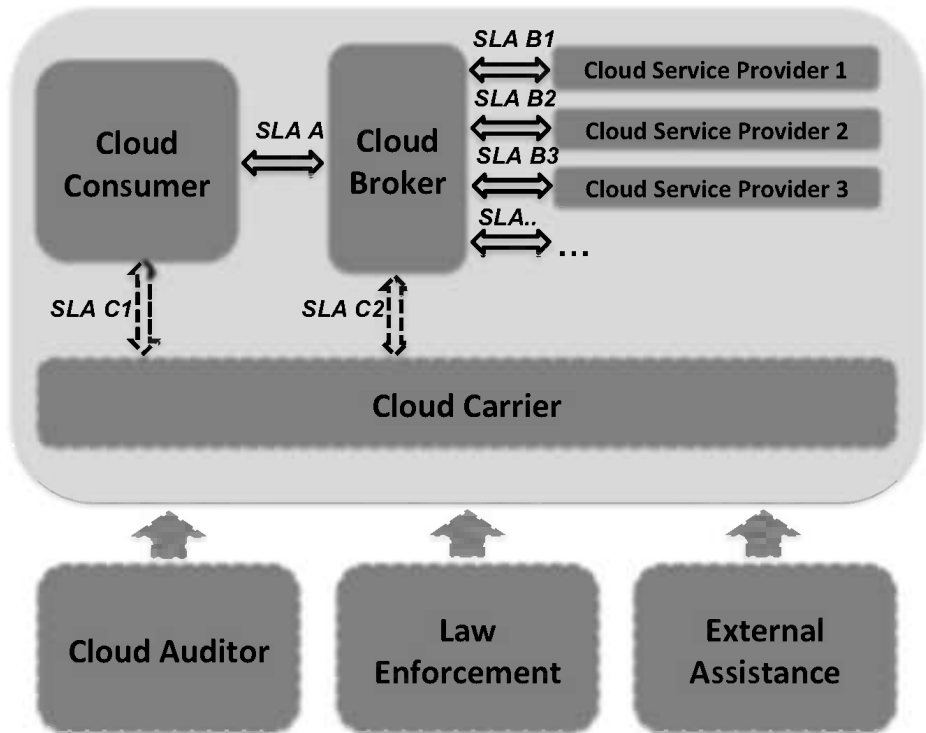


Fig. 7. Cloud Actors Interaction Scenario 2

4.3 Scenario 3

In the third scenario demonstrated in Fig 8, there is a liner chain of dependencies between cloud entities. One cloud consumer uses service(s) from a cloud provider, which uses service(s) from another cloud provider. It is a repetition of scenario 1.

Each pair of service relation between two cloud entities is defined via a SLA (e.g., SLA A1, SLA A2, ..). In cases when services are delivered through a cloud carrier, separate SLAs (e.g. SLA B1, SLA B2, SLA B3) are specified between the cloud entity and the cloud carrier. A cloud auditor might be involved to audit the SLAs among cloud entities, in which case forensic requirements and performances should be audited and evaluated.

An internal investigation happens within the cloud systems shared among the chain of cloud entities. An external investigation happens when law enforcement initiate an investigation to one or more or all entities in the chain of cloud entities which might anyways affect the whole chain of cloud entities later on in the investigative process. Any pair of cloud entities on two sides of a SLA might resort to external assistance in enhancing forensic capabilities in both internal and external investigations, which should be specified in the SLA.

Forensic artifacts are scattered throughout the chain of cloud entities in shared environment. Segregation of duties between each pair of the entities (one acts as provider, another acts as consumer) is similar to scenario 1 described earlier.

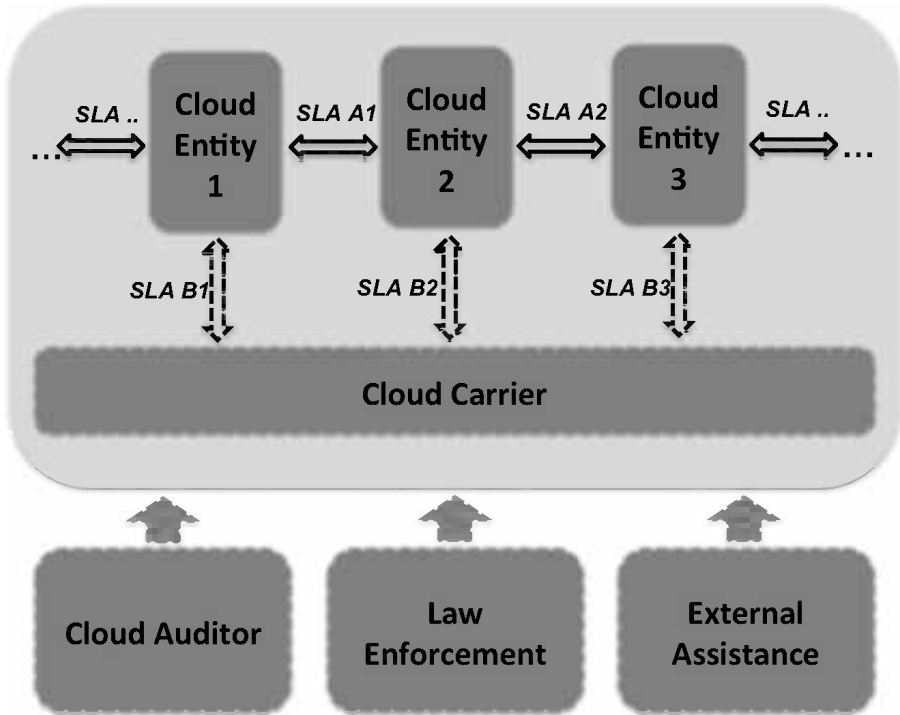


Fig. 8. Cloud Actors Interaction Scenario 3

5 Cloud Deployment Models and Forensic Implications

There are four types of cloud deployment models according to Liu et al. 2011. In this section, forensic implications in technical, organizational and legal dimensions of these four deployment models are analyzed based on the three-dimensional model proposed in Ruan et al. 2011.

5.1 Public Cloud

A public cloud is one in which the cloud infrastructure and computing resources are made available to the general public over a public network, as shown in Fig 9. A public cloud is owned by an organization selling cloud services, and serves a diverse pool of clients (Liu et al. 2011)

Salesforce Chatter, Gmail, Dropbox are popular public SaaS offerings. Force.com and Google App Engine are leading public PaaS offering providers. Amazon Web Service (AWS) and Windows Azure are leading public IaaS offering providers.

5.1.1 Cloud Consumers Accessing the Cloud over a Network

In this case, cloud consumers are often small-scale enterprises or personal users who have minimum or none forensic capabilities of their own, or large enterprise or

government agencies seeking cheap deployment or storage for non-mission critical services.

Technically, this deployment model often allows easy registration and anonymous usage that could be exploited by malicious users. Personal users need to pay attention to how Personal Identifiable Information (PII) information are used, stored and transferred in the cloud system. Providers need to deliver strong capabilities in evidence segregation in elastic multi-tenant environment and evidence acquisition with the proliferation of client endpoints.

Organizationally, policies and procedures on forensic capabilities and implementations mostly rely on the provider side.

Legally, multiple jurisdictions are a default scenario and there is often standard SLA between provider and consumer with little room for customization and negotiation.

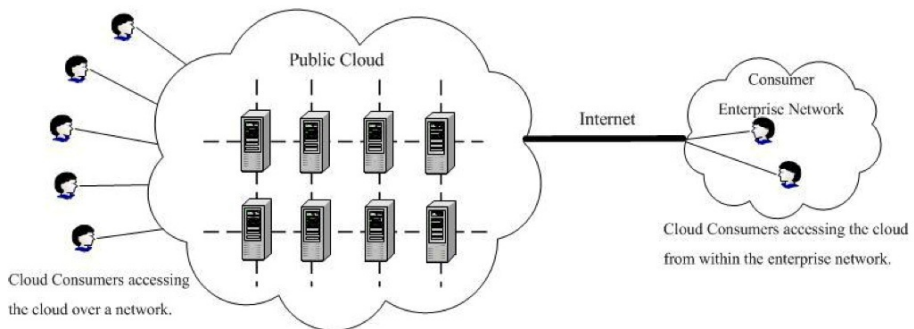


Fig. 9. Public Cloud (Liu et al. 2011)

5.1.2 Cloud Consumers Accessing the Cloud from within the Enterprise Network

In this case, cloud consumers are often enterprises (or government agencies) that deploy non-mission critical services in the public cloud. These consumers typically have certain level of internal security/forensic implementations before migrating to the cloud.

Technically, the default level of security/forensic implementations of the provider can sometimes be higher than consumer's legacy implementations, thus migrating to the cloud can result in an "upgrade" in security/forensic implementations from the consumer side. An extra layer of authorization/authentication and access control can be added through the enterprise network.

Organizationally, consumer may share some of the responsibilities on policy and procedures on forensic implementations.

Legally, consumer can specify the jurisdiction where its data resides via SLA.

5.2 Private Cloud

As shown in Fig 10, a private cloud gives a single cloud consumer's organization the exclusive access to and usage of infrastructure and computational resources. It may be managed either by the cloud consumer organization or a third party, and may be hosted on the organization's premises (i.e. on-site private clouds) or outsourced to a hosting company (i.e. outsourced private clouds) (Liu et al. 2011)

Oracle Grid, IBM Cloudburst are leading private IaaS offerings. Oracle Fusion, IBM Dynamic Infrastructure are leading private PaaS offerings. Sun Comms Suite, IBM LotusLive iNotes, IBM Smart Analytics Cloud, e.g., are private SaaS solutions.

5.2.1 On-Site Private Cloud

This deployment model is similar to traditional internal enterprise IT infrastructure. In this case, the cloud consumers are often medium-large enterprise or government agencies that deploy mission critical services in the private cloud. These consumers typically have a high level of internal security/forensic implementation before migrating to the cloud.

Technically, when the level of the consumer's legacy security/forensic implementations is higher than the provider's default offering, cloud migration can result in a "downgrade" on security/forensic implementations on the consumer side in exchange for a reduced cost of IT infrastructure and such risk of "downgrade" needs to be thoroughly assessed before migration.

Organizationally, collaborative efforts need to be made by forensic teams from both provider and consumer side to deliver strong forensic capabilities.

Legally, data resides on-premise thus evidence will be in the same jurisdiction(s) as consumer.

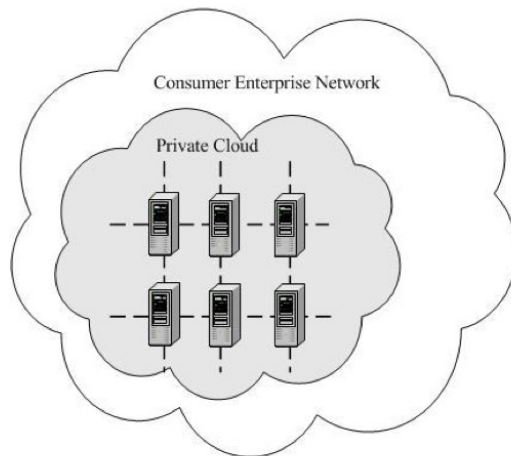


Fig. 10. On-site private cloud (Liu et al. 2011)

5.2.2 Out-Sourced Private Cloud

Out-sourced private cloud, as shown in Fig 11, is cheaper compare to on-site private cloud deployment model because maintenance and infrastructure of the private cloud is off-premise. All implications are similar to previous case except that legally, the private cloud can be in different jurisdiction(s) than the consumer.

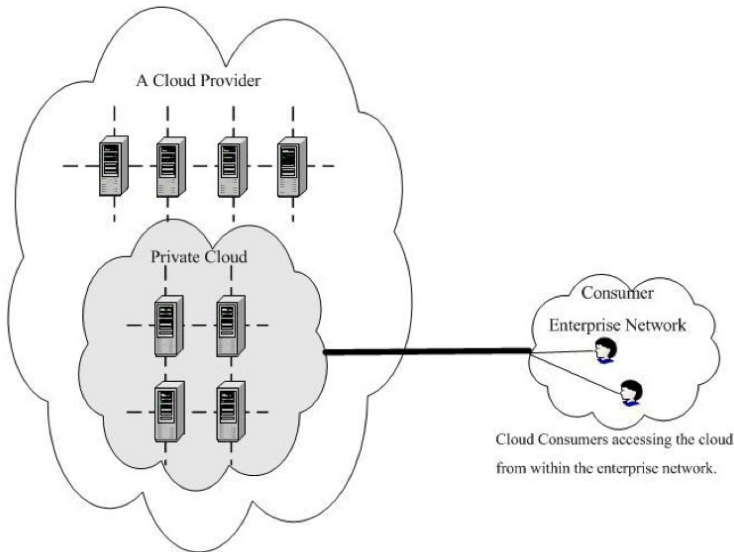


Fig. 11. Out-sourced Private Cloud (Liu et al. 2011)

5.3 Community Cloud

As shown in Fig 12, a community cloud serves a group of cloud consumers who have shared concerns such as mission objectives, security, privacy and compliance policy, rather than serving a single organization as a private cloud does. Similar to private clouds, a community cloud may be managed by the organizations or by a third-party, and may be implemented on customer premise (i.e. on-site community cloud) or outsourced to a hosting company (i.e. outsourced community cloud) (Liu et al. 2011)

IBM's Federal Community Cloud (FCC), for example, is a community cloud solution for federal organizations. NYSE Technologies supports a community cloud called Capital Markets Community Cloud.

5.3.1 On-Site Community Cloud

In this case, cloud resources are hosted by one or multiple organizations in the same community that provide and consumer these cloud resources, and these cloud resources can be accessed remotely from other organizations in the same community.

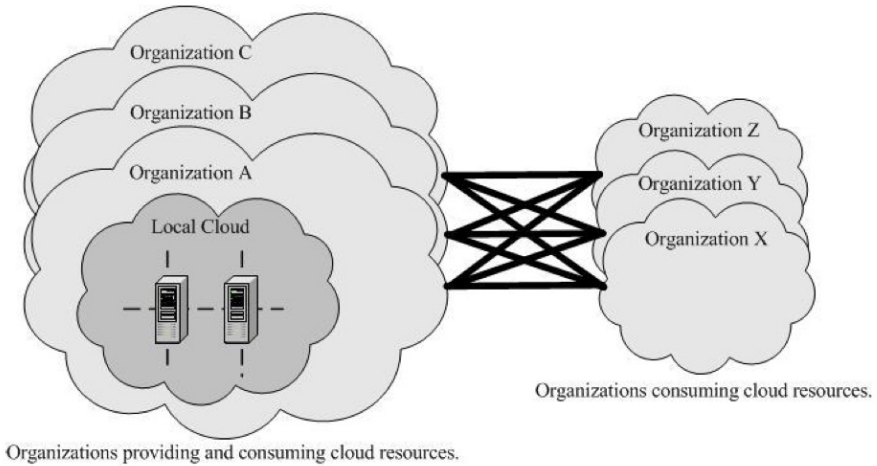


Fig. 12. On-site Community Cloud (Liu et al. 2011)

Technically, forensic capabilities are delivered by multiple hosting organizations with a joint effort. Evidence segregation is needed among multiple tenant organization(s) consuming the community cloud.

Organizationally, policies and procedures on forensic implications are shared among hosting organizations and tenant organizations.

Legally, evidence can reside in different jurisdiction(s) when hosting organization(s) and tenant organization(s) are geographically remote. Multi-tenant issues exist among tenant organizations within the community.

5.3.2 Outsourced Community Cloud

In the case of outsourced community cloud as shown in Fig 13, multiple organizations in the same community share a private cloud hosted by a cloud provider and access cloud resources remotely. Outsourced community cloud is cheaper than on-premise community cloud because maintenance and infrastructure of the community cloud is off-premise.

Technically, forensic capabilities are provided by the cloud provider and the tenant organizations in the community. Evidence segregation is needed among multiple consumer organizations consuming the community cloud.

Organizationally, policies and procedures on forensic implications are shared among provider and consumer organizations.

Legally, evidence can reside in different jurisdiction(s) when provider and consumer organizations are geographically remote. Multi-tenant issues exist among consumer organizations within the community.

5.4 Hybrid Cloud

As shown in Fig 14, a hybrid cloud is a composition of two or more clouds (on-site private, on-site community, off-site private, off-site community or public) that remain as distinct entities but are bound together by standardized or proprietary technology that enables data and application portability (Liu et al. 2011)

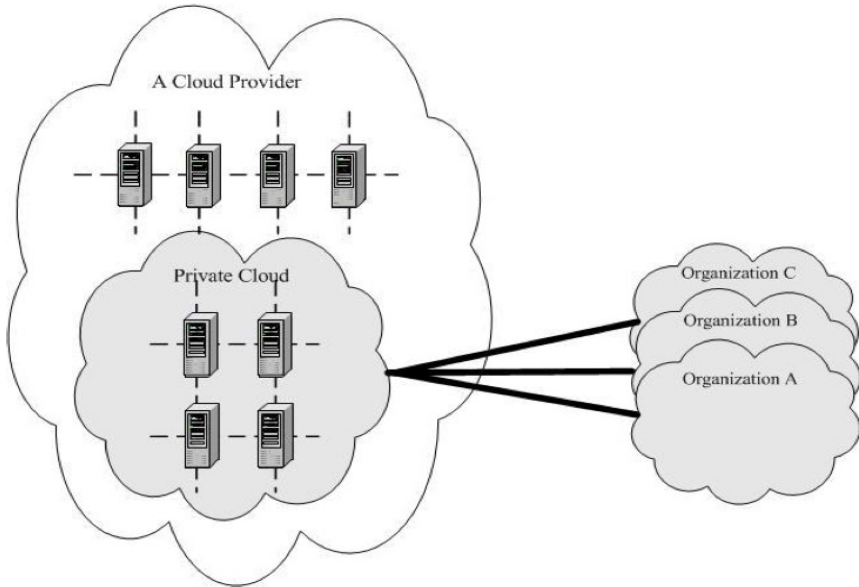


Fig. 13. Outsourced Community Cloud (Liu et al. 2011)

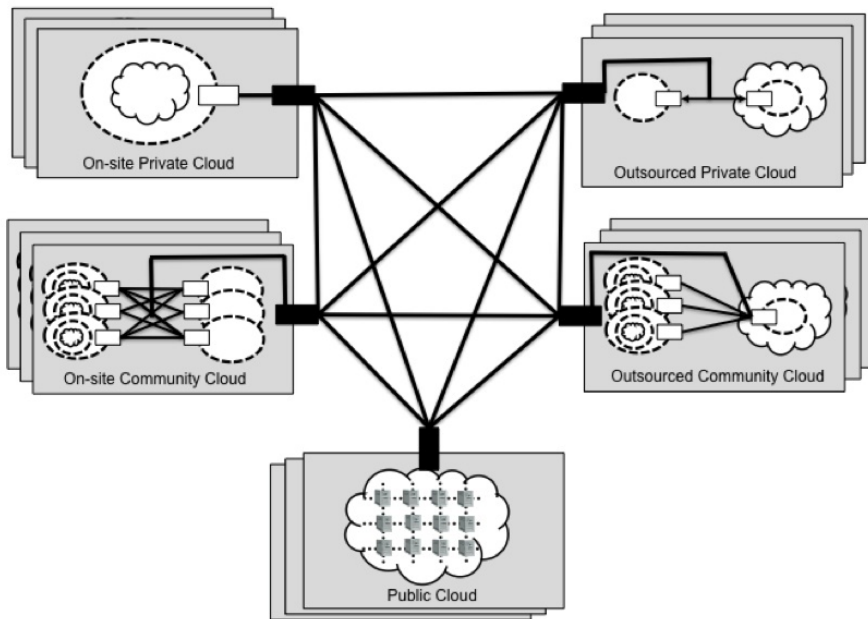


Fig. 14. Hybrid Cloud (Liu et al. 2011)

According to Garner, hybrid computing is among top 5 trends of cloud computing and could lead to a unified model over time in which there is a single “cloud” made up of multiple cloud platforms (internal or external) that can be used as needed based on changing business requirements (Cearley and Smith 2012). Both security and forensic implications are extremely complex and are out of scope at current stage.

6 Conclusions and Future Work

Based on the preliminary analysis of the cloud reference architecture researchers conclude the following directions are important for better integration the missing considerations of forensic capabilities in cloud standardization process.

A standardization gap analysis is needed for forensic capabilities based on a mapping of traditional forensic process models to cloud environments. A forensic reference architecture for the cloud needs to be developed to be used as a baseline for analyzing and discussing forensic issues in cloud environments. A forensic capability model needs to be developed for cloud environments specifying segregation of duties of all cloud actors and mechanisms to access and audit such capabilities. Pro-active and re-active forensic artifacts need to be identified across cloud system stack with order of volatility for collection. A set of forensic interfaces need to be defined and implemented between cloud actors, especially between provider and consumer on the service layer at current stage, in order to collect and aggregate forensic artifacts for both internal and external investigative purposes. Such interfaces can be integrated as a service from the provider. Forensic considerations need to be included in the cloud interoperability discussions as the emergence of cloud brokerage and hybrid cloud deployment model indicates that the complexity of cloud forensics will soon go beyond provider and consumer, becoming a challenge for the entire cloud ecosystem.

Researchers are actively working on some of directions above.

References

- Ciardhuain, S.O.: An Extended Model of Cybercrime Investigations. *International Journal of Digital Evidence* 3(1) (2004)
- Selamat, S.R., Yusof, R., Sahib, S.: Mapping Process of Digital Forensic Investigation Framework. *International Journal of Computer Science and Network Security* 8(10) (2008)
- Spyridopoulos, T., Katos, V.: Requirements for a Forensically Ready Cloud Storage Service. *International Journal of Digital Crime and Forensics* 3(3), 19–36 (2011)
- Birk, D., Wegener, C.: Technical issues of forensic investigations in cloud computing environments. In: 2011 IEEE Sixth International Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE), pp. 1–10 (2011), doi:10.1109/SADFE.2011.17
- Biggs, S., Vidalis, S.: Cloud computing: The impact on digital forensic investigations. In: *International Conference for Internet Technology and Secured Transactions, ICITST 2009*, pp. 1–6 (2009)
- Cohen, F.: Putting the Science in Digital Forensics. *Journal of Digital Forensics, Security and Law* 6(1), 7–14 (2011)

- CSA, Cloud Computing Market Maturity Study Results. Cloud Security Alliance, ISACA (September 2012)
- Galante, J., Kharif, O., Alpeyev, P.: Sony Network Breach Shows Amazon Cloud's Appeal for Hackers, Bloomberg (May 16, 2011), <http://www.bloomberg.com/news/2011-05-15/sony-attack-shows-amazon-s-cloud-service-lures-hackers-at-pennies-an-hour.html> (retrieved on June 22, 2012)
- Cearley, D.W., Smith, D.M.: Five Cloud Computing Trends That Will Affect Your Cloud Strategy Through 2015, Gartner (February 10, 2012)
- Thomason, I.: Cloud Services a Decade away from Maturity, V3.co.uk (August 19, 2010), <http://www.v3.co.uk/v3-uk/news/1999968/cloud-services-decade-away-maturity>
- Hogan, M., Liu, F., Sokol, A., Tong, J.: NIST Cloud Computing Standards Roadmap. National Institute of Standards and Technology. Special Publication 500-291 (2011)
- Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., Leaf, D.: 'NIST Cloud Computing Reference Architecture' National Institute of Standards and Technology. Special Publication 500-292 (2011)
- Mell, P., Grance, T.: 'The NIST Definition of Cloud Computing' National Institute of Standards and Technology, Special Publication 800-145 (2011)
- Ruan, K., Carthy, J., Kechadi, T., Crosbie, M.: Cloud Forensics. In: Peterson, G., Sheno, S. (eds.) Advances in Digital Forensics VII. IFIP AICT, vol. 361, pp. 35–46. Springer, Heidelberg (2011a)
- Ruan, K., James, J.I., Carthy, J., Kechadi, T.: Key Terms For Service Level Agreements To Support Cloud Forensics. In: Peterson, G., Sheno, S. (eds.) Advances in Digital Forensics VIII. IFIP AICT, vol. 363, pp. 201–212. Springer, Heidelberg (2012)
- Ruan, K., Carthy, J., Kechadi, T.: Survey on cloud forensics and critical criteria for cloud forensic capability: A preliminary analysis. In: 6th Annual Conference of the ADFSL Conference on Digital Forensics, Security and Law, Richmond, Virginia, USA (2011b)
- Cheslow, D.: Interpol to crack down on cyber crime, MSNBC.com (2012), http://www.msnbc.msn.com/id/47338831/ns/technology_and_science-tech_and_gadgets/t/interpol-crack-down-cyber-crime/ (retrieved June 22, 2012)
- Gonsowski, D.: Compliance in the Cloud & the Implication on Electronic Discovery. In: Ruan, K. (ed.) Cyber Crime and Cloud Forensics: Applications of Investigative Processes. IGI Global (December 2012)
- Barrett, D.: Security Architecture and Forensic Awareness: Forensics in Virtualized Environments. In: Ruan, K. (ed.) Cyber Crime and Cloud Forensics: Applications of Investigative Processes. IGI Global (December 2012)

Cloud Forensic Maturity Model

Keyun Ruan and Joe Carthy

Center for Cybersecurity and Cybercrime Investigation
University College Dublin
{keyun.ruan, joe.carthy}@ucd.ie

Abstract. In this paper we present a shortened version of the Cloud Forensic Maturity Model (CFMM). It composes of two inter-related parts, i.e., the Cloud Forensic Investigative Architecture (CFIA) and the Cloud Forensic Capability Matrix (CFCM). The CFMM is developed in order to create a reference model to evaluate and improve cloud forensic maturity. It is a part of an on-going project, and is evaluated by a panel of experts and practitioners as a first step for further cloud forensic standardization efforts.

Keywords: Cloud Forensics, Cloud Computing, Digital Forensics, Cloud Forensic Maturity Model, Cloud Forensic Investigative Architecture, Cloud Forensic Capability Matrix, Cloud Forensic Standardization.

1 Introduction

As the cloud paradigm emerges, the need for carrying out digital investigation in cloud computing environments has become inevitable, no matter it is internal investigation initiated by one of the cloud actors to investigate security incidents and policy violations, or external investigation initiated by law enforcement to investigate criminal or civil cases. Cloud forensics is at its infancy. It is faced with challenges in technical, organizational, and legal dimensions, as well as promising opportunities as listed in Ruan et al. (2011A). The cloud paradigm shift has initiated a major standardization wave. It is an unique timing to analyze and integrate missing forensic considerations and capabilities into the standardization and maturing process of cloud computing.

Based on the survey “Cloud Forensics and Critical Criteria for Cloud Forensic Capability” carried out for the purpose of this research (Ruan et al. 2011B), we propose the Cloud Forensic Maturity Model (CFMM), a reference model for evaluating, developing and improving cloud forensic maturity. CFMM composes of two inter-related parts, i.e., the Cloud Forensic Investigative Architecture (CFIA), and the Cloud Forensic Capability Matrix (CFCM). CFIA is a conceptual reference architecture for digital investigations in cloud computing environments. CFCM is a matrix to evaluate and improve capabilities that correspond to components in CFIA.

In this paper we introduce a shortened version of CFMM due to the page limit. We discuss the initial validation and feedback for CFMM carried out by a panel of digital forensic experts and practitioners. We then provide three brief use cases of CFMM. Firstly we use it to discuss investigative scenarios and generate process models. Secondly we use it to compare current cloud capabilities of several leading cloud offerings. Lastly we use it to analyze cloud forensic standardization gaps.

This research is still on-going with various in-depth analyses and mappings, however, we believe it is useful to share with the research community our current thinking and progress on developing such a model in order to lay a foundation and provide a structure for various discussions and efforts at the early days of cloud forensic research and development.

2 Cloud Forensic Investigative Architecture (CFIA)

The Cloud Forensic Investigative Architecture (CFIA) is developed to include key components for enabling digital investigations in cloud computing environment. A high level component based representation of the Cloud Forensic Investigative Architecture is shown in Figure 1.

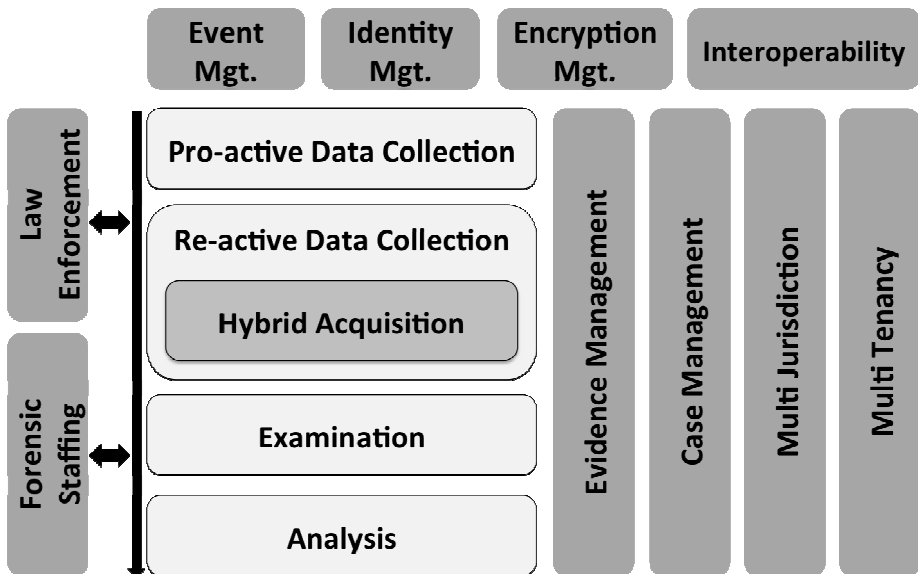


Fig. 1. Cloud Forensic Investigative Architecture

The Cloud Forensic Investigative Architecture is composed of four main sections as follows

- Pre-investigative Readiness
- Core-forensic Process
- Supportive Processes
- Investigative Interfaces

As shown in Fig 1, the black line represents the concept of interface. Everything on the right hand side of the black line is the cloud environment being investigated. The cloud environment consists of its technical infrastructure that is often a shared environment, the organizational interaction among all cloud actors (cloud consumer,

cloud provider, cloud broker, etc.) and its legal complications such as multi-jurisdiction and multi-tenancy.

On the left hand side of the black line are the “investigators”, either internal forensic team or external law enforcement, who carry out the investigation by utilizing and managing the forensic capabilities within the cloud environment adding their own forensic capabilities.

On the top of the architecture are the pre-investigative readiness components. Pre-investigative readiness components include event management, identity management, encryption management, and interoperability. These components are essential to ensure investigative preparedness and enable investigations.

In the centre of the architecture in the vertical layout are the core forensic process components. Core forensic process components include pro-active data collection, re-active data collection, hybrid acquisition, examination and analysis. Hybrid acquisition is a part of re-active data collection, however, as it includes a wide range of forensic acquisition techniques which will be discussed later in the paper, it is prudent to consider it as a separate core forensic phase.

On the right of the architecture in the horizontal layout are the supportive processes components. Supportive processes components include evidence management, case management, multiple jurisdiction and multi-tenancy. They are needed throughout the timeline of an investigation.

3 Cloud Forensic Capability Matrix (CFCM)

Borrowing core concepts of the Capability Maturity Model (CMM) for Software developed by Paulk (1993), the Cloud forensic Capability Matrix is a capability maturity model for assessing and improving cloud forensic capability maturity for any given cloud actor (i.e. cloud consumer, cloud provider, cloud broker, cloud carrier, cloud auditor) or law enforcement.

The Cloud Forensic Capability Matrix composes of six maturity levels from low to high as follows:

- Level 1 Minimum
- Level 2 Basic
- Level 3 Ad-hoc
- Level 4 Well-formalized
- Level 5 Mature
- Level 6 Advanced

Cloud forensic capabilities are the basis for the Cloud Forensic Capability Matrix, and are divided into four main categories corresponding to the cloud forensic architecture:

- Pre-investigative capabilities: capabilities in preparation for both internal and external investigations
- Investigative capabilities: capabilities required in the core investigative process
- Supportive capabilities: capabilities required to support and complete the investigation case

- Interfacing capabilities: capabilities dealing with the internal and external interface between the cloud system environment and investigative parties involved in cloud investigations.

Each capability is composed of key capability, sub capability and a set of key criteria in organizational, legal and technical dimensions. Figure 2 shows all key capabilities with sub capabilities as a detailed view of the CFIA.

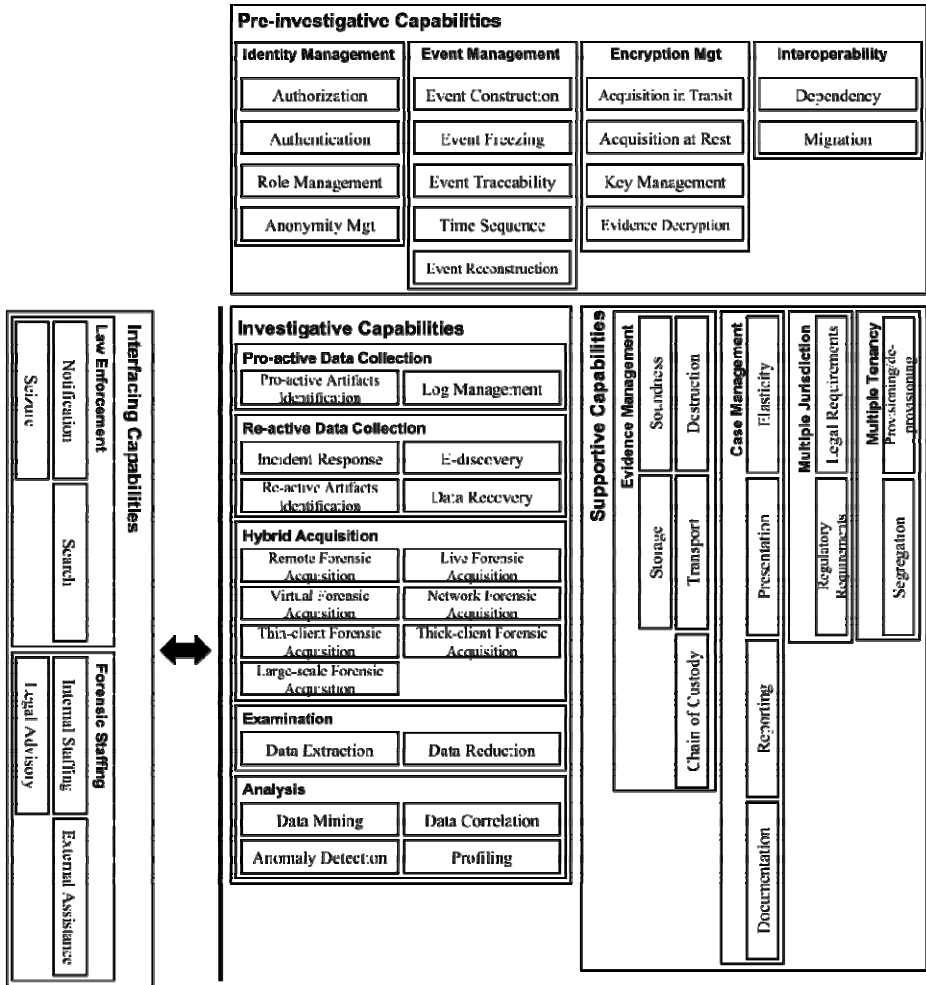


Fig. 2. Cloud Forensic Capabilities

Due to page limit, we provide a section of the CFCM, a top-level pre-investigative capability matrix for cloud consumer, cloud provider, cloud auditor and law enforcement as shown in Table 1 below.

Detailed descriptions and criteria requirements for each actor on each level for each capability are being developed and refined. Use cases are being collected for validation.

Table 1. Pre-investigative Capability Matrix for Cloud Consumer, Cloud provider, Cloud Auditor and Law Enforcement

		Consumer						Provider						Auditor						Law Enforcement							
		1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6		
Pre-investigative capabilities																											
Identity management																											
Authorization		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X								
Authentication		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X								
Role management		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X								
Anonymity management					X	X	X			X	X	X	X			X	X	X	X						X	X	
Event management																											
Event construction					X	X				X	X	X				X	X	X									
Event freezing					X	X				X	X					X	X							X	X		
Event traceability					X	X				X	X	X				X	X	X						X	X	X	
Time sequence					X	X	X			X	X	X	X			X	X	X	X								
Event reconstruction					X	X	X			X	X	X	X			X	X	X	X					X	X	X	X
Encryption management																											
Acquisition in transit					X	X				X	X	X				X	X	X							X	X	
Acquisition at rest					X	X	X			X	X	X	X			X	X	X	X					X	X	X	

- Event construction capability: the ability of a cloud entity to properly define what is considered to be an “event” in a cloud system, including a set of information needed to describe the “who”, “what”, “when”, “where” and “how” of the event.
- Event freezing capability: the ability of a cloud entity to “freeze” the event at the immediate state in case of criminal offense, intrusion, or investigation.
- Event traceability capability: the ability of a cloud entity to trace the state(s) of an event in the cloud system, or back to its original state.
- Time sequence capability: the ability of a cloud entity to maintain a definite and synchronized time sequence in the (shared) cloud system including maintaining time synchronization across the cloud environment.
- Event reconstruction capability: the ability of a cloud entity to reconstruct the past state of an event with a level of accuracy that the reconstructed information can be admitted as digital evidence.

4.1.3 Encryption Management

Encryption management capability is the ability of a cloud entity to search, acquire and access encrypted forensic data in shared cloud environment without breaching privacy or data protection regulation under jurisdiction(s) of concern. It includes four sub capabilities:

- Acquisition in transit capability: the ability of a cloud entity to search and acquire potential evidence from encrypted data in transit in live cloud transactions on the service layer of the cloud system.
- Acquisition at rest capability: the ability of a cloud entity to search and acquire potential evidence from encrypted data at rest in physical or virtual cloud storage.
- Key management capability: the ability of a cloud entity to ensure encryption keys are accessible to authorized internal investigators (human) or investigative agents (machine) to decrypt information that might be relevant to the investigation.
- Evidence decryption: the ability of a cloud entity to ensure potential digital evidence in the cloud environment can be appropriately decrypted for the purpose of lawful investigation without breaking laws or regulations under the jurisdiction(s) where the services operate.

4.1.4 Interoperability

Interoperability capability is the ability of a cloud entity to ensure forensic readiness in inter-cloud environments. It includes two sub capabilities:

- Dependency capability: the ability of a cloud entity to ensure forensic readiness when there is a chain of dependency of multiple service providers.
- Migration capability: the ability of a cloud entity to ensure forensic readiness when forensic data or digital evidence is migrated from one cloud to another.

4.2 Investigative Capabilities

4.2.1 Pro-active Data Collection

Pro-active data collection capability is the ability of a cloud entity to maximize its potential to use digital evidence while minimizing the cost of an investigation, i.e., the

preparedness and readiness of a cloud entity before an investigation. Pro-active data collection includes two sub capabilities:

- **Pro-active artifacts identification:** the ability of a cloud entity to identify, document and collect a list of digital artifacts that are essential for a digital investigation or can facilitate a digital investigation that need to be managed pro-actively before an investigation ensuring forensic soundness. These artifacts can be scattered all over the system and the organization, and might include non-digital information that needs to be digitized for future use. These artifacts vary greatly among different cloud actors and in different cloud offerings and are mostly static input for cloud forensic examination and analysis.
- **Log management capability:** the ability of a cloud entity in dealing with, often large volumes of, log messages generated from a cloud system while ensuring forensic soundness. Log messages generated from log management are the main input of static forensic data in forensic collection, and can also be useful for the purpose of regulatory compliance.

4.2.2 Re-active Data Collection

Re-active data collection capability is the ability of a cloud entity to trigger forensic data collection after an incident, either immediately (e.g. an intrusion alert) or after a period of time until the incident is discovered internally in the cloud system or externally notified by the law enforcement. Re-active data collection capabilities include four sub capabilities:

- **Incidence response capability:** the ability of a cloud entity to receive, review and respond to a (security) incident, from intrusion to criminal act. Analysis can be applied on synthesizing data from various sources to determine trends and patterns in incident activity. This information can be used to help predict future activity or to provide early warning when the activity matches a set of previously determined characteristics. In case of cloud investigation, notice from law enforcement can also be considered as an incident that needs to be responded to.
- **Re-active artifacts capability:** the ability of a cloud entity to have a well-defined and documented list of forensic artifacts that are essential for a digital investigation or can facilitate a digital investigation that need to be identified, collected and managed re-actively after an investigation. These artifacts can be scattered all over the cloud environment, i.e., in the service layer, abstraction layer, physical layer of the cloud stack, and among all cloud actors. They are often a hybrid combination of static and volatile digital artifacts, and might also include non-digital information that needs to be digitized for forensic examination and analysis. These artifacts vary greatly among different cloud actors and in different cloud offerings. Re-active artifacts capability also includes the ability of a cloud entity to specify the order of volatility of the forensic artifacts in re-active data collection. Generally the order should follow a. Service layer artifacts b. Abstraction layer artifacts c. Physical layer artifacts.
- **E-discovery capability:** the ability of a cloud entity to search and locate electronically stored information (ESI) about specific topic in the cloud

environment and provide them in a sound fashion. In case of digital investigation, e-discovery is a part of re-active data collection.

- **Data recovery capability:** the ability of a cloud entity to salvage data from damaged, failed, corrupted, inaccessible, or compromised physical or virtual storage media in the cloud environment when it cannot be accessed normally. Recovery may be required due to physical damage to the storage device, logical damage to the file system that prevents it from being mounted by the host operating system, or intentional damage by the criminal to destroy the digital evidence.

4.2.3 Hybrid Acquisition

Hybrid acquisition capability is the ability of the cloud entity to search and acquire forensic data from different layers and different components in the cloud environment. Cloud computing is a hybrid collection of many existing network, mobile, virtual and grid computing technologies, thus a hybrid combination of forensic acquisition techniques need to be configured in different investigative scenarios. Hybrid forensic acquisition capabilities include seven sub capabilities:

- **Remote forensic acquisition capability:** the ability of a cloud entity to search and acquire forensic data from geographically remote physical infrastructure via an active network connection. Remote forensic acquisition often consists of installing forensic agent on the remote hardware infrastructure, and grant access to search content and acquire decrypted forensic data to an authorized investigation request.
- **Live forensic acquisition capability:** the ability of a cloud entity to search and acquire forensic data from a running/live/volatile/dynamic system. Live forensic acquisition is usually carried out as a part of incident response to capture volatile forensic data from a live system before switching off the power to preserve memory, process, and network information that would be lost with traditional forensic approach. Cloud system cannot be easily ‘switched off’, thus making live forensic acquisition capability an essential capability for a cloud investigation to capture volatile forensic data from a cloud system.
- **Virtual forensic acquisition capability:** the ability of a cloud entity to search and acquire forensic data from virtualized environment, i.e., virtual machines, virtual images, hypervisors, and cloud resource abstraction layer in general.
- **Network forensic acquisition capability:** the ability of a cloud entity to search and acquire forensic data from a dynamic network. Broad network access is one of the essential characteristics of cloud computing thus making network forensic acquisition and essential capability for cloud investigations.
- **Thin client forensic acquisition capability:** the ability of a cloud entity to search, recover and acquire forensic data from thin clients, such as web-browser, mobile devices, smart phones, iPads, or any digital device that has both internal memory and communication ability, that are connected to the cloud and heavily dependent on services from the cloud. The rise of cloud computing is enabling a proliferation of “thin” endpoints globally, making thin-client forensic acquisition essential to cloud investigations.

- Thick client forensic acquisition capability: the ability of a cloud entity to search, recover, and acquire forensic data from thick clients, such as workstations, that are connected to the cloud.
- Large-scale forensic acquisition capability: the ability of a cloud entity to search, recover and acquire forensic data from large-scale systems with large data volume. It consists of the techniques to locate and search in large-scale data sets, and to process and transfer large volume of data.

4.2.4 Examination

Examination capability is the ability of a cloud entity to examine forensic data collected from the collection phase to generate input for further forensic analysis. Examination capability includes two sub capabilities:

- Data extraction capability: the ability of a cloud entity to retrieve data out of, often unstructured or poorly structured, raw forensic data sets collected from various sources in a cloud system for further forensic examination and analysis.
- Data reduction capability: the ability of a cloud entity to minimize the amount data that needs to be examined and analyzed in a forensic investigation. It is an automatic or semi-automatic process that can dramatically eliminates redundant data and reduces cost of investigation. Typical techniques of data reduction include data compression, filtering, and data de-duplication.

4.2.5 Analysis

Analysis capability is the ability of a cloud entity to analyze forensic data and generate analysis result as digital evidence. Analysis capability includes four sub categories:

- Data mining capability: the ability of a cloud entity to extract knowledge from large volume data sets in a human-understandable structure automatically or semi-automatically. Data correlation capability is the ability of a cloud entity to analyze whether and how strongly pairs of variables are related using statistical techniques. It is an essential capability to analyze forensic datasets generated from diverse sources.
- Anomaly detection capability: the ability of a cloud entity to detect patterns in a given dataset that do not conform to an established normal behavior in the forensic analysis phase. The patterns detected are called anomalies and are often critical in further analysis of the digital evidence.
- Profiling capability: the ability of a cloud entity to analyze traces from large volume data set in order to draw a profile relevant to the supporting of a digital investigation. It is an analysis process to discover from the correlations between data in forensic datasets that can be used to identify and represent a human or nonhuman subject (individual or group), and/or the application of profiles (sets of correlated data) to individuate and represent a subject or to identify a subject as a member of a group or category.

4.3 Supportive Capabilities

4.3.1 Evidence Management

Evidence management capability is the ability of a cloud entity to make sure evidence is kept and handled in a fashion ensuring the integrity of evidence throughout the evidence timeline so that the evidence is admissible to court, i.e., from acquisition, examination, analysis, transport, storage, presentation, to disposal. Evidence management capability includes five sub capabilities:

- Evidence transport capability: the ability of a cloud entity to transport evidence in a forensically sound manner to preserve evidence in its original form without undetectable addition, modification, and deletion of bits.
- Evidence storage capability: the ability of a cloud entity to store digital evidence so that it is well preserved when stored physically or electronically, ensuring the soundness of the evidence and the chain of custody in an investigation.
- Evidence destruction capability: the ability of a cloud entity to destroy evidence and other information associated with a legal matter after its use in the matter ends, often under the order from the courts. In the cloud scenario, the complete destruction of data means the destruction of the actual physical storage (e.g. hard drive) in a way that it is impossible for the data to be recovered.
- Evidence soundness capability: the ability of a cloud entity in ensuring the digital evidence remains in its original form without undetectable addition, deletion or modification of evidence data, throughout the evidence timeline within the cloud entity.
- Chain of custody capability: the ability of a cloud entity to chronologically document the entire digital evidence timeline, showing the seizure, custody, control, transfer, analysis and disposition of the physical or electronic evidence.

4.3.2 Case Management

Case management capability is the ability of a cloud entity to manage the investigative case in an appropriate, sufficient, and well-archived fashion. Case management capability includes three sub capabilities:

- Documentation capability: the ability of a cloud entity to appropriately document the investigative process throughout the case timeline, aspects include investigative techniques applied, chain of custody of evidence, investigators involved in the case, etc.
- Presentation capability: the ability of a cloud entity to appropriately present evidence, analysis, and interpretations in the investigative process in the form of expert reports, depositions, and testimony, aspects ranging from the order of presentation of information to the use of graphics and demonstrations.
- Reporting capability: the ability of a cloud entity to appropriately report the result of the investigative process, whether or not there are enough evidence to validate the hypothesis, based on which the investigate is carried out.

- **Elasticity capability:** the ability of a cloud entity to be flexible with the scale of the case size. As elasticity is one of the essential characteristics of cloud computing, services are easily scaled up and down based on demand, forensic cases can also range from small scale to large scale in one cloud environment, thus making elasticity a necessary capability for case management.

4.3.3 Multi-jurisdiction

Multi-jurisdiction capability is the ability of a cloud entity to have a clear understanding of different legal, regulatory requirements and forensic process under multiple jurisdictions so that the investigation is carried out in an appropriate, sufficient and legitimate manner. Multi-jurisdiction capability includes three sub capabilities:

Legal requirements: the ability of a cloud entity to have a clear understanding of the legal process(s) required for a digital investigation under the jurisdiction(s) services operate, including the aspects of criminal/civil processes, warrant, notification, search, seizure, evidence admissibility, etc.

Regulatory requirements: the ability of a cloud entity to have a clear understanding of the regulatory requirements related to digital investigation under the jurisdiction(s) service operate, including the aspects of data retention, evidence decryption, etc.

4.3.4 Multi-tenancy

Multitenancy capability is the ability of a cloud entity (provider, or broker on behalf of providers) to provision and de-provision forensic implementations among multiple tenants sharing same computing resources, as well as the ability to segregate tenants' data throughout the investigation process. Multitenancy capability includes two sub capabilities:

- **Segregation capability:** the ability of a cloud entity to segregate forensic data among different tenants in a shared cloud environment. In the public and community cloud environment, computing resources are shared on the physical and abstraction control layer of the cloud system stack among multiple tenants, and in both internal and external investigation, there is a need to rapidly and clearly segregate forensic data among different tenants so that tenants who are not related to the investigative case can stay out of the forensic process.
- **Provisioning/de-provisioning capability:** the ability of a cloud entity to rapidly provision and de-provision computing resources along with the forensic implementations for those computing resources among different tenants when needed.

4.4 Interfacing Capabilities

4.4.1 Law Enforcement

Law enforcement interface capability is the ability of a cloud entity to appropriately interface law enforcement in cases of external investigations while minimizing internal loss due to search and seizure of computing resources in the cloud

environment by the law enforcement. Law enforcement capability includes three sub capabilities:

- Notification capability: the ability of a cloud entity to notify all other cloud actors involved in a specific cloud service under investigation of law enforcement in a timely and appropriate manner
- Search capability is the ability of a cloud entity to interface with the law enforcement when facing a search (with warrant).
- Seizure capability: the ability of a cloud entity to properly respond and react to the request from law enforcement to seize its computing resources, or suspend its services to maintain business continuity or minimize financial loss.

4.4.2 Forensic Staffing

Forensic staffing capability is the ability of a cloud entity to organize a functional staffing structure to facilitate both internal and external investigations. Forensic staffing capability includes four sub capabilities:

- Internal forensic team capability: the ability of a cloud entity to form an ad-hoc or well-formalized team of forensic specialists to be in charge of full range of internal forensic capabilities.
- External assistance capability: the ability of a cloud entity to hire external assistance to assist in forensic capabilities, e.g., hybrid forensic acquisition, when they cannot be met internally.
- Legal advisory capability: the ability of a cloud entity to consult both internal and external legal advisory to assist internal or external investigations

5 Initial Validation and Feedback

As part of the initial validation process, a panel of 8 forensic practitioners and experts from law enforcement and academia was invited to assess and evaluate the proposed model based on a shortened description of the Cloud Forensic Investigative Architecture and the Cloud Forensic Capability Matrix. The panel was asked the following 3 questions:

- (1) Do you think the investigative architecture can work as a high-level reference architecture for investigation in cloud environments?
- (2) Are there any major aspects that are missing in this architecture/model?
- (3) In your opinion, is this model possibly a good foundation and first step for cloud forensic standardization? If yes, are there any aspects that can be further improved? If no, why?

All 8 experts answered yes to the first question. In the comments, one expert mentioned that the matrix table is particularly useful for identifying what role a cloud provider/auditor can play, especially on the pro-active side.

When answering the second question, one expert suggested to include 'data access/control' in the case management sub capabilities. The reason is many forensic programs offer a review piece that maybe hosted in the Cloud.

All 8 experts answered yes to the third question. One expert mentioned that the cloud actors and roles need to be more clearly defined, but developing CFMM based on the concept of CMM is a good idea.

6 Sample Usage of CFMM

In this section we demonstrate usage of CFMM through three simple analyses.

6.1 Building Investigative Procedures

We take re-active data collection as an example, take out relevant components of the CFIA, and discuss the following three investigative scenarios and procedures.

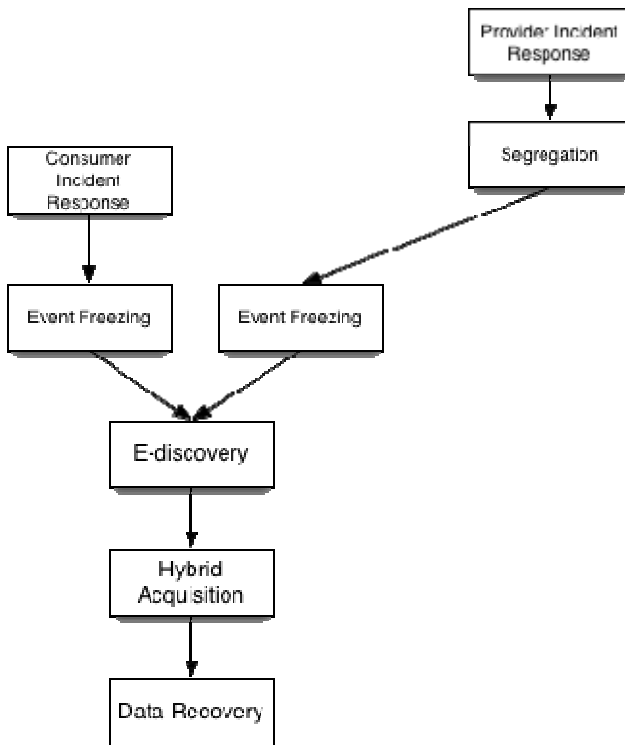


Fig. 3. Re-active Collection Scenario 1

Fig 3 describes the scenario when only one consumer and its provider are involved in an (internal or external) investigation case. In many cases, this scenario is initiated from the consumer side. The consumer first starts the incident response procedure,

and then makes an attempt to freeze the “event” on the consumer side (event freezing is a high-level forensic cloud forensic capability, which will be discussed in the next chapter), at the mean time the provider triggers its incident response to the same incident, segregates resources of the consumer in question, makes an attempt to freeze the “event” for that particular consumer after the segregation. In the next step the consumer and the provider should coordinate forensic capabilities to carry out e-discovery, hybrid acquisition and data recovery to collect re-active forensic artifacts according to the order of volatility in the cloud environment they share. Event freezing as a sophisticated capability is not possible until a mature level of cloud forensics capability, in which case the consumer and the provider should coordinate efforts immediately after incident response.

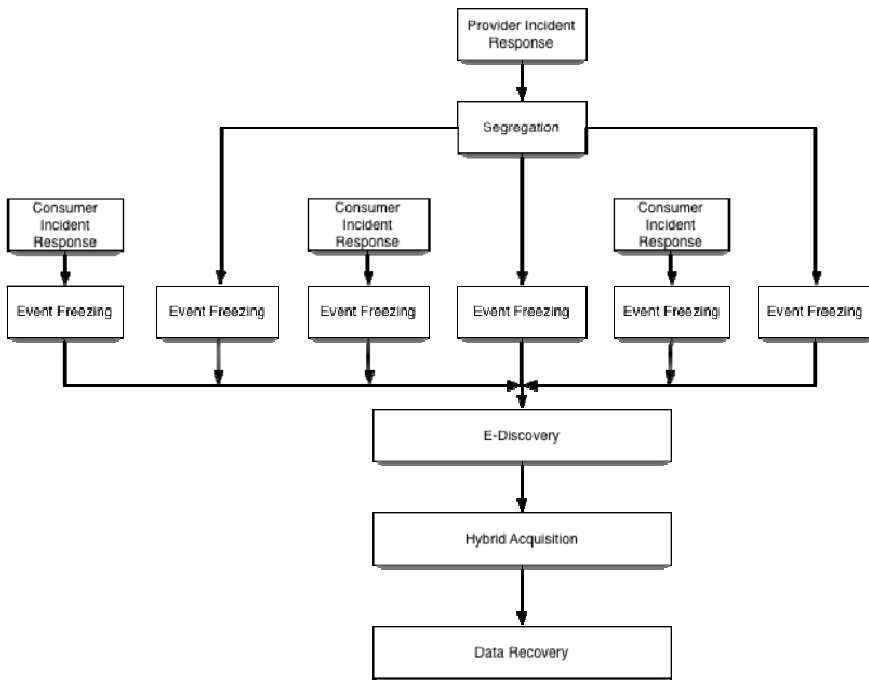


Fig. 4. Re-active Collection Scenario 2

Fig 4 describes the scenario when one provider and more than one of its consumers are involved in an (internal or external) investigation. In many cases, this scenario is initiated from the provider side. The provider first starts its incident response procedure, segregate resources for the consumers in question, and makes an attempt to freeze the “events” for those consumers. At the mean time, various consumers are notified with the same

incident, initiate their incident response procedures respectively, and make attempts to freeze their “events” accordingly as the provider. In the next step, the provider and the consumers coordinate their forensic capabilities to carry out e-discovery, hybrid acquisition and data recovery to collect re-active forensic artifacts according to the order of volatility in the cloud environment they all share.

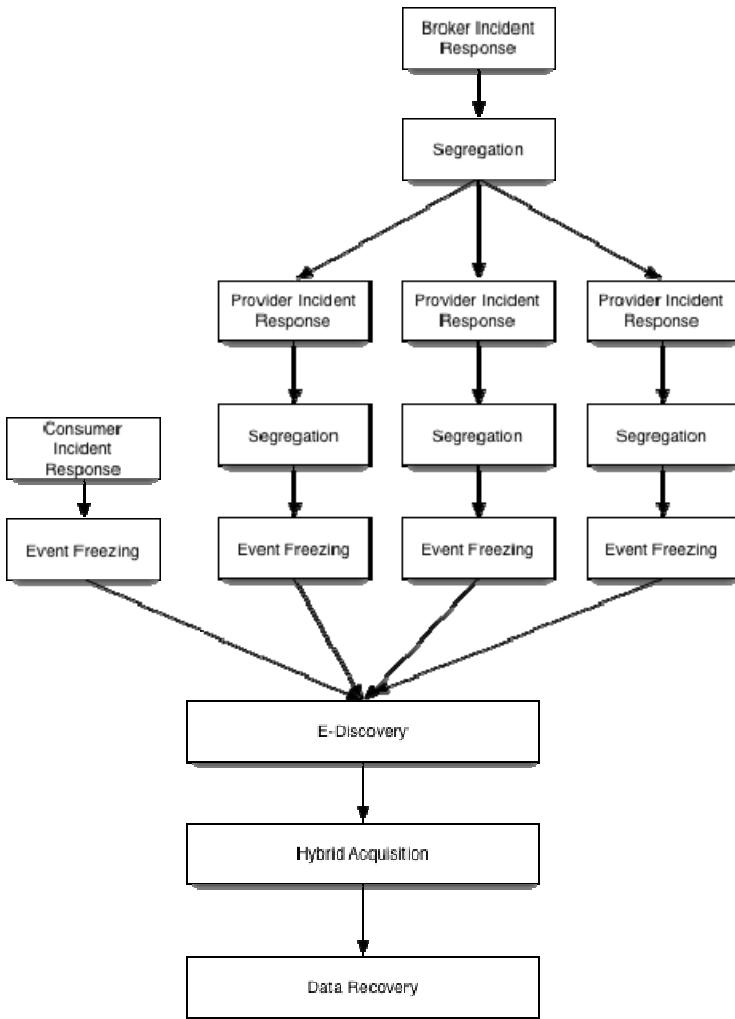


Fig. 5. Re-active Collection Scenario 3

Fig 5 describes the third scenario when one consumer and its broker are involved in an (internal or external) investigation. The broker is coordinating services provided by multiple providers and these providers are often hidden from the consumer. In this case, the broker starts its incident response procedure, segregate resources for the consumer in question, notify the providers for that consumer, the providers starts their incident response procedures respectively, segregates resources for that consumer, and make attempts to freeze the “event” for that consumer. At the mean time, the consumer starts its incident response procedure, makes an attempt to freeze the “event”. In the next step, broker coordinates its providers to aggregate forensic capabilities with the consumer’s forensic capabilities to carry out e-discovery, hybrid acquisition, and data recovery to collect re-active forensic artifacts according to the order of volatility in the cloud environment they all share.

6.2 Comparing Forensic Capabilities of Cloud Offerings

In this section we take several capabilities specified in the CFCM as examples to compare forensic capability of cloud offerings from four major providers, i.e. Amazon Web Services (Amazon 2011), Google Apps (Google 2011), Force.com (Salesforce.com 2012), and Windows Azure (Kaufman and Venkatapathy 2010), and identity current capabilities that can be utilized or leveraged for investigative purposes. The results are shown in Table 2-6.

Table 2. Encryption in Transit

Provider	Capabilities
Force.com	End-to-end TLS/SSL encryption
Windows Azure	Critical internal communications are protected using SSL encryption
Amazon CloudFront	HTTPS can be configured for all requests
Amazon	All requests are HMAC-SHA1 signed in Amazon Elastic MapReduce, CloudFront, Auto Scaling, CloudWatch, and Simple Storage Service (Amazon S3)
Google Apps	Google Apps for Business and Google Apps for Education: offer domain administrators the ability to force all users in their domain to use HTTPS

Table 3. Encryption at Rest

Provider	Capabilities
Force.com	Customer passwords stored after applying MD5 hash function; supports the encryption of field data in custom fields.
Windows Azure	.NET Cryptographic Service Providers (CSPs) can be integrated to provide AES algorithms, MD5 and SHA-2 hash functionality, RNGCryptoServiceProvider class, Straightforward key management methods, etc.
Amazon	Amazon S3, EBS, Amazon Simple DB, Amazon Simple Queue Service (Amazon SQS) recommend consumers to encrypt sensitive data before uploading
Google Apps	Data chunks are not stored in clear text so that are not humanly readable

Table 4. Authentication

Provider	Capabilities
Force.com	Two-factor authentication processes; Federated authentication single sign-on; Delegated authentication single sign-on
Windows Azure	Windows Live ID (one of the longest-running Internet authentication services available); Subscription based; SMAPI Authentication
Amazon Web Services	AWS IAM enables a customer to create multiple users and manage the permissions for each of these users within their AWS Account. A user is an identity (within a customer AWS Account) with unique security credentials that can be used to access AWS Services. AWS MFA allows Multi-factor authentication
Google Apps	Service-to-service authentication; x509 host certificates; Two factor authentication mechanisms; Optional two step verification (a built-in two-factor authentication capability);
	Single Sign-On (SSO) with Google Apps for Business, Google Apps for Education, and Google Apps for ISPs

Table 5. Data recovery

Provider	Capabilities
Amazon Web Services	<p>Amazon S3, Amazon Simple DB: removal of the mapping from the public name to the object starts immediately, and is generally processed across the distributed system within several seconds. Once the mapping is removed, there is no remote access to the deleted object. The underlying storage area is then reclaimed for use by the system.</p> <p>Amazon Relational Database Service (Amazon RDS): once an Amazon RDS DB Instance deletion API is run, the DB Instance is marked for deletion and once the instance no longer indicates 'deleting' status, it has been removed. At this point the instance is no longer accessible and unless a final snapshot copy was asked for, it cannot be restored and will not be listed by any of the tools or APIs.</p> <p>Amazon S3, Amazon SimpleDB, Amazon Elastic Block Store (EBS): data is redundantly stored in multiple physical locations as part of normal operation of those services at no additional charge.</p> <p>Amazon S3 and Amazon SimpleDB store objects multiple times across multiple Availability Zones on the initial write and then actively doing further replication in the event of device unavailability or detected bit-rot.</p> <p>Amazon EBS stores replication within the same Availability Zone.</p> <p>Amazon S3 regularly verifies the integrity of data stored using checksums, and calculates checksums on all network traffic to detect corruption of data packets when storing or retrieving data. If corruption is detected, it is repaired using redundant data.</p>
Windows Azure	Windows Azure's Storage subsystem makes customer data unavailable once delete operations are called. All storage operations including delete are designed to be instantly consistent. Successful execution of a delete operation removes all references to the associated data item and it cannot be accessed via the storage APIs. All copies of the deleted data item are then garbage collected. The physical bits are overwritten when the associated storage block is reused for storing other data, as is typical with standard computer hard drives.
Google Apps	After a Google Apps user or Google Apps administrator deletes a message, account, user, or domain, and confirms deletion of that item (e.g., empties the Trash), the data in question is removed and no longer accessible from that user's Google Apps interface. The data is then deleted from Google's active servers and replication servers. Pointers to the data on Google's active and replication servers are removed. De-referenced data will be overwritten with other customer data over time. Google Apps data is replicated to multiple systems within a data center, and also replicated to a secondary data center.

Table 6. Evidence Destruction

Provider	Capabilities
Amazon Web Services	AWS uses the techniques detailed in DoD 5220.22-M (National Industrial Security Program Operating Manual) or NIST 800-88 (Guidelines for Media Sanitization) to destroy data as part of the decommissioning process. If a hardware device is unable to be decommissioned using these procedures, the device will be degaussed or physically destroyed in accordance with industry-standard practices.
Google Apps	When retired from Google's systems, disks containing customer information are subject to a data destruction process before leaving Google's premises. First, policy requires the disk to be logically wiped by authorized individuals. using a full write of the drive with all zeroes (0x00) followed by a full read of the drive to ensure that the drive is blank. Then, another authorized individual is required to perform a second inspection to confirm that the disk has been successfully wiped. These erase results are logged by the drive's serial number for tracking. Finally, the erase drive is released to inventory for reuse and redeployment. If the drive cannot be erased due to hardware failure, it must be securely stored until it can be destroyed. Each facility is audited on a weekly basis to monitor compliance with the disk erase policy.

From this comparison analysis we also discovered the following additional capabilities provided by several cloud offerings that worth noticing.

Event reconstruction: Amazon S3 Versioning enables customers to preserve, retrieve, and restore every version of every object stored in Amazon S3 bucket. With Versioning, customer can easily recover from both unintended user actions and application failures. By default, requests will retrieve the most recently written version. Older versions of an object can be retrieved by specifying a version in the request.

Multi-jurisdiction: Windows Azure allow all customers choose where their data is stored. Data in Windows Azure is stored in Microsoft datacenters around the world based on the geo-location properties specified by the customer using the Windows Azure Portal.

6.3 Analyzing Standardization Gaps

In this section we list major international cloud standardization working projects that are relevant to forensics capabilities and can be venues to bridge standardization gaps for cloud forensic maturity.

On an architectural and matrix level, the NIST Cloud Computing Security Working Group (NCC-SWG) (NIST 2012) and the Cloud Security Alliance Cloud Control Matrix (CCM) (CCM 2012) are the best fits for forensic related standardization efforts.

On interoperability issues, the Standard for Intercloud Interoperability and Federation (SIIF) being developed by IEEE P2302 InterCloud Working Group (IEEE 2012) is the best fit for defining interoperability capability for cloud forensics.

On interfacing capability provided by cloud management interfaces, the DMTF Cloud Management Working Group (DMTF 2011) is addressing requirements for the management interfaces between the cloud service consumer/developer and the cloud service provider, which can be leveraged for forensic interfaces.

On evidence management, SNIA (Storage Networking Industry Association) Cloud Storage Security working group (SNIA 2011) is developing a standard called Cloud Data Management Interface (CDMI), where basic evidence management requirements can be included and forensic interfaces can be considered.

7 Conclusions and Future Work

In this paper we present a shortened version of the Cloud Forensic Maturity Model and its two inter-related parts, i.e. the Cloud Forensic Investigative Architecture, and the Cloud Forensic Capability Matrix. According to initial evaluation and feedback, experts and practitioners agree that this is a good foundation and first step for cloud forensic standardization. We are still actively collecting use cases to validate and refine the model. We are also working on a detailed mapping of the Cloud Forensic Three-Dimensional Model to the CFMM to analyze the interactions and overlap of legal, technical and organizational key criteria to inspire more inter-disciplinary research approaches.

References

- Amazon, Amazon Web Services: Overview of Security Processes (2011)
- CCM (2012), <https://cloudsecurityalliance.org/research/ccm/> (retrieved on July 7, 2012)
- DMTF, Cloud Management WG Charter v1.1 – (May 1, 2011), <http://members.dmtf.org/apps/org/workgroup/cmwg/> (retrieved on June 26, 2012)
- Eucalyptus Systems, Eucalyptus 3.0.1 Administration Guide (2012)
- Google Inc., Security Whitepaper: Google Apps Messaging and Collaboration Products (2011)
- Kaufman, C., Venkatapathy, R.: Windows Azure Security Overview (2010)
- Salesforce.com, Inc., Security Implementation Guide (2012)
- Ruan, K., Baggili, I., Carthy, J., Kechadi, T.: Survey on cloud forensics and critical criteria for cloud forensic capability: a preliminary analysis. *Journal of Network Forensics* (2011B)
- Ruan, K., Baggili, I., Cathy, J., Kechadi, T.: Cloud forensics definitions and critical criteria for cloud forensic capability: an analysis of survey results. *Digital Investigation* (2012) (under review)
- Ruan, K., Carthy, J., Kechadi, T., Crosbie, M.: Cloud Forensics. In: Peterson, G., Sheno, S. (eds.) *Advances in Digital Forensics VII*. IFIP AICT, vol. 361, pp. 35–46. Springer, Heidelberg (2011a)
- IEEE, ICWG/2302 WG – Intercloud WG (ICWG) Working Group (2012), http://standards.ieee.org/develop/wg/ICWG-2302_WG.html (retrieved on July 6, 2012)
- SNIA, Information Technology – Cloud Data Management Interface (CDMI) Version 1.0.1 (September 15, 2011)
- NIST, Cloud Security (2012), <http://collaborate.nist.gov/twiki-cloud-computing/bin/view/CloudComputing/CloudSecurity> (retrieved on July 7, 2012)
- Paulk, M.: *Capability Maturity Model for Software*. John Wiley & Sons (1993)

Identifying Remnants of Evidence in the Cloud*

Jeremy Koppen, Gerald Gent, Kevin Bryan, Lisa DiPippo, Jillian Kramer**,
Marquita Moreland***, and Victor Fay-Wolfe

University of Rhode Island,
Kingston, RI USA
{bryank, dipippo, wolfe}@cs.uri.edu

Abstract. With the advent of cloud computing, law enforcement investigators are facing the challenge that instead of the evidence being on a device that they can seize, the evidence is likely located in remote data centers operated by a service provider; and may even be in multiple locations (and jurisdictions) across the world. The most practical approach for an investigator when cloud computing has been used is to execute a warrant that requires the service provider to deliver the evidence. However, to do this, the investigator must be able to determine that a cloud application was used, and then must issue a warrant with reasonable scope (e.g. the subject's username at the cloud provider, the name of the documents, the dates accessed, etc). Fortunately, most cloud applications leave remnants (e.g. cached web sites, cookies, registry entries, installed files, etc) on the client devices. This paper describes the process for identifying those remnants and parsing them to generate the data required by law enforcement to form warrants to cloud service providers. It illustrates the process by obtaining remnants from: Google Docs accessed by Internet Explorer, Dropbox, and Windows Live Mesh.

Keywords: cloud computing, cloud forensics, digital forensics.

1 Introduction

Cloud computing, where applications and data storage are provided as services to users via the Internet, is becoming more and more prevalent - and because of it, law enforcement investigators are facing new challenges in obtaining evidence. Instead of the evidence being on a device that they can seize, the evidence is likely located in a data center at a service provider that is often not geographically easily accessible. In fact, the data may be stored in multiple physical locations (and jurisdictions) across

* This work was supported by a grant from the U.S. Department of Justice's National Institute of Justice Electronic Crimes Research and Development program – Grant # 2011-FD-CX-K011.

** Supported as a National Science Foundation Research Experience For Undergraduate student researcher from Villanova university under grant NSF 1004409.

*** Supported as a National Science Foundation Research Experience For Undergraduate student researcher from Purdue university under grant NSF 1004409.

the world. The problem is particularly acute for law enforcement investigators from smaller organizations, where extensive traveling to obtain evidence is not feasible. Furthermore, the volume of data kept by these service providers is so vast and the data is so complex that it is often impractical for an investigator armed with a warrant to extract the evidence from the data centers of most service providers, even if he/she were physically present.

The most practical approach for State and local law enforcement is to execute a warrant through the service provider's Keeper of Records that requires the service provider to deliver the evidence. This mitigates the issues of having to travel to remote and multiple physical locations, and issues of needing to understand data formats to find the evidence in vast data storage centers. Although there are other potential problems with this approach, such as uncooperative service providers, a warrant to a service provider to acquire evidence is the best means available to law enforcement when cloud applications have been used by a suspect.

However, there are several substantial barriers to an investigator obtaining evidence from cloud service providers. First, the investigator must be able to determine that a cloud application was used. Typically all they have to work from are seized devices (computers, phones, etc); and determining that the suspect was using a cloud application by examining a seized device is often very difficult. Second, even if law enforcement seizes the suspect's devices, the suspect may use other devices/means to access his/her cloud data to modify or delete it. This makes it essential that law enforcement *quickly* deduce that cloud computing was used so that they can issue preservation orders to the service provider. Third, to meet practicality considerations and restrictions on scope, preservation orders and warrants must be specific and indicate details such as the cloud application used, the user account, the dates it was used, and files of interest - information that can be even more difficult for the investigator to obtain from the suspect's devices. Finally, cloud computing is in its "Wild West" stage where new cloud applications are coming and going daily. Keeping up with which cloud applications the suspect might have used and where on devices that the applications keep the data necessary for a preservation order and warrant is impractical for State and local law enforcement investigators.

This paper reports our research to determine what remnants are left on devices (computers, phones, iPads, etc.) and how to collect and present those remnants necessary for law enforcement to meet restrictions on scope in warrants and preservation orders served to the cloud service provider. These remnants of cloud applications that are left on devices include data found in file system data structures, cached web sites, cookies, index.dat entries, registry entries, and several other places on devices used by the suspect. This includes information such as the cloud applications used, usernames at the service provider, dates and times that the cloud applications were used, and cloud application document names involved.

Section 2 provides background on cloud computing and related forensics tools. Section 3 describes the process used to identify and find remnants. Section 4 provides details on the initial cloud applications that we used for proof of concept. Section 5 summarizes and describes the next steps in creating a full, robust, tool to support law enforcement in investigations that involve evidence in The Cloud.

2 Background

This section presents background on cloud computing and on related digital forensics tools that have established the paradigm of focusing on specific classes of evidence – the paradigm on which we base the notion of searching specifically for cloud remnants.

2.1 Cloud Computing

According to the latest definition from NIST, *cloud computing* is "a model for enabling convenient, on-demand network access to a shared pool of configurable resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." [1].

NIST's cloud model promotes availability and is composed of five essential characteristics:

- On-demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service

Most cloud computing infrastructures consist of services delivered through common centers and are built on servers. Clouds often appear as single points of access for users' computing needs. The major cloud service providers include Amazon [2], Apple [3], Dropbox [4], Rackspace Cloud [5], Salesforce [6], Skytap [7], Microsoft Windows Live [8] and Google [9]. Some of the larger IT firms that are actively involved in cloud computing are Huawei [10], Cisco [11], Fujitsu [12], Dell [13], Hewlett Packard [14], IBM [15], VMWare [16], Hitachi [17] and NetApp [18]. A more complete list is provided in Section 4.1.

The fundamental concept of cloud computing is that the computing is "in the cloud" i.e. the processing (and the related data) is not in a specified, known or static place(s). In fact, data may be stored in many different locations. This is contrary to what law enforcement investigators are more used to: where processing takes place on a client device or in one or more specific servers that are known. Although an empowering and "freeing" concept for users, the removal of data to the cloud presents problems for law enforcement who often need to find the place(s) of the evidence.

Cloud Architecture. NIST's cloud general architecture that describes the delivery of cloud computing, typically involves multiple cloud components communicating with each other over application programming interfaces, usually web services.

The two most significant components of cloud computing architecture are known as the *front end* and the *back end*. The front end, also called the *cloud client*, is the

part seen by the user. This includes the user's device (computer, phone, etc) and the applications used to access the cloud, such as a web browser. The back end of the cloud computing architecture is the 'cloud' itself, comprising various computers, servers and data storage devices networked together.

Cloud services can be delivered at three levels: Service, Platform and Infrastructure, and any particular cloud application can provide one or more of these services.

Cloud application services, also known as "*Software as a Service (SaaS)*," deliver software over the Internet, eliminating the need to install and run the application on the customer's own devices and simplifying maintenance and support. People tend to use the terms 'SaaS' and 'cloud' interchangeably, when in fact SaaS is just one type of cloud service available. Key characteristics of SaaS include:

- Network-based access to, and management of, commercially available (i.e., not custom) software;
- Activities that are managed from central locations rather than at each customer's site, enabling customers to access applications remotely via the Web;
- Application delivery that typically is closer to a one-to-many model (single instance, multi-tenant architecture) than to a one-to-one model, including architecture, pricing, partnering, and management characteristics;
- Centralized feature updating, which obviates the need for downloadable patches and upgrades.

Cloud platform services or "*Platform as a Service (PaaS)*" deliver a computing platform and/or solution stack as a service, often consuming *cloud infrastructure* and sustaining *cloud applications*. It facilitates deployment of applications without the cost and complexity of buying and managing the underlying hardware and software layers.

Cloud infrastructure services, also known as "*Infrastructure as a Service (IaaS)*", delivers computer infrastructure - typically a platform virtualization environment - as a service. Rather than purchasing servers, software, data-center space or network equipment, clients instead buy those resources as a fully outsourced service. Suppliers typically bill such services on a utility computing basis and amount of resources consumed (and therefore the cost) will typically reflect what the client uses.

Cloud Data. The cloud model has been criticized by privacy advocates for the greater ease with which the companies hosting the cloud services can control and monitor the communication and data stored between the user and the host company [19, 20]. Regulations governing cloud data storage include FISMA [21], HIPAA [22] and SOX [23], the credit card industry's PCI DSS [24], and SAS 70 Type II certification [25]. In fact many research projects on "cloud forensics" tend to focus on privacy issues – not the pragmatic issues of how to perform forensics with a valid warrant when the suspect uses cloud computing.

Significance of the Cloud. According to an April 2011 forecast by Forrester Research, the volume of the global Cloud computing market will reach \$241 billion by the year

2020, from just \$40.7 billion in 2009 [26]. Similarly, a report from 451 Market Monitor predicts a 24% compound annual growth rate (CAGR) of Cloud computing revenue between 2010 and 2013 [27]. Cisco reported in 2011 that global Cloud IP traffic will increase twelvefold in the five years from 2010 to 2015 [27]. This prediction indicates an overall CAGR of 66% over that time period. According to the IDC, the revenue of worldwide public Cloud services has a growth rate which exceeds that of the global IT market as a whole by a factor of four [28]. This rapidly increasing popularity is becoming a considerable contribution to the IT market's overall growth.

Law Enforcement Investigations and Cloud Computing. Given the rapid growth in the use of Cloud Computing, the Cloud will likely be the next evolution in the history of computing, following in the footsteps of mainframes, minicomputers PCs, servers, smart phones, and so on. It could radically change the way enterprises manage information technology. As such, cloud computing has the potential to be the next disruptive technology that prevents law enforcement from performing effective digital forensics [29].

2.2 Digital Forensics Tools

The concept of identifying specific classes of remnants that this project uses is consistent with a trend in digital forensics tools - the use of focused special-purpose tools that integrate with an overall forensics tool suite. These special-purpose tools focus on a specific class of evidence, gather it, and present it either as a report or as data suitable to be imported into a forensic analysis tool such as FTK [30], EnCase[31], or X-Ways[32]. Some of these focused tools include:

- *P2P Marshal* - ATC-NY developed P2P Marshal [33] to automatically analyze peer-to-peer (P2P) usage on disk images (Forensic Edition) and live systems (Field Edition). It detects what P2P client programs are, or were, present, extracts configuration and log information, and displays the shared (uploaded) and downloaded files. It also includes extensive search capabilities and a thumbnail browser and image viewer. The tool produces reports in RTF, PDF, and HTML formats and runs on Windows machines.
- *Mac Marshal* - ATC-NY developed Mac Marshal [34] to analyze Mac OS X file system images. It scans a Macintosh disk image, automatically detects and displays Macintosh and Windows operating systems and virtual machine images, then runs a number of analysis tools on the image to extract Mac OS X-specific forensic evidence written by the OS and common applications. Mac Marshal Forensic Edition runs on an investigator's Mac workstation to analyze a disk image. Mac Marshal Field Edition runs on a Mac target machine from a USB drive. It extracts volatile system state data, including a snapshot of physical RAM. Mac Marshal follows forensic best practices and maintains a detailed log file of all activities it performs. It produces reports in RTF, PDF, and HTML formats, and runs on Mac OS X-based analysis machines.

- *Cyber Marshal Dropbox Reader* – ATC-NY released this collection of command line tools to parse Dropbox configuration and cache files stored as SQLite databases. The reader works with Dropbox Cloud storage software on Windows, Macintosh, or Linux systems [35].
- *Gargoyle* - Wetstone Technologies developed Gargoyle [36] to detect malware on a computer. It has data sets of remnants indicative of over 10,000 programs, most of them being malware. It can mount disk images, presents reports of the likely software it finds, and exports to formats for import to EnCase, FTK, and spreadsheets.
- *NetAnalysis* - Digital Detective developed the NetAnalysis tool [37] to collect and report browser remnants (cached web sites, history, cookies, etc) from disks and disk images in an easy to understand and easy to use format.
- *Internet Evidence Finder* – JADsoftware Incorporated created this tool to scan a computational device and identify web browser artifacts. It is designed to find existing and deleted data that has been left behind by Internet communications. The reporting style allows the user to search, filter, and bookmark results [38].
- *RedLight* - Our group at the URI Digital Forensics Center developed RedLight [39] to detect pornography in files on mounted drive or drive image. RedLight is based on how law enforcement investigates a case - by finding likely pornography in images very quickly, allowing visual confirmation by the investigator through a display of thumbnails, and then exporting selected images, reports, and hash sets suitable for importing into EnCase, FTK, and X-Ways.

All of these tools, and many other useful digital forensics tools, are designed to search computers and/or disks or disk images, find a particular class of evidence (e.g. peer-to-peer application remnants, malware remnants, browser remnants, etc), and report it to the investigator and/or allow its import into a major analysis tool.

There are no forensics tools specifically meant to collect and report remnants of Cloud applications – this research is the first step towards that goal.

3 Finding Cloud Remnants

This section presents our approach to determining what remnants cloud applications leave. There are two primary classes of cloud applications: *browser-based*, and *installed components*. Browser-based cloud applications execute exclusively in the browser and thus their remnants are found in browser remnants. Installed component cloud applications require software to be installed on the device.

3.1 Cloud Remnant Locations

The remnant data sets vary based on the operating system (e.g. Windows, MacOS, Linux, Android, iOS, etc.) and for browser-based application they also depend on the browser. We have identified several key places that cloud application remnants can be found.

- *Cached web sites* - Cached web sites are the files that the browser downloads and stores on the client device when a web site is requested. Cached web sites will often indicate the cloud application used, the dates it was used (file modified, created, access times), the username (which often appears in a "logged on" message on the cloud application web pages), account names and kind of account, and more. Most cloud applications display this information in known areas of the web page using known HTML-based formatting tags that the resulting cloud analysis tool can search for. How browsers cache web sites varies based on the browser, so the remnant data set will have to be developed differently for all prominent browsers, but the web page content that it looks for will typically be the same.
- *Web history* - Web sites visited are often tracked in web history files, index.dat, files and places like the Windows registry. The URLs in the web history indicate that the user likely visited the web site (e.g. to use a cloud application), and occasionally show the HTTP parameters in the URL which can indicate things like user IDs and actions to be performed.
- *Cookies* - Most cloud applications use cookies to track user activity and facilitate ease of re-connection. The format of most cookies requires some decoding, but the formats are standard. Even encoded proprietary information such as user identifiers that may be meaningless when pulled off a device can be provided to the service provider in the warrant for interpretation and decoding to useful evidence.
- *Installed files and registry entries* - Installed component cloud applications, such as Dropbox [4], require the installed client software to communicate with the cloud application via the Internet. Presence of these installed components indicate that the cloud application was possibly used, and often contain configuration files and registry entries with specifics such as usernames, account names and types, IP addresses and port numbers of the service, and history of use. Some of these configuration file formats and registry keys may be encoded and proprietary, but they can be obtained and presented to the service provider in the warrant for decoding to useful evidence.
- *Modified files and registry entries* - Some web applications, like online storage applications, download files as part of their on-going operation. The modified, accessed and created dates of these files and how they were created (e.g. by a service) can be important in some investigations.

3.2 Identifying Cloud Signature Remnants

To identify cloud application remnant data sets we first identify which files the cloud applications typically install and/or access, then we use software to parse these files to extract specific data that is of relevance for forming warrants.

Determining Modified Files. We use two primary techniques for compiling a list of files that are modified by cloud applications: *hash sets* and *monitoring tools*.

Hash sets are lists of MD5 hash values of files that can be created by a software tool, such as AccessData's Forensic Toolkit (FTK) [30], scanning each file on the device and recording the MD5 hash value of that file. We used FTK to create a hash set of all files on a VMWare virtual machine (VM) that uses the target operating system (e.g. Windows 7). We then launch a cloud application on that virtual machine. We again take hash sets of the entire system after performing various tasks in the application including, but not limited to: connection, logon, using the application, and saving data on the cloud storage. Differences in the hash sets indicate the files that have been changed in the respective steps of using the cloud application. Although useful, this technique can yield a great deal of changed files, all of which have to be further inspected to ascertain their relevance, if any, to the signature of the application.

To refine the results of hash set monitoring, we used several special-purpose commercial monitoring tools. We used two categories of monitoring tools – *dynamic monitoring tools* and *static monitoring tools*.

Dynamic monitoring tools monitor an executing system and report live results. Our primary dynamic monitoring tool was SysInternals/Microsoft's Process Monitor [40], which is a tool that analyzes a process and reports on the CPU utilization, file I/O, Registry operations, network operations, and memory statistics. By observing its live reports during the various phases of using a cloud application, we were able to determine what system resources the cloud application used. For instance, we were able to observe the registry keys locked by Windows Live Mesh while it executed – the presence of these registry keys then being an element of the remnant data set for that application.

The primary static monitoring tool that we used for Windows systems is InCtrl5 [41] that monitors the state of the system before and after installation of software. We used InCtrl5 to establish what files installed-component software placed on the system, what registry entries they inserted, and what existing files they modified. Other tools such as Total Uninstall [42], and Spy Me [43] provide similar insight.

Parsing Files. The above techniques yield which files are added, and modified, by cloud applications. These files then need to be parsed to extract the specific data required by law enforcement.

Parsing requires searching the files for known keywords or substrings and then further processing of the text around those substrings for known parameters. For instance, when Google Docs is used on a Windows 7 computer via Internet Explorer, the substring *docs.google.com/document/create?* appears in an index.dat file and possibly the web history. The text following that string can contain the file name of the created document. In some circumstances a key string is first found and then subsequent searching and parsing is necessary. For instance, the presence of the string *docs.google.com/* in the index.dat file can indicate that the HTML code of the cached web sites on the device should be parsed for the specific HTML that Google Docs uses to display the username on all Google Docs pages. The parser then extracts the

username from the HTML code as a use remnant for law enforcement to use in their warrant.

Note that there is a substantial manual process that involves using these tools in a laboratory setting to pinpoint which changes to the system are made by the use of a cloud application (and which are not), and what relevant information those changes might yield. This research is the first step in establishing a process by which cloud remnant data sets can be added to a tool that will support law enforcement investigations.

4 Results

We constructed partial data sets for some important and representative cloud applications as a proof of concept. This section describes data sets for *Google Docs* as an example of browser-based applications, and of *Dropbox* and *Windows Live Mesh* as examples of installed component applications.

4.1 Browser-Based Cloud Applications

To demonstrate our techniques for searching browser-based cloud applications, we investigated Microsoft's Internet Explorer web browser versions 6, 7, 8, and 9 on Windows XP and Windows 7. All of these versions of Internet Explorer implement browser data in the *index.dat* file [44] structure from version 5. Overall, all of the Internet Explorer versions share the same *index.dat* structure, but the location of the temporary Internet files may change. Using the hash set technique of Section 3.2 we were able to determine the *index.dat* file used for Google Docs.

We wrote a parsing program to search the *index.dat* file to extract the last accessed time, the URL visited, and the filename based on keywords. The last accessed time indicates when the website was visited by the user. The URL visited stores the ASCII string representation of the URL that was visited, and the filename stores the name of the file that was accessed and temporarily stored by the browser on the physical device. The filename may not be set in all of the entries of an *index.dat* file because not all entries correspond to a file that is being temporarily stored on the computing device. As noted before, Internet Explorer versions 6, 7, 8, and 9 all implement the same data structure to store temporary Internet files. These applications implement Client URL Cache Version 5.

In every *index.dat* file there exists a 32-bit integer value at offset 0x20, which indicates where the entries are being stored on the file. Once the parsing program moves to the beginning of the entry storage, it starts to search for any valid entries.

An entry can be one of three types:

- REDR - a browser redirect.
- LEAK - an error that was generated. Usually this is generated due to an error occurring during the deletion operation of a URL entry.
- URL – URL that the user visited.

The parsing program only scans for entries marked LEAK or URL. It will not search for REDR entries because the final URL that the user would arrive at would still be indicated as a URL entry. After parsing the data, the parsing program moves to the URL and Filename offsets to parse the ASCII String representation of the bytes located in these data fields. After all of the data for the entry has been parsed, the parsing program determined if the entry has any evidentiary value. If the entry does contain evidentiary value, it is added to the set of data to report. The tool then continues searching for another iteration of a URL or LEAK entry.

Below in Figures 1 and 2, a URL entry is shown after it has been parsed by a custom WinHex template.

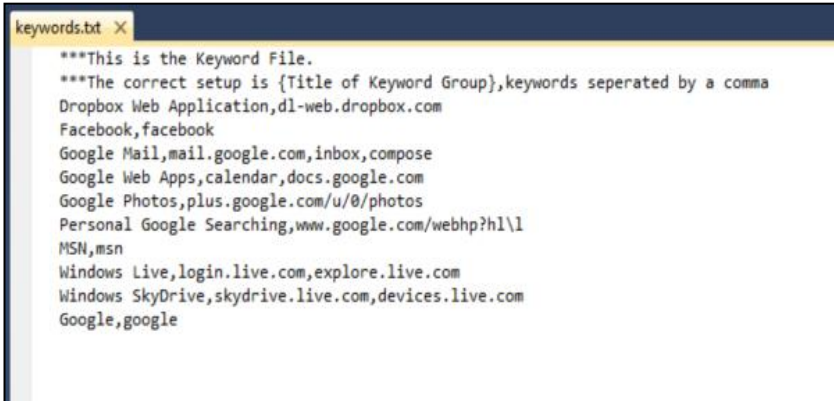
Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00005100	55	52	4C	20	02	00	00	00	A0	78	09	E5	D2	ED	CC	01	URL
00005110	A0	78	09	E5	D2	ED	CC	01	51	42	70	02	00	00	00	00	x àÔiï QBp
00005120	97	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00005130	60	00	00	00	68	00	00	00	FE	00	10	10	8C	00	00	00	h p
00005140	01	00	10	00	00	00	00	00	00	00	00	00	00	00	00	00	
00005150	52	40	5B	02	04	00	00	00	00	00	00	00	52	40	5B	02	R@[R@[
00005160	00	00	00	00	EF	BE	AD	DE	43	6F	6F	6B	69	65	3A	6B	i%-bCookie:k
00005170	6F	70	70	65	6E	40	77	77	77	2E	6D	69	63	72	6F	73	oppen@www.micros
00005180	6F	66	74	2E	63	6F	6D	2F	00	BE	AD	DE	4D	4C	54	4C	oft.com/ %-pMLTL
00005190	30	46	37	4E	2E	74	78	74	00	BE	AD	DE	EF	BE	AD	DE	0F7N.txt %-pi%-p
000051A0	EF	BE	AD	DE	EF	BE	AD	DE	EF	BE	AD	DE	EF	BE	AD	DE	i%-pi%-pi%-pi%-p

Fig. 1. Hex view of Index.dat File URL Entry



Fig. 2. Parsed URL Bytes

To determine if a parsed entry is of evidentiary value, it is compared against a list of user-specified keywords, such as the following: *Google Documents, Google Mail, Google Plus, Personal Google Web Searches*. Figure 3 shows an example keyword file used by the parsing program.



```

keywords.txt X
***This is the Keyword File.
***The correct setup is {Title of Keyword Group},keywords seperated by a comma
Dropbox Web Application,dl-web.dropbox.com
Facebook,facebook
Google Mail,mail.google.com,inbox,compose
Google Web Apps,calendar,docs.google.com
Google Photos,plus.google.com/u/0/photos
Personal Google Searching,www.google.com/webhp?hl\l
MSN,msn
Windows Live,login.live.com,explore.live.com
Windows SkyDrive,skydrive.live.com,devices.live.com
Google,google

```

Fig. 3. Example Investigator Keyword File

The parsing program then searches each of the entries found in an `index.dat` file to determine if any of the keywords from the user generated `keyword.txt` file is present. If any keyword matches, then the entry is added as a result for the corresponding keyword group. Once the parsing of an entire `index.dat` file is finished, the entries that contained keywords are then written out to a Keyword Group in a report file formatted in HTML. The same steps are repeated for each `index.dat` file found, with a separate HTML report for each `index.dat` file.

When we applied this technique to the remnants left by the use of Google Docs, we have found that Google Docs leaves cached web sites that can determine the application and dates/times of use:

- *Start page:* -URL: <https://docs.google.com/> -Title: Google Docs
- *File listing:* -URL: <https://docs.google.com/#all> -Title: Google Docs - All items
- *Create:* -URL: <https://docs.google.com/document/create?hl=en> -Title: create
- *New document:* -URL: https://docs.google.com/document/d/1CjOwcXrET-uaFGPzalbNPS_P6kgQNhB1whAMprwNHXs/edit?hl=en -Title: Untitled document - Google Docs
- *Save as new name (test):* -URL: https://docs.google.com/document/d/1CjOwcXrET-uaFGPzalbNPS_P6kgQNhB1whAMprwNHXs/edit?hl=en# -Title: test - Google Docs

(where the long string in the URL parameter list is constant and likely an identifier that can be part of the warrant to Google.) These are just some sample cached files. Furthermore, the Google username is stored in an `index.dat` file as well as being available by subsequent parsing of some cached web sites.

We have monitored other browser-based applications such as iCloud and Zoho. Those two, for instance, leave cookies as well as Internet history.

4.2 Installed Applications

As noted before, a client-based cloud-computing application requires the user to install a piece of client software on a physical computing device. This paper reports our results on collecting artifacts from the *Dropbox* and *Windows Live Mesh* applications. Dropbox was chosen due to its immense popularity. In April 2011, Dropbox announced they had over 25 million users (Arrington, 2011). Windows Live Mesh was chosen due to the fact that it is supported by Microsoft, and is expected to be included in future operating systems (“BUILD, 2011”). Both of these allow for a user to upload files and folders via a client application. Then these files and folders can be downloaded on any other computer (“Dropbox”, “Windows Live Mesh 2011”).

Dropbox. The main focus of our investigation of Dropbox was on Dropbox SQLite database files that are typically present and contain data that would be of evidentiary value. These two database files are named *config.db* and *filecache.db*. The *config.db* database file contains the following information regarding the user’s Dropbox account:

- Dropbox Version - The version of Dropbox that is being used
- Unique Dropbox Host ID - A unique 128 bit key pertaining to a user account
- Dropbox Path - The path on the computational device where Dropbox has mounted its virtual folder
- Dropbox Username - A string username that is specified by the user
- Recently Changed Files - List of files that were most recently changed on the Dropbox account

Table: config		New Record		Delete Record	
	key	value			
1	config_schema_version				1
2	show_bubbles				1
3	fixed_dropbox_perms				1
4	recently_changed3	(lp1			
5	host_id	f1a8a2e7b15b0ba60d8474a55a2d6c3f			
6	dropbox_path	C:\Documents and Settings\Jeremy\My Documents\Dropbox			
7	root_ns				4827460
8	email	jeremy.koppen1@gmail.com			
9	stats_dont_send_until_upgrade				
10	stats_next_report_time				1332213286.078
11	stats_next_report_id				33916433
12	stats_build	S'Dropbox-win-1.1.35'			
13	ns_p2p_key_map	(dp1...			
14	sandboxes	(lp1			
15	last_update	(F1331003687.6400001...			

Fig. 4. Dropbox Config.db

The filecache.db database file contains a table called file_journal, which contains the following information for any files stored on the Dropbox account:

- Server Path - Stores the path of the file with the server identification
- Local Filename - Stores the local filename of the file that was added to the Dropbox account
- SHA-256 Hash - Stores the SHA-256 Hash in a Base-64 encoded string
- Local Size (MB) - Stores the Local Size of the file in MB
- Modified Time (UTC) - Stores the time that the file was modified
- Created Time (UTC) - Stores the time that the file was created

id	server_path	local_filename	local_blocklist	local_size	local_mtime	local_ctime
61	4827460:/packetfilt...	Test Code Output.txt	Cq-EOPDpjhmPPG...	1685	1258497754	1329590553
62	4827460:/packetfilt...	Sample output.txt	rJ3DnT0C8ZRsoxi...	3632	1258497768	1329590553
63	4827460:/packetfilt...	PacketFilter1.vcproj	ORUNeRmC2GJQQ...	4540	1256241354	1329590553
64	4827460:/packetfilt...	stdafx.obj	KmSDgccE1f80Ze...	12601	1258413050	1329590553
65	4827460:/packetfilt...	PacketFilter1.pch	jAIOpC4rqJbe0s3...	3211264	1258413050	1329590578
66	4827460:/packetfilt...	mt.dep	3bBkxXd6JvUWaP...	66	1258497836	1329590553
67	4827460:/packetfilt...	PacketFilter1.suo	qjf4x1kFRgZdPn1...	8704	1258497840	1329590553
68	4827460:/packetfilt...	BuildLog.htm	TvqI0TSLK:kn9tig...	46118	1258497836	1329590563
69	4827460:/packetfilt...	PacketFilter1.cpp	SFccZh06NvyZxmi...	110602	1258497826	1329590564
70	4827460:/packetfilt...	PacketFilter1.exe	1L1HjBSduUQVAa...	159744	1258497836	1329590564
71	4827460:/packetfilt...	vc90.idb	_jHwVpaikDl7w1F...	183296	1258497836	1329590564
72	4827460:/packetfilt...	vc90.pdb	t53p7iIN7BM0RQH...	282624	1258497834	1329590564

Fig. 5. File_Journal Table From File_Cache.db

Our parsing program scanned every user account described in the investigator's keyword list to determine the presence of a Dropbox directory. In Windows 7 the tool scans the user's AppData directory, and in Windows XP the tool scans the user's Application Data directory. If the Dropbox directory is found, then the software parses the config.db database file by using an instance of an SQLReader class that we programmed to parse SQLite databases. After the data in the SQLite database that matches keywords from the investigator's list has been determined, it is written out to a Dropbox Results HTML webpage report. Next, the software parses the filecache.db database file and sends all of the parsed data to the same Dropbox Results HTML webpage report. This information provides law enforcement unique user identification information, along with all of the files that the user had placed on the Dropbox account for storage.

Windows Lives Mesh. Windows Live Mesh is an application that allows a user to sync multiple folders, and subsequent files, within a supported Windows operating system to a cloud storage device, which is maintained by Microsoft.

Windows Live Mesh installs programs in the folder %PROGRAMFILES%\Windows Live. and in registry entries: HKCU\Software\Microsoft\Windows Live and HKLM\Software\Microsoft\Windows Live.

We found that there are several .edb database files used by Windows Live Mesh. These files are Extensible Storage Engine, JET Blue, database files. JET Blue was created by Microsoft and implements an Indexed Sequential Access Method (ISAM), data storage approach. In order to parse this database file the parsing program will use

the ManagedEsent .NET library. This library provides the ability to load the database and extract data.

Inside of this Windows Live Mesh directory there exists a directory titled “DB”. The “DB” directory contains one subdirectory called “Device” which contains Device.edb database file. In this file there is a User table that corresponds to the account information. The User table contains all of the user specific information such as: the email address of the account used, a unique ID, the last time the account was updated, the published date of the account, and the Name that corresponds to the user’s account. This table provides a wealth of information about the user’s Windows Live Mesh account. Once all of this table’s information is parsed, it will be printed out to a table inside of an .HTML webpage for a report to the law enforcement investigator.

The “DB” directory also contains subdirectories for each user account that was accessed, which are named from a unique user GUID. This GUID can be found in the User table from the Device.edb file. This directory contains an .edb database file that is also named based off of the unique user GUID. This file contains information corresponding to the user’s files.

Inside of this user GUID database file there are several tables. Our parsing software first accesses the MeshObject table, which contains fields called “Id”. These unique Ids correspond to directories that were synced with Windows Live Mesh. Windows Live Mesh creates tables for each directory that the user synced. The tool then parses each Id found in the Mesh Object in order to detect the names of the tables that contain information regarding the files that were synced. Once this Id has been parsed, the tool opens another table that is titled “{MeshObject Id}_DataEntity_Enclosure”.

Inside of this table there exists the following information for each file or directory added:

- Filename
- Parent History
- Creation Time (UTC)
- File Last Write Time (UTC).

5 Conclusion

Many informed predictions believe that cloud computing will become the predominant way that digital data is processed and stored. As such, it will necessitate changes in law enforcement policy and practice when performing investigations with digital evidence. The results reported here in the application of techniques to determine cloud remnants, and to parse those remnants for data that is required by law enforcement investigators is the first step in developing a tool to arm investigators against the looming threat that cloud computing poses - the threat of vast amounts of digital evidence not being available in the form that investigators have been trained to handle.

Based on the work reported here, we are developing a tool, called *Cloud Signature*, that performs the parsing described in Section 4 and generates reports that are easy for

law enforcement to use to fashion preservation letters and warrants. The tool will use the remnants described in this paper as well as remnants from other browsers (we are currently integrating the parsing of the Chrome browser remnants) and eventually remnants from mobile devices, specifically iOS and Android devices. Cloud Signature is being designed so that as the data sets for more cloud applications are determined, they can easily be added helping ensure that Cloud Signature keeps pace with the rapidly evolving landscape of the Cloud.

References

1. Mell, P., Grance, T.: The NIST Definition of Cloud Computing. Information Technology Laboratory (2009), <http://www.nist.gov/itl/cloud/upload/cloud-def-v15.pdf>
2. Amazon, Inc., <http://www.amazon.com/>
3. Apple iCloud, <http://www.apple.com/icloud/>
4. Dropbox, <http://www.dropbox.com/>
5. Rackspace Cloud, <http://www.rackspace.com/cloud/>
6. Salesforce, Inc., <http://www.salesforce.com/>
7. Skytap, Inc., <http://www.skytap.com/>
8. Microsoft Corporation, Windows Live, <http://explore.live.com/>
9. Google, <http://www.google.com/>
10. Huawei Technologies Co., Ltd., <http://www.huawei.com/>
11. Cisco Systems, Inc., <http://www.cisco.com/>
12. Fujitsu, Ltd., <http://www.fujitsu.com/global/>
13. Dell, Inc., <http://www.dell.com/>
14. Hewlett-Packard Development Company, L.P., <http://www.hp.com/>
15. IBM, <http://www.ibm.com/>
16. VMware, Inc., <http://www.vmware.com/>
17. Hitachi, Ltd., <http://www.hitachi.com/>
18. NetApp, Inc., <http://www.netapp.com/>
19. Security Guidance for Critical Areas of Focus in Cloud Computing V2.1. Cloud Security Alliance (2009)
20. Mather, T., Kumaraswamy, S., Latif, S.: Cloud Security and Privacy: An Enterprise Perspective on Risk and Compliance, p. 239. O'Reilly Media (2009)
21. Federal Information Security Management ACT (FISMA) Implementation Project, <http://csrc.nist.gov/groups/SMA/fisma/index.html>
22. Health Information Privacy: The Health Insurance Portability and Accountability Act (HIPPA), <http://www.hhs.gov/ocr/privacy/>
23. The Sarbanes-Oxley Act, <http://www.soxlaw.com/>
24. PCI SSC Data Security Standards Overview, https://www.pcisecuritystandards.org/security_standards/index.php
25. SAS 70 Definition: Type II, <http://www.sas70.us.com/what-is/definition-of-sas70.php>
26. Kirilov, K.: Cloud Computing Market Will Top \$241 Billion in 2020. Cloud Tweaks (2011), <http://www.cloudtweaks.com/2011/04/cloud-computing-market-will-top-241-billion-in-2020/>

27. Columbus, L.: Roundup of Cloud Computing Forecasts and Market Estimates. A Passion for Research (2012),
<http://softwarestrategiesblog.com/2012/01/17/roundup-of-cloud-computing-forecasts-and-market-estimates-2012/>
28. IDC Cloud Research, http://www.idc.com/prodserv/idc_cloud.jsp
29. Bigsey, Cloud Computing and the Impact on Digital Forensic Investigations. ZDNet (2009), <http://www.zdnet.co.uk/blogs/cloud-computing-and-the-impact-on-digital-forensic-investigations-10012285/>
30. Access Data FTK, <http://accessdata.com/products/computer-forensics/ftk>
31. EnCase Forensic v7, <http://www.guidancesoftware.com/encase-forensic.htm>
32. X-Ways, <http://www.x-ways.net/>
33. ATC P2P Marshal, <http://p2pmarshal.atc-nycorp.com/>
34. ATC Mac Marshal, <http://macmarshal.atc-nycorp.com/>
35. ATC Cyber Marshall Dropbox Reader, <http://www.cybermarshal.com/index.php/cyber-marshal-utilities/dropbox-reader>
36. Gargoyle Investigator, <http://www.wetstonetech.com/cgi-bin/shop.cgi?view,2>
37. NetAnalysis, <http://www.digital-detective.co.uk/netanalysis.asp>
38. JADsoftware's Internet Evidence Finder, <http://www.jadsoftware.com/internet-evidence-finder/>
39. RedLight, <http://www.dfc.cs.uri.edu/redlight.php>
40. Process Monitor v3.01, <http://technet.microsoft.com/en-us/sysinternals/bb896645>
41. PCMag InCtrl5, <http://www.pcmag.com/article2/0,2817,25126,00.asp>
42. Total Uninstall, <http://www.martau.com/>
43. Spy Me, <http://www.ghacks.net/2009/03/01/software-installation-monitor/>
44. Jones, K.J.: Forensic Analysis of Internet Explorer Activity Files (2003), <http://www.mcafee.com/us/resources/white-papers/foundstone/wp-pasco.pdf>

On Improving Authorship Attribution of Source Code

Matthew F. Tennyson

Bradley University, Department of Computer Science & Information Systems, Peoria, IL, USA
mtennyson@bradley.edu

Abstract. Authorship attribution of source code is the task of deciding who wrote a program, given its source code. Applications include software forensics, plagiarism detection, and determining software ownership. A number of methods for the authorship attribution of source code have been proposed. This paper presents an overview and critique of the state of the art in the field. An independent comparative study is presented using an unprecedented experimental design and data set, as well as proposals for improvements and future work.

Keywords: authorship attribution, software forensics, plagiarism detection.

1 Introduction

In 1993, the term "software forensics" was coined to refer to the process of analyzing software – usually malicious remnants left after an attack – to identify the authors of the software in question or to at least identify characteristics of the authors [1]. The basic premise behind software forensics is that programmers generally apply a unique style to the code they write. As a result, programmers often leave "fingerprints" by embedding idiosyncratic features in their software. By identifying such features and associating them with a particular programmer, the original author of software whose author is otherwise unknown can be discovered.

The term "authorship attribution" refers simply to "the task of deciding who wrote a document" [2]. Features are generally analyzed regarding the style in which the document was written. These stylistic features might include the frequency or use of certain words, word length, word patterns, etc. Typically, documents of known authorship are used as training data, and the training results are then used to attribute an author to documents of unknown authorship. Numerous methods for authorship attribution have been proposed for natural language documents, including lexical methods, grammatical methods, and language-model methods.

Recent surveys of authorship attribution methods include those of Patrick Juola [3] and Efstathios Stamatatos [4]. Juola provides an historical context and analysis of some state-of-the-art methods in order to ultimately offer a recommendation for best practices. Stamatatos discusses the myriad applications of authorship attribution. An analysis of authorship attribution methods is also provided, which is focused on textual representation and computational requirements, providing a perspective grounded in information science.

Authorship attribution of source code, specifically, refers to the task of deciding who wrote a source code document. Authorship attribution is, therefore, a tenet of software forensics. Applications of source code authorship attribution include not only forensics investigations, but also plagiarism detection, software ownership disputes, and other similar activities.

Many researchers have contributed to the depth of knowledge regarding source code authorship attribution. Oman and Cook [5] were one of the first researchers to present a method for identifying authorship of programs based on programming style. Gray, Sallis, and MacDonell [6] introduced several metrics that can be used for authorship attribution of source code, and developed a tool called IDENTIFIED capable of extracting those metrics. Krsul and Spafford [7] performed one of the first in-depth studies of source code authorship attribution, analyzing several methods for determining authorship including discriminant analysis and several classification techniques using a tool called LNKnet. MacDonell, Gray, MacLennan, and Sallis [8] utilized neural networks, multiple-discriminant analysis, and case-based reasoning. Ding and Samadzadeh [9] utilized canonical discriminant analysis and 56 metrics to determine authorship. Lange and Mancoridis [10] utilized the similarity of histogram distributions of 18 code metrics, which were selected using a genetic algorithm. Elenbogen and Seliya [11] utilized a C4.5 decision tree. Frantzeskou, Stamatatos, Gritzalis, Chaski, and Howald [12] utilized Source Code Author Profiles (SCAP) using n-grams to represent programs and a similarity measure to determine authorship.

In 2010, Burrows [13] presented a comparative study that included most of the aforementioned methods of source code authorship attribution. The study consisted of a 10-class experiment (determining the author of a program from a set of ten candidate authors). A "leave-one-out cross validation" experimental design was used (each program in the data set was selected, in turn, as a query program while the remaining programs were used as training data). The results were measured in terms of accuracy (as a percentage of programs whose authors were correctly identified). The most effective method was found to be the Frantzeskou method [12].

In addition to presenting the comparative study, Burrows also presented a new method of source code authorship attribution. This new method was evaluated using the same 10-class experiment used to evaluate the other methods. The Burrows method performed the best on 3 out of 4 segments of the data set, while Frantzeskou performed the best on the remaining segment.

The Burrows and Frantzeskou methods are clearly state of the art in authorship attribution of source code. This paper presents an overview and critique of these methods, an independent comparative study of them using an unprecedented experimental design and data set, as well as proposals for improvements and future work.

2 Overview

Both the Frantzeskou and Burrows methods utilize n-grams to represent programs, and they both use a similarity measure to determine authorship. However, they are significantly different in the way the n-grams are formed and the specific similarity

measures that are used. How the training programs are grouped is also a key difference. These differences will be delineated in the following sections.

2.1 The Frantzeskou Method

The Frantzeskou method uses source code author profiles to characterize the programs written by each author. The concept of author profiles is derived from the work of Keselj [14]. The concept of an author profile is defined as the set of the most frequent n-grams used in all sample works by that author with their normalized frequencies. So, a profile is a set of ordered pairs (x_i, f_i) , where x_i is the i th most frequent n-gram used by that author and f_i is the normalized frequency of that n-gram. The number of n-grams in the set is L , so the set contains the L most frequently-used n-grams. Frantzeskou uses raw frequencies, rather than normalized frequencies, however. The contention is that the frequencies are not used except to sort the n-grams, so normalization is not necessary.

The n-grams are extracted at the byte level for programs in the data set, which means that all information stored in the source files are represented in the profiles – no information is lost. Whitespace, comments, every single byte saved in the source file is processed and included as n-grams in the author profiles.

The similarity measure used in the Frantzeskou method is the Simplified Profile Intersection (SPI). The SPI is simply a count of the number of n-grams that a profile and a query program have in common: $|P_A \cap P_p|$, where P_A represents the author profile and P_p represents the program profile. This simple metric is used to determine which author profile in the data set is most similar to a query program, and it is the author whose profile is most similar that is attributed to be the author. So, in essence, it is the author who frequently uses the sequences of characters that appear most frequently in the query program that is attributed to be the author.

2.2 The Burrows Method

The Burrows method uses a lossy approach for representing programs. In this method, n-grams are based on tokens. Tokens include selected operators, keywords, and white space. Programs are scanned (such that information deemed irrelevant is lost), and the token stream is broken into n-grams using a sliding window approach.

Based on empirical results, the authors of this method chose $n=6$ for the n-gram size. The similarity measure used is Okapi BM25 [15]. This measure was selected among five similarity measures that were evaluated: Okapi BM25, Cosine, Pivoted Cosine, language modeling with Dirichlet smoothing, and a metric developed specifically for source code authorship attribution called *Author1*. Through empirical testing, Okapi BM25 was found to be the most effective.

The actual approach for attributing authorship is typical for similarity-based authorship attribution methods. To determine the author of a program, that program is considered to be a query. The query is compared using a similarity measure to all of the programs in the data set. The author of the most-similar program is considered the

author of the query program. So, in essence, it is the author who wrote the program that is most similar to the query program that is attributed to be the author.

3 The Comparative Study

Although both the Burrows and Frantzeskou methods have been shown to be state of the art, this is the first independent comparative study that has been performed on them. This study consisted of a 20-class experiment. A leave-one-out cross validation experimental design was used. The results were measured as a percentage of programs correctly identified. The data set included both C++ and Java programs.

The collection of programs used in the study included a total of 7517 Java and C++ documents. The programs consisted of sample programs distributed with introductory programming and data structures textbooks. The textbooks included twenty Java textbooks and twenty C++ textbooks. Among the Java textbooks, there were no duplicate authors. Similarly, among the C++ textbooks, there were no duplicate authors. There were, however, nine textbooks that overlapped between the two languages. That is, nine textbooks were selected that had a Java edition and an equivalent C++ edition. So, there were a total of 31 unique authors represented (11 unique to the Java collection, 11 unique to the C++ collection, and 9 that overlapped between the two languages). There were 3906 documents in the C++ collection and 3611 documents in the Java collection, meaning the C++ collection had an average of 195 documents per author while the Java collection had an average of 181 documents per author.

Sample programs from programming textbooks were used, in part, to provide an accessible analog to student-submitted programs. The programs are academic in nature, varied according to the nature of the material being exemplified in each sample program, and reasonably close to "perfect ground truth." Copyright laws and reputations would prohibit plagiarism. Consistency in approach and style would be self-enforced for reasons related to both pedagogy and software engineering. Moreover, sample programs from textbooks are generally feely available and easily accessible.

The study was conducted as a series of 20-class experiments, using a leave-one-out cross validation experimental design, and the results were measured as a percentage of programs correctly identified. A 20-class experiment means that the author of each document was determined from a set of 20 candidate authors. A leave-one-out cross validation experimental design means that each program in the data set was selected, in turn, as a query program while the remaining programs were used as training data. The results being measured in terms of accuracy means that the results were measured as a percentage of programs whose authors were correctly identified.

Each experiment was conducted as follows: Every program in the data set was represented as dictated by the method being evaluated. For the Burrows method, each program was tokenized. For the Frantzeskou method, a source code author profile (SCAP) was created for each author. Each program in the data set was selected, in turn, as a query program. The author of that program was attributed according to the method being evaluated, using the remaining programs in the data set as the programs of known authorship. Each program was marked as either correctly attributed

or incorrectly attributed. The overall results of the experiment were measured as a percentage of the programs correctly attributed.

The Frantzeskou method successfully attributed 94.3% of the documents, while the Burrows method successfully attributed 89.5% of the documents. One could argue, however, that this comparison is unfair. The Burrows method inherently anonymizes the data by removing all comments and string literals. By not anonymizing the data, the Frantzeskou method has an intrinsic advantage of including in its similarity calculations information contained therein. So, the methods were also compared using anonymized documents. When using anonymized documents, the overall results for the methods were quite similar.

In the end, a total of six individual experiments were conducted: (1) the Burrows method was used to attribute the C++ documents, (2) the Burrows method was used to attribute the Java documents, (3) the Frantzeskou method was used to attribute the C++ documents, (4) the Frantzeskou method was used to attribute the Java documents, (5) the Frantzeskou method was used to attribute anonymized versions of the C++ documents, and (6) the Frantzeskou method was used to attribute anonymized versions of the Java documents. The results of the study are shown in Figure 1.

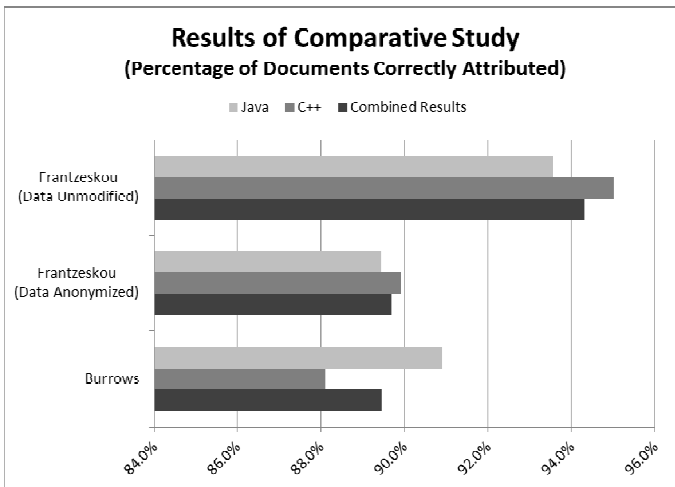


Fig. 1. Results of comparative study

Frantzeskou successfully attributed a larger percentage of the C++ documents, even after they were anonymized. However, Burrows attributed a larger percentage of the Java documents. Also, a relatively large discrepancy can be seen in the number of Java documents and C++ documents successfully attributed by Burrows. This discrepancy could be explained by the features selected for tokenization, because an utterly different set of features were used for each language. Feature selection is discussed further in the Analysis section.

4 Analysis

The two methods evaluated are clearly state of the art in authorship attribution of source code. In the worst case (the Burrows attribution of C++ documents), over 88% of the documents were successfully attributed. Given the large nature of the data set, these results are remarkable. However, opportunities to improve both methods clearly exist. This section describes some of those opportunities.

4.1 The Burrows Method

Perhaps the most obvious improvement to be made to the Burrows method is that of feature selection. The features were selected by creating six classes of features: operators, keywords, input/output tokens, function tokens, white space tokens, and literal tokens. Selected features were categorized into these classes. Sets of features were formed from all possible combinations of the classes, and empirical means were used to select the most significant feature classes. In the end, the feature classes selected were operators, keywords, and white space tokens.

An issue with this methodology is that the initial selection of features was somewhat arbitrary, as was the categorization into the six classes. Moreover, the fact that features were grouped and evaluated thusly meant that individual features were not evaluated – rather, somewhat arbitrary groupings of features were evaluated. In some of these classes, obvious omissions were made. For example, the white space tokens included carriage returns and new lines, but did not include line feeds. Furthermore, commonly-used symbols that are often emphasized in regards to programming style – such as semicolons and "curly braces" – were not even considered.

The Burrows method uses the "single best result" metric to assign authorship. That is, the author of the top-ranked document returned by the search query is attributed to be the author of said query document. This metric was selected based on an empirical comparison over two other metrics that were also considered. Additional metrics could certainly be considered. One possibility is to utilize an idea from the Frantzeskou method, and represent an author's entire corpus as a profile. If the work of each author were represented as a single document, it would certainly make sense for the author of the "single best result" to be attributed as the author.

In the Burrows experiments presented in this paper, the query documents were left in the corpus when the indexes were created by the search engine. As a result, the query document affects the Okapi similarity calculations. So, even though the query document was omitted from the results returned by the search engine, the query document still played a role in determining which results were returned in the first place. Therefore, the results of the comparative study are thusly skewed in favor of the Burrows method. For a better comparison, the query document itself should provide absolutely no knowledge in determining the authorship of said document.

4.2 The Frantzeskou Method

One potential improvement to be made to the Frantzeskou method is that of the similarity metric, the so-called SPI. The metric is used to determine which author profile in the data set is most similar to a query program, simply by determining how many n-grams the author profile and query program have in common. So, essentially, it is the author who frequently uses the sequences of characters that appear most frequently in the query program that is attributed to be the author. This similarity metric is quite simplistic, and a more sophisticated metric might be apropos.

In the Frantzeskou method, an author profile includes the L most frequently occurring n-grams used by that author, where L is a parameter. Choosing the size of L is a difficult task. Indeed, Frantzeskou leaves the determination of the optimal value for L to future work. Burrows suggests that the optimal value of L is effectively infinity, such that author profiles are not truncated at all, noting that this technique is equivalent to coordinate matching [16-17].

5 Conclusion and Future Work

The two methods evaluated are clearly state of the art in authorship attribution of source code. We've shown that the Frantzeskou method can successfully attribute over 94% of documents in a 20-class experiment. When the data has been anonymized by stripping out all comments and string literals, the success rate still approaches an impressive 90%. Likewise, the Burrows method, which inherently anonymizes data, successfully attributed nearly 90% of all documents.

Incremental improvements can likely be made to both methods. Selecting different feature sets, tweaking Okapi parameters, selecting an altogether different similarity measurement, utilizing a metric other than "single best result" to assign authorship, and representing an author's entire set of work as a profile rather than as individual documents would all be viable opportunities for improving the Burrows method. Potential improvements to the Frantzeskou method might include utilizing other similarity metrics and investigating the choice for the L parameter. When performing studies utilizing the Burrows method, it is also imperative that the query document not be included when generating the search engine indexes.

Future work also includes investigating ways of combining the two current state-of-the-art methods to create a new, more-effective method. Perhaps a "confidence level" could be applied to each document attribution. If the confidence is deemed low, alternative factors and/or methods could be utilized to assist in the final determination. Perhaps "identifying" data such as comments and string literals could be handled separately from the primary method of authorship attribution. If such information is unavailable, it won't affect the primary means of attribution. If it is available, it could be used in a secondary manner to supplement the primary method. This could make sense, considering the data contained in comments and string literals are mostly free from the confines of the programming language syntax and so, perhaps, should intrinsically be analyzed differently. Other factors such as identifiers and file names could potentially be analyzed in a similar way.

References

1. Spafford, E.H., Weeber, S.A.: Software Forensics: Can We Track Code to its Authors? *Computers & Security (COMPSEC)* 12(6), 585–595 (1993)
2. Zhao, Y., Zobel, J.: Effective and Scalable Authorship Attribution Using Function Words. In: Lee, G.G., Yamada, A., Meng, H., Myaeng, S.-H. (eds.) *AIRS 2005*. LNCS, vol. 3689, pp. 174–189. Springer, Heidelberg (2005)
3. Juola, P.: Authorship attribution. *Foundations and Trends in Information Retrieval* 1(3), 233–334 (2007)
4. Stamatatos, E.: A Survey of Modern Authorship Attribution Methods. *Journal of the American Society for Information Science and Technology* 60(3), 538–556 (2009)
5. Oman, P.W., Cook, C.R.: Programming Style Authorship Analysis. In: *Proceedings of the 17th Conference on ACM Annual Computer Science Conference (CSC)*, pp. 320–326 (1989)
6. Gray, A., Sallis, P., MacDonell, S.: IDENTIFIED (Integrated Dictionary-based Extraction of Non-language Dependent Token Information for Forensic Identification, Examination, and Discrimination): A Dictionary-based System for Extracting Source Code Metrics for Software Forensics. In: *Proceedings of the International Conference on Software Engineering (ICSE)*, pp. 252–259 (1998)
7. Krsul, I., Spafford, E.H.: Authorship Analysis: Identifying the Author of a Program. *Computers & Security (COMPSEC)* 16(3), 233–257 (1997)
8. MacDonell, S.G., Gray, A.R., MacLennan, G., Sallis, P.J.: Software Forensics for Discriminating between Program Authors using Case-based Reasoning, Feedforward Neural Networks and Multiple Discriminant Analysis. In: *Proceedings of the 6th International Conference on Neural Information Processing (ICONIP)*, pp. 66–71 (1999)
9. Ding, H., Samadzadeh, M.H.: Extraction of Java Program Fingerprints for Software Authorship Identification. *The Journal of Systems and Software* 72, 49–57 (2004)
10. Lange, R., Mancoridis, S.: Using Code Metric Histograms and Genetic Algorithms to Perform Author Identification for Software Forensics. In: *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pp. 2082–2089 (2007)
11. Elenbogen, B.S., Seliya, N.: Detecting Outsourced Student Programming Assignments. *Journal of Computing Sciences in Colleges* 23(3), 50–57 (2008)
12. Frantzeskou, G., Stamatatos, E., Gritzalis, S., Chaski, C.E., Howald, B.S.: Identifying Authorship by Byte-Level N-Grams: The Source Code Author Profile (SCAP) Method. *International Journal of Digital Evidence* 6(1), 1–18 (2007)
13. Burrows, S.D.: Source Code Authorship Attribution. Dissertation. RMIT University, Melbourne, Australia (2010)
14. Keselj, V., Peng, F., Cercone, N., Thomas, C.: N-gram Based Author Profiles for Authorship Attribution. In: *Proceedings of the Pacific Association for Computational Linguistics*, pp. 255–264 (2003)
15. Robertson, S.E., Walker, S.: Okapi/Keenbow at TREC-8. In: Voorhees, E., Harman, D. (eds.) *Proceedings of the Eighth Text Retrieval Conference*, pp. 151–162. National Institute of Standards and Technology, Gaithersburg (1999)
16. Witten, I.H., Moffat, A., Bell, T.C.: *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann, San Francisco (1999)
17. Uitdenbogerd, A.L., Zobel, J.: Music ranking techniques evaluated. In: Oudshoorn, M., Pose, R. (eds.) *Proceedings of the Twenty-Fifth Australasian Computer Science Conference*, pp. 275–283. Australian Computer Society, Melbourne (2002)

Towards Automated Malware Behavioral Analysis and Profiling for Digital Forensic Investigation Purposes

Ahmed F. Shosha, Joshua I. James, Alan Hannaway,
Chen-Ching Liu, and Pavel Gladyshev

UCD School of Computer Science and Informatics,
University College Dublin, Dublin, Ireland
{Ahmed.Shosha, Alan.Hannaway}@ucdconnect.ie,
{Joshua.James, Liu, Pavel.Gladyshev}@ucd.ie

Abstract. Digital forensic investigators commonly use dynamic malware analysis methods to analyze a suspect executable found during a post-mortem analysis of the victim's computer. Unfortunately, currently proposed dynamic malware analysis methods and sandbox solutions have a number of limitations that may lead the investigators to ambiguous conclusions. In this research, the limitations of the use of current dynamic malware analysis methods in digital forensic investigations are highlighted. In addition, a method to profile dynamic kernel memory to complement currently proposed dynamic profiling techniques is, then, proposed. The proposed method will allow investigators to automate the identification of malicious kernel objects during a post-mortem analysis of the victim's acquired memory. The method is implemented in a prototype malware analysis environment to automate the process of profiling malicious kernel objects and assist malware forensic investigation. Finally, a case study is given to demonstrate the efficacy of the proposed approach.

Keywords: Dynamic Malware Analysis, Kernel Object Profiling, Malware Investigation, Memory Forensics, Post-Mortem Analysis.

1 Introduction

Malware, or malicious software, has become a commonly used tool to commit crimes on the Internet, and poses significant threat to the security of computer systems and privacy of computer users. To defend against malware, a large body of computer security research has resulted in various techniques to analyze, detect and eliminate malware [1-8]. Although proposed approaches assist malware analysts in accomplishing their mission, advanced malware countermeasure techniques have been developed to generate variants of the malicious code in an attempt to elude detection from traditional methods. Further, the substantial increase in discovered malware samples every day negatively impacts the effectiveness of traditional static analysis approaches. As such, highly automated dynamic techniques have been called for [9]. A variety of automated dynamic malware analysis approaches have been proposed to cope with the large number of discovered malware samples. These dynamic methods were implemented in various

sandbox solutions to provide the required process automation, and assist malware analysts in acquiring required knowledge about the malicious code's behavior [10]. A sandbox, in this work, refers to a managed virtual environment with a pre-determined software configuration used to implement proposed methods and to observe a behavior of a malicious binary through its execution process [10].

From a digital forensic investigation perspective, when investigators are confronted with an investigation involving a suspect executable, different incident response procedures are followed to analyze and investigate the suspect binary. Dynamic malware analysis methods proposed in computer security research are commonly used to allow an investigator to understand the behavior of a suspect executable. Analysis of extracted traces, correlating evidence and artifacts to the suspect executables' behavior, however, is manually conducted by the investigator, and solely relies on his or her expertise. This manual process is time consuming, error prone and allows for inconsistent interpretation of malicious evidence which threatens the integrity of the investigation [11]. Moreover, the use of dynamic malware analysis methods for forensic investigation purposes has a number of limitations. Currently proposed methods are designed to assess the behavior of malicious code for signature development purposes, and have not been designed specifically considering the concepts and principles of digital forensic investigations. Thus, employing these methods in malware forensic investigations may result in inaccurate conclusion.

This work highlights the limitations of the use of currently proposed dynamic malware analysis methods applied to digital forensic investigations, and proposes a set of improvements to utilize these methods for forensic investigation purposes. To this end, a method for dynamically profiling the kernel memory of malware objects is proposed. The proposed method allows for automated identification and extraction of malicious kernel objects from a victim's acquired forensic memory image during a post-mortem forensic analysis. In addition, it can be extended to profile different behavioral aspects of malware execution, and allow an investigator to automate the process of malware traces detection in a post-mortem forensic analysis of the victim's computer system. To demonstrate the applicability of the proposed method, a prototype forensic-specific dynamic analysis sandbox solution has been developed and implements the proposed profiling technique. Developed sandbox is evaluated through a case study involving profiling a commonly used malware tool-kit that emerged over the last few years to commit financial crimes on the Internet. Developed profiles are, then, used to automate the analysis of dynamic kernel memory during post-mortem forensic analysis and automatically identify malware related kernel objects.

To summarize, the contribution of this paper is as follows:

- This work highlights the limitations of the use of currently proposed dynamic analysis methods in malware forensic investigations, and outlines required improvements to utilize the capabilities of these methods for digital forensic investigation purposes.
- This work proposes a dynamic profiling method applied to dynamic kernel memory to automate the process of identification and extraction of malicious

kernel objects in acquired forensic memory images during a post-mortem forensic analysis.

- This work present a prototype dynamic malware analysis sandbox for digital forensic investigation purpose based on the proposed dynamic kernel memory profiling method.

Paper Organization. In section 2, limitations of currently proposed dynamic malware analysis methods in forensic investigations are described. In section 3, profiling of dynamic kernel memory for digital forensic investigation purposes is presented and described in details. Section 4 describes the prototype implementation of the proposed approach, and gives a case study. Section 5 gives a discussion about the proposed method and outlines future research work. Finally, section 6 concludes the paper.

2 Limitations of Dynamic Analysis Methods from Digital Forensic Investigation Perspective

Dynamic analysis of malware is an automated approach to identify a behavior of malicious program through observation of the program's execution in a managed environment [12]. Typically, malicious programs are automatically loaded into a managed virtual machine environment and executed. Interactions between the malicious program and an operating system are observed to provide human analysts an overview about the sample's behavior and whether further analysis is required or not. Observed interactions of the malicious program in monitored operating systems include which system calls are invoked, and arguments used to interact with the operating system kernel. Finally, a detailed report about the program's activities, i.e. file activities, Windows Registry activities, and networking activities, are provided to the analyst. Such information allows a human analyst to identify if a program subject to analysis is a new malware sample, a variant sample or a benign program. Based on the analyst's decision, proper detection signature is developed. In contrast, a number of anti-analysis techniques have been developed by malware authors to disrupt malware analysis process, and impede further investigations [13, 14].

Although currently proposed dynamic analysis methods substantially automate and improve the process of malware analysis and malicious code signature development in computer security research, the use of these methods is limited in digital forensic investigations. Thus, a part of malware analysis for digital forensic investigations is accomplished manually despite the fact that it can be automated, if digital forensic investigation objectives were initially considered and integrated into the design of these methods. More important, relying on results of currently proposed methods may contribute in resulting inaccurate forensic investigation conclusions.

This section highlights a number of limitations that hinder utilization of currently proposed dynamic malware analysis methods in digital forensic investigations, and proposes a set of improvements that, if considered, assist in automating malware investigation and preserve the integrity forensic analysis.

2.1 Multiple Malicious Execution Paths

Malware developers employ different methods to impede dynamic analysis of malware and malicious code investigation [14]. A prevalent feature in malware is the frequent collection of intelligence about the surrounding environment and attempting to detect whether it is an analysis or debugging environment. If an analysis environment is detected, malware may suppress its execution and terminate malicious payload installation, or may execute a different execution path that results in benign traces in an attempt to evade the human analyst. This behavior is termed “*malware’s evasion personalities*” [15]. To defend against evasion personalities, various approaches have been proposed to disguise the analysis environment, so that, it becomes transparent to a malware. Although proposed disguising methods to defend against evasion personalities substantially contribute to the intended analysis goal, a possibility of existence of multiple execution paths is still valid. Malware may have different malicious payloads or have different behaviors based on certain properties of the compromised environment: the existence of a predetermined Internet browser version, or installation of specific software or hardware, for example. Since currently implemented dynamic analysis methods do not consider tracking multiple execution paths [16] and developed sandbox solutions cannot consider all possible environment configurations, analysis may result in an execution path that has never been executed on the victim system subject of forensics investigation. This incomplete analysis could lead digital forensic investigators to reach an invalid conclusion based on incomplete knowledge of the behavior of the malware.

To overcome multiple execution paths in computer security research, paths tracking techniques have been proposed to execute all possible paths in malicious programs [16]. Unfortunately, proposed techniques are computationally expensive when applied to thousands of malware samples collected every day.

Investigation of all possible malware execution paths can be associated with observations of the state of the system to determine possible explanations that could have resulted in observed system state. Formal theories have been proposed to provide required explanations in the context of multiple execution paths, and to reconstruct events related to a certain execution path [17, 18]. Although these theories have applications in different forensic investigation domains, an application to malware evasion personality detection is still missing. Thus, inclusion of these approaches to malware analysis provides more information to assist investigators in deriving reasonable conclusions.

2.2 Interrelation between Observed Objects

Dynamic malware analysis methods monitor the interactions between a malware sample and an operating system kernel [19, 20], e.g. invoked system calls and its arguments. Other methods, such as those proposed in [21, 22] not only observe objects interactions, but also, profile the interaction patterns such as evolving pattern of a malicious object’s data structure in dynamic kernel memory. Profiled patterns are further used to derive a malware detection signature. In digital forensic investigations

of malware, interrelations between observed objects are essential to deduction of further actions invoked by a malicious object [23]. If relations between observed objects are not properly defined, it may not be possible to infer an instance of an action. That is, investigators are required to manually define the relations between observed objects. Currently proposed dynamic malware analysis methods do not observe and define the mutual relation between malicious objects, although, there are various extensions that can provide necessary information about malicious objects interrelationships [24].

Different methods allow tracking information flow between objects, denoted as dynamic taint analysis, which is considered a complementary approach to dynamic analysis approaches [25]. In dynamic taint tracking, information is labeled and tracked throughout program execution for different purposes. More precisely, propagation of labeled information in dynamic tainting systems is tracked in the context of a malicious objects' execution. Currently proposed dynamic tainting and tracking systems focus on tracking data between objects; however, data propagation paths can be used to derive the interrelations between observed objects. Such derived information allows automation of the process of object interrelation construction, and allows investigators to infer further actions based on defined objects relationships. That is, extending dynamic taint tracking to consider dynamic identification of interrelationships between observed malicious objects and integration of such methods in dynamic analysis approaches is essential for digital forensic investigation, and assists investigators in automating the forensic analysis processes based on object relationships.

2.3 Profiling Dynamic Kernel Objects

Memory forensics is an important portion of digital forensic investigation process when malware is concerned. Various signature-based approaches have been proposed to extract kernel data structures from dynamic kernel memory [26]. These methods scan the dynamic kernel memory to detect and extract different kernel data structure types such as processes, threads, network or VAD objects in Windows operating system kernel [27, 28]. Forensic analysis of extracted objects, and determining if an object belongs to a malware, is a manual process that relies on the investigator's expertise. Forensic analysis of kernel data structure objects requires, as well, deep knowledge of the operating system internals and techniques employed by malware to disrupt investigation through the manipulation of the kernel object characteristics. Moreover, specification of the kernel data structures are likely to change with new builds of the operating system kernel. Thus, manual investigation of the kernel data structure in acquired memory is a significant challenge for forensic investigators.

In computer security research, different approaches have been proposed for automated profiling of kernel objects characteristics in dynamic kernel memory, to assist malware signature development process [22]. Proposed methods are designed to identify the evolving patterns of kernel data structures in memory and profile such pattern. However, these methods are insufficient for forensic analysis of kernel data

structure, as they do not allow for automated identification of malicious objects in post-mortem forensic investigations based on developed profiles.

Profiling for digital forensic investigation purposes has been proposed in different investigative domains [29]. However, profiling malware behavior for digital forensic investigation is still missing. A profiling method to automate forensic identification and extraction of malicious objects will, significantly, assist the process of malware forensic investigation and memory forensic analysis.

3 Profiling Dynamic Kernel Memory

In this section, a method for profiling dynamic kernel memory for digital forensic investigation purposes is presented. The proposed method allows for automated identification of malicious kernel objects in post-mortem forensic analysis of acquired memory.

Dynamic kernel memory is a memory portion where dynamically allocated kernel data structure objects are present. Dynamic kernel memory recently became a target of an increasing amount of kernel level malware such as rootkit attacks [30]. These attacks employ advanced stealth techniques to control and manipulate an operating system kernel. For example, Direct Kernel Object Manipulation (DKOM) attack allows rootkits to hide malicious kernel objects in the operating system kernel through manipulation of malicious kernel object's characteristics [30]. Other attacks such as hijacking kernel execution – denoted as Kernel Object Hooking attack (KOH) [31]– allow kernel level malware to execute compromised code after hijacking the kernel code control flow.

Digital forensic investigators of dynamic memory are required to investigate various kernel level data structure types to identify a presence of rootkits, and existence of malicious kernel objects. As previously discussed, this investigative process has limitations that may compromise the investigation's integrity. Thus, an approach for automated forensic investigation of dynamic kernel objects is required.

In the proposed method, a procedure to monitor kernel object's characteristics is proposed and utilized to develop a profile for malicious kernel objects. Developed profiles will allow investigators to automatically determine kernel objects related to malware in an acquired forensic memory image during post-mortem forensic analysis based on Object-To-Profiles matching procedure.

3.1 Profiling Malicious Kernel Objects for Forensic Investigation Purposes

Program execution process in the operating system requires allocation of memory regions to the program to execute its instructions, and creation of kernel objects in dynamic kernel memory to manage the program execution. These kernel objects control every aspect of the program's execution in the operating system kernel. For example, `EPROCESS` in Windows operating systems [32] or `task_struct` in UNIX based systems represent and manage running program's processes and threads in the operating system kernel.

Since kernel object data structures are formally defined by the operating system code, and instances of these objects are allocated in dynamic kernel memory, investigators attempt to differentiate between benign kernel objects and malicious kernel objects. This process is essential to determine which memory regions are allocated to malware, and which regions are suspicious but non-conclusive and require further analysis.

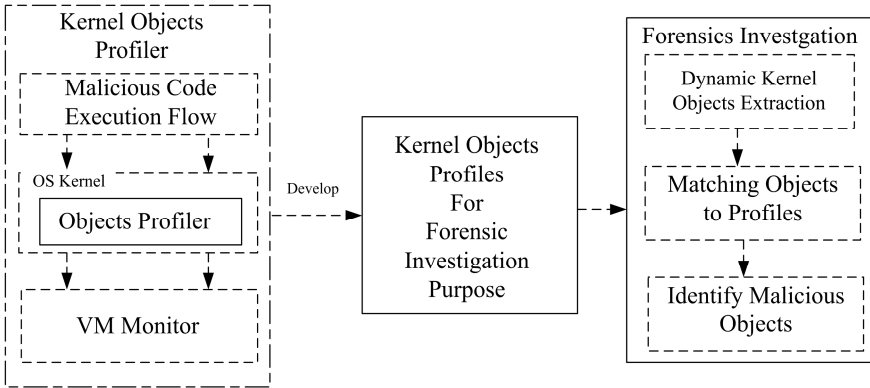


Fig. 1. Kernel Object Profiler Process Model

To automate the process of malicious kernel object identification in post-mortem memory analysis, the proposed method profiles the characteristics of a malicious kernel objects in dynamic kernel memory. When a program is being executed in, there exist a unique set of characteristics of kernel object's properties that identifies the program. Determining such characteristics and monitoring its values, while program code is being executed, allows for development of an object profile that can be used to assist kernel objects investigation. To determine characteristics of malicious kernel objects, memory monitoring and introspection technique are employed to observe memory regions allocated to the dynamic kernel objects.

The proposed profiling process model is shown in Figure 1. A malicious code executed instructions are monitored through a dynamic analysis method in a managed virtual environment. Through malicious code execution, snapshots of memory allocated to the kernel objects that represent malware execution is acquired at each executed instruction, and are added to the object profile for further use in a digital forensic investigation. Acquired memory snapshots are automatically matched to the kernel object's definition to identify kernel object property values and determine values that have changed as a result of instructions execution. Finally, at post-mortem memory investigation, developed profiles are used to automate malicious kernel object identification. This is accomplished by extracting dynamic kernel objects from acquired memory and automatically match extracted objects with developed profiles.

3.2 Kernel Object Memory Profiling Formalization

This section presents a formalization used in profiling memory allocated to malicious kernel objects, and determining object properties at different execution states. A malicious kernel object O_m represents a malicious code execution in an operating system kernel. Memory region μ is a dynamic kernel memory space allocated to O_m . A kernel object O_m holds a set of properties ρ_n that used by the operating system kernel to manage the program execution, such that:

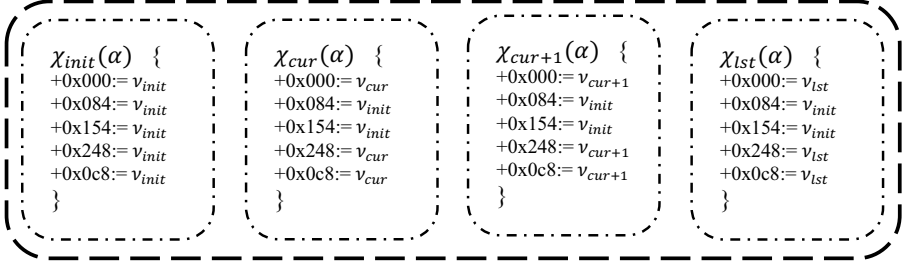


Fig. 2. An Example of Kernel Object Profile

$$O_x = \{ \rho_1, \dots, \rho_k \mid \rho_j \text{ is allocated at memory offset } \mu_j \}$$

A *Forensic Kernel Object Profile* (OP) is a set of elements that represent memory snapshots for memory allocated to the kernel object O_x through execution of a program represented by O_x . An element in the set is called *kernel object's memory snapshot* (α) at an executed instruction χ , and is defined as a set of 2-tuples (μ_ρ, v_ρ) , where μ_ρ represents a memory offset for a kernel object property $\rho_x \in O$ and v_ρ represents a value assigned to property ρ_x as a result of an execution state χ , such that:

$$\chi_{cur}(\alpha) = \{ (\mu_1 := v_1) \wedge (\mu_2 := v_2) \dots \wedge \dots (\mu_n := v_n) \}$$

Figure 2 presents an example of forensic kernel object profiling of the EPROCESS dynamic kernel object in Windows operating system kernel. The rounded boxes in Figure 2 show various *kernel object's memory snapshot* (α) at different executed instructions. For example, at the initialization state, the operating system initializes properties of a kernel object through assigning a process name, unique process id, initializing the process kernel information, determining control flags and assigning proper access security token at offsets, +0x154, +0x084, +0x000, +0x248 and +0x0c8, respectively [32]. Through program execution, properties of the kernel object are changed to allow the program to execute intended code, i.e. new security flags are assigned and existing control flags are updated, etc.

Thus, according to presented profiling method, different kernel object snapshots (α) for dynamic memory allocated to the object are acquired. For example, at instruction execution state χ_{cur+1} , characteristics of profiled kernel object are defined as:

$$\chi_{cur+1}(\alpha) = \{(0x000 := v_{cur+1}) \wedge (0x084 := v_{init}) \wedge \dots (0x0c8 := v_{cur+1}) \dots\}$$

and the Forensic Kernel Object Profile (OP), is defined as:

$$OP(EPROCESS_{malware}) = \{\chi_{init}(\alpha), \chi_{cur}(\alpha), \chi_{cur+1}(\alpha), \dots, \chi_{lst}(\alpha)\}$$

Fundamentally, profiled memory snapshots encode changes in the object properties at different execution states, and determine characteristics of profiled object at every execution state. The observed changes in monitored object properties, results in a unique property updates pattern that allows for development of a malicious kernel object profile and assists in differentiating malicious kernel objects from benign kernel objects.

To accurately profile properties of a specific kernel object, some properties in kernel object definition may include host or user specific data, e.g. timestamp of the object creation or user directory of downloaded malicious programs, etc. Such information is specific to the analysis environment configuration and may contribute to inaccurate profiles. Thus, kernel object properties of interest – and that are considered in profiling process – are properties that affect program execution in the operating system kernel and, if tampered with, monitored program may produce unpredictable behavior [33, 38]. Thus, user or host specific information is defined as a set of properties ρ_{ex} and are excluded from profiling process. Hence, a final profile denoted as $OP(Obj_x)$ is formalized as follow:

$$OP(Obj_x) = \bigcup \chi_m(\alpha) - \bigcap_{\rho_{ex} \in P} \rho_{ex}$$

This formula represents the process of profiling a memory snapshot of a kernel object of interest at different instruction execution states. To generalize developed profiles and exclude properties that may produce false negative results, user specific information such as, process id, user timestamps or an executable location, are eliminated from profiling process and kernel specific properties are only considered in the profiling procedure.

3.3 From Malicious Code Execution to Object Profiles

As previously illustrated, memory regions allocated to a malicious kernel objects are profiled to determine the object characteristics in different execution states. This section presents a formalization of code execution states that stimulate presented profiling process. A malicious executable P is modeled as a binary program that holds a set of assembly language instructions $I = \{I_1, I_2 \dots I_n\}$. The execution of the

malicious program P possesses a sequential execution of instruction set I in P with an exception to instructions that change program control-flow, e.g. jump instructions.

Consequently, malicious program execution can be presented as a *control flow graph* (CFG) [34, 39]. CFG (P) can be defined as 2-tuple (S, E) , where $S \in \mathcal{E}$ is an execution state presented as assembly instructions, and E is a set of edges $E \subseteq S \times S$, where E represents a transition corresponding to execution of a malicious instruction in memory.

The kernel object memory profiling procedure based on presented formalization is defined as 3-tuple, $P = \llbracket I, E, PR \langle I_g \rangle \rrbracket$, where:

- I is a set of execution states, each representing an instruction in determined malicious executable.
- E is a set of edges corresponds to transition to an instruction.
- $PR \langle I_g \rangle$ is the profiling procedure that acquire a snapshot of memory allocated to malicious kernel object at execution of an instruction I_g .

#	INS	Argument
1	push	dwDesiredAccess
2	call	ds:openMutex
3	cmp	[ebp+var_4] , eax
4	Jz	Short loc_30902E
5	push	offset Name
6	push	0
7	push	0
8	call	ds:createMutex
loc_30902E		
9	push	0
10	call	ds:exit
loc_30903A		
11	mov	esp , ebp
12	pop	esp
13	retn	

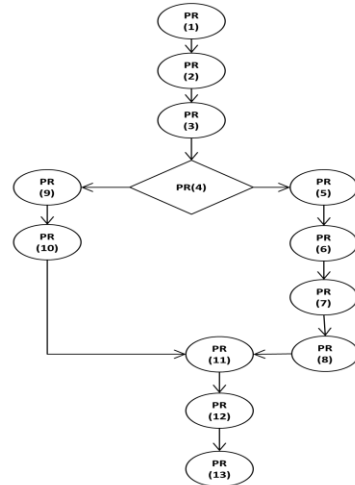


Fig. 3. Executed Code CFG to Memory Profiles Snapshots CFG

Figure 3 presents a practical example of modeling a malicious executable, and creating a profile of memory allocated to kernel objects in dynamic kernel memory. Code in Figure 3 is modeled as a CFG graph. Each instruction is modeled as an execution state, and execution of next instruction is represented as a transition in the CFG graph. Instructions similar to those presented in line 4 represents a branching transition of the CFG graph to instructions located at `loc_30902E` in memory. Through execution of the malware CFG modeled graph, malicious kernel objects are profiled using presented method, and added to the malicious kernel object profiling space.

Intuitively, the object profile space can, as well, be modeled as CFG of profiles analogy to code CFG. Profile CFG illustrated in Figure 3 represents an EPROCESS

object of a running malicious process in Windows operating system kernel. Consecutive memory snapshots are acquired for malicious object through malicious code execution in a managed environment. Acquired memory profiles are, then, used to determine malicious EPROCESS kernel objects in a digital forensic investigation of memory images that infected with profiled malware sample.

4 Implementation and Case Study

Implementing a prototype dynamic kernel object profiler for digital forensic investigation purpose requires dynamic access to memory regions allocated to malicious kernel objects and monitoring executed instructions by malicious code. That is, QEMU [35], an open source processor emulator was used to accomplish aforementioned requirements. QEMU was customized to allow instruction emulation to stimulate proposed kernel object profiling procedure. Note that, in this research, Windows operating system kernel is approached for presented profiling process, specifically dynamic kernel objects that represent running process in Windows, such as, EPROCESS and its substructures: _KPROCESS and _KTHREAD. This is because EPROCESS kernel object is a common target for forensic investigators of malware and references different types of kernel objects that are essential to the investigation. For example, EPROCESS keeps track of memory allocated to a program through the Virtual Address Descriptor data structure, and files mapped in memory [36]. Determining memory regions allocated to a program in QEMU is accomplished through monitoring the value loaded into CR3 processor registers. This value represents the page-directory base register (PDBR) of physical memory address of current program's process loaded into QEMU processor [32]. Monitoring the aforementioned register enables determining the physical memory address of currently loaded EPROCESS into the emulation processor. Once memory region for an EPROCESS is determined, the memory region is mapped to the formal definition of EPROCESS as described in Windows operating system kernel specification to identify the offset of each property in monitored object and its value. Finally, the proposed profiling procedure snapshots identified memory offsets, as previously described, at invocation of emulated instructions in QEMU processor. Acquired malicious EPROCESS profiles are, then, used to automate identifying if a kernel objects is malicious or not in post-mortem analysis of a memory.

To automate the process of malicious EPROCESS extraction and identification, a plugin to Volatility Memory Forensic Framework [26] developed to automatically extract kernel objects and match extracted objects with developed malware profiles. If an extracted object matches a profile, memory regions allocated to the suspect program and referenced by the suspect EPROCESS are automatically extracted for further forensic analysis.

4.1 Zeus Toolkit Profiling Case Study

To evaluate the efficacy of the proposed method, a forensic profile for Zeus malware [37] was developed. Zeus is a toolkit that is commonly used to commit financial crimes on the

Internet [37]. In the last few years, Zeus toolkit has become a dominant tool for cyber criminals since it allows to, easily, configure a malicious binaries to commit a variety of cybercrimes, such as stealing the users' Internet banking accounts and credit card information and leaking user-sensitive financial information to a black market.

To verify developed Zeus's profiles, four Windows 7 virtual machines infected with Zeus malware were deployed. Dynamic kernel memory of each infected VM acquired for analysis, and matched with developed Zeus profiles. Kernel objects in each forensic memory image have been processed using Volatility with developed extraction and identification plugin.

Table 1. Results of Profiling the CFG Graphs Corresponds to Zeus's Executable

Zeus Variants	<i>Acquired memory Snapshots</i>	<i># Benign Kernel Objects</i>	<i>False Detections</i>
<i>ntos.exe</i>	4511	64	-
<i>oembios.exe</i>	4009	52	-
<i>Sdra64.exe</i>	3794	52	-
<i>PP08.exe</i>	3401	43	-

This allows automatic identification of malicious kernel objects related to Zeus, and also automatically extracted memory regions referenced by Zeus's EPROCESS kernel object.

Table 1 shows the results of Zeus's profiling process and characteristics of each acquired forensic memory image for investigation. As shown in Table 1, Zeus's Kernel Object Profile (OP) is consists of up-to 4500 object memory snapshots.

In essence, acquired memory snapshots of Zeus's kernel object correspond to emulated instructions of Zeus's executable and executed states in Zeus's modeled CFG graph, as previously described. In addition, each acquired memory image has up-to 60 EPROCESS kernel objects for commonly-used benign software e.g. Microsoft Internet Explorer, MS Media Player, and MS Office.

Matching extracted kernel objects with acquired profiles resulted in identification of Zeus's EPROCESS objects in all memory images without producing false positives with benign EPROCESS objects. Furthermore, to verify the preciseness of acquired profiles, Zeus's profiles have been used to investigate freely available [26] seven different Windows XP SP2 forensic memory images infected with different malware samples. Developed profiles, however, did not produce false results with other malicious kernel objects.

5 Discussion and Future Work

Although presented profiling method shows promising results in determining characteristics of malicious kernel objects and automating malicious kernel object identification in post-mortem memory analysis, some improvements are required.

The proposed method is considered a complementary approach for dynamic analysis techniques; thus, challenges to dynamic analysis approaches may, also, affect the proposed method. For example, to develop a complete object profile, all execution paths in malicious code's CFG graph have to be considered. Otherwise, if a malicious code has multiple execution paths, proposed method may result in incomplete profiles and may produce false results. Thus, the proposed method has to be assisted with proposed improvements to dynamic analysis approaches for digital forensic investigation.

Hence, our future work plan includes approaching proposed dynamic analysis improvements and implementing improved approaches in a forensic-specific malware investigation platform.

6 Conclusion

This research highlighted the limitations of employing dynamic malware analysis approaches in digital forensic investigations of malware, and proposed a set of improvements to presented limitations. Based on highlighted limitations, a method proposed to profile malicious kernel objects in dynamic kernel memory. Developed malware profiles allow investigators to automatically identify malicious kernel objects during post-mortem memory analysis of acquired dynamic kernel memory of the victim's computer. To allow an automated profiling of malicious kernel objects, a prototype malware sandbox solution developed and used to profile a malware family that is commonly used to commit financial crime on the Internet.

References

1. Yin, H., et al.: Panorama: Capturing System-wide Information Flow for Malware Detection and Analysis. In: Proceedings of the 14th ACM Conference on Computer and Communications Security (2007)
2. Yin, H., Liang, Z., Song, D.: HookFinder: Identifying and Understanding Malware Hooking Behaviors. In: Proceedings of Distributed System Security Symposium (2008)
3. Kolbitsch, C., et al.: Effective and Efficient Malware Detection at the End Host. In: Proceedings of the 18th Conference on USENIX Security Symposium (2009)
4. Vasudevan, A., Yerraballi, R.: Cobra: Fine-grained Malware Analysis using Stealth Localized-Executions. In: Proceedings of IEEE Symposium on Security and Privacy (2006)
5. Dinaburg, A., et al.: Ether: Malware Analysis Via Hardware Virtualization Extensions. In: Proceedings of the 15th ACM Conference on Computer and Communications Security (2008)
6. Lanzi, A., Sharif, M., Lee, W.: K-Tracer: A System for Extracting Kernel Malware Behavior. In: Proceedings of the 16th Annual Network and Distributed System Security Symposium (2009)
7. Bayer, U., et al.: Dynamic Analysis of Malicious Code. *Journal in Computer Virology* 2(1), 67–77 (2006)

8. Christodorescu, M., Jha, S.: Static Analysis of Executables to Detect Malicious Patterns. In: Proceedings of the 12th USENIX Security Symposium (2003)
9. Moser, A., Kruegel, C., Kirda, E.: Limits of Static Analysis for Malware Detection. In: Proceedings of Computer Security Applications Conference (2007)
10. Egele, M., et al.: A Survey on Automated Dynamic Malware Analysis Techniques and Tools. *ACM Comput. Surv.* 44(2), 1–42 (2012)
11. Farmer, D., Venema, W.: *Forensic Discovery*. Addison-Wesley (2005)
12. Nance, K., Bishop, M., Hay, B.: Virtual Machine Introspection: Observation or Interference? In: *IEEE Security and Privacy* (2008)
13. Sharif, M., et al.: Impeding Malware Analysis Using Conditional Code Obfuscation. In: Proceedings of the Network and Distributed System Security Symposium (2008)
14. You, I., Yim, K.: Malware Obfuscation Techniques: A Brief Survey. In: Proceedings of the Int. Conf. on Broadband, Wireless Company (2010)
15. Balzarotti, D., et al.: Efficient Detection of Split Personalities in Malware. In: Symposium on Network and Distributed System Security (NDSS) (2010)
16. Moser, A., Kruegel, C., Kirda, E.: Exploring Multiple Execution Paths for Malware Analysis. In: *IEEE Symposium on Security and Privacy* (2007)
17. Shosha, A.F., James, J.I., Gladyshev, P.: A Novel Methodology for Malware Intrusion Attack Path Reconstruction. In: Gladyshev, P., Rogers, M.K. (eds.) *ICDF2C 2011*. LNICST, vol. 88, pp. 131–140. Springer, Heidelberg (2012)
18. Gladyshev, P., Patel, A.: Finite State Machine Approach to Digital Event Reconstruction. In: *Digital Investigation* (2004)
19. Forrest, S., Hofmeyr, S., Somayaji, A.: The Evolution of System-Call Monitoring. In: Proceedings of the Annual Computer Security Applications Conference (2008)
20. Mutz, D., et al.: Anomalous System Call Detection. *ACM Trans. Information System Security* (2006)
21. Riley, R., Jiang, X., Xu, D.: Multi-Aspect Profiling of Kernel Rootkit Behavior. In: Proceedings of the 4th ACM European Conference on Computer Systems (2009)
22. Rhee, J., Lin, Z., Xu, D.: Characterizing Kernel Malware Behavior With Kernel Data Access Patterns. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security (2011)
23. Malin, C., Casey, E., Aquilina, J.: *Malware Forensics: Investigating and Analyzing Malicious Code*. Syngress (2008)
24. Newsome, J., Song, D.: Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploits on Commodity Software. In: Proceedings of Network and Distributed System Security Symposium (NDSS) (2005)
25. Schwartz, E., Avgerinos, T., Brumley, D.: All You Ever Wanted to Know About Dynamic Taint Analysis and Forward Symbolic Execution. In: *IEEE Symposium on Security and Privacy (Oakland 2010)* (2010)
26. Volatility.: An Advanced Memory Forensics Framework (2012), <https://www.volatilesystems.com/default/volatility>
27. Dolan-Gavitt, B.: The VAD Tree: A Process-Eye View of Physical Memory. In: *Digital Investigation* (2007)
28. Schuster, A.: Searching for Processes and Threads in Microsoft Windows Memory Dumps. In: Proceedings of the 6th Annual Digital Forensic Research Workshop (2006)
29. Marrington, A., et al.: A Model for Computer Profiling. In: *The Third International Workshop on Digital Forensics* (2010)
30. Hoglund, G.: *Rootkits: Subverting the Windows Kernel*. Addison-Wesley (2005)

31. Wang, Z., et al.: Countering Kernel Rootkits With Lightweight Hook Protection. In: Proceedings of the 16th ACM Conference on Computer and Communications Security (2009)
32. Russinovich, M.: Windows Internals. Microsoft Press (2009)
33. Dolan-Gavitt, B., et al.: Robust Signatures for Kernel Data Structures. In: Proceedings of the 16th ACM Conference on Computer and Communications Security (2009)
34. Clarke, E., Grumberg, O., Peled, D.: Model Checking. MIT Press (2000)
35. Bellard, F.: QEMU, A Fast and Portable Dynamic Translator. In: Proceedings of the Annual Conference on USENIX Annual Technical Conference (2005)
36. Van Baar, R.B., Alink, W., Van Ballegooij, A.R.: Forensic Memory Analysis: Files Mapped in Memory. Digital Investigation (2008)
37. Binsalleeh, H., et al.: On the Analysis of the Zeus Botnet Crimeware Toolkit. In: Proceedings of the Eighth Annual International Conference on Privacy Security and Trust (2010)
38. Shosha, F.A., James, J., Chen-Ching, L., Gladyshev, P.: Evasion-Resistant Malware Signature Based on Profiling Kernel Data Structure Objects. In: Proceedings of the 7th Intl. Conference on Risks and Security of Internet Systems (CRiSIS) (2012)
39. Shosha, A.F., James, J.I., Liu, C.-C., Gladyshev, P.: Towards Automated Forensic Event Reconstruction of Malicious Code (Poster abstract). In: Balzarotti, D., Stolfo, S.J., Cova, M. (eds.) RAID 2012. LNCS, vol. 7462, pp. 388–389. Springer, Heidelberg (2012)

Measuring the Preference of Image Content for Self-reported Consumers of Child Pornography

Kathryn C. Seigfried-Spellar

The University of Alabama, 428 Farrah Hall, Tuscaloosa AL, 35487, USA
kseigspell@bama.ua.edu

Abstract. Research has begun to critically analyze the types of images collected by child pornography consumers. However, the collections of child pornography consumers may not necessarily be representative of their preferences. In addition, a literature review of the available scales or measurements, which assessed pornography preference, yielded scarce results regarding images of child sexual victimization. First, this paper will review some of the empirical literature on the various types of images collected by child pornography consumers. Next, this author will discuss the development of the Child Pornography Image Preference Scale (CPIPS), a self-report measure of child pornography image preference. Finally, the results of a preliminary test of the CPIPS will be discussed along with the study's limitations. Overall, by introducing this scale to the academic and law enforcement community, further validation through empirical testing may be achieved for the Child Pornography Image Preference Scale (CPIPS).

Keywords: child pornography, image content, measurement, self-report.

1 Content of Child Sex Abuse Images

Research has begun to critically analyze the types of images collected by child pornography consumers. In 2003, research was conducted by the National Center for Missing & Exploited Children, which became known as the National Juvenile Online Victimization (N-JOV) study. The N-JOV study nationally sampled law enforcement agencies regarding the number of arrested cases involving Internet crimes against children between July 1, 2000 and June 30, 2001 [1]. Of the estimated 2,577 arrests, 36% were for possession, distribution, or trading of Internet child pornography. According to the report, the majority of the child pornography images collected by the offenders depicted young, prepubescent children between the ages of 6 and 12 years with some of the offenders possessing images of children younger than 3 years of age [1]. In addition, the content of the child pornography images were graphic in nature, such as the sexual penetration of a child and sadistic violence including rape and torture. Overall, the authors concluded the majority of child pornography images seized by law enforcement depict the explicit sexual and/or violent abuse of prepubescent children [1].

In a follow up analysis, Wells, Finkelhor, Wolak, and Mitchell examined the law enforcement cases identified from the N-JOV study in which no arrest was made by any U.S. law enforcement agency [2]. Wells et al. yielded a final sample of 68 non-arrest cases, and 34 of those cases involved Internet child pornography. Of the 34 child pornography cases, 70% involved images depicting graphic sexual abuse with nearly half of the images showing penetration. In addition, 44% of the images were of an adult engaging in a sexual act with a child [2].

In a study conducted by Webb, the pornography collections of 90 men convicted of Internet child pornography offences were analyzed using the COPINE continuum. The number of images was 16,698 but the collections ranged in size from 2 to 921,000 child pornography images. For 72 of the men, information was available regarding the age of the victims in their collections, and 86% of these cases included images of children under the age of 10 years [3]. 31% of the men collected images at the highest level of child sexual victimization (sadistic/bestiality) with the majority collecting images categorized as explicit erotic posing (71%), explicit sexual activity (71%), assault (80%), and gross assault (76%)[3]. Overall, the study suggested the majority of the offenders are collecting images of young children with higher levels of child sexual victimization.

More recently the Child Exploitation and Online Protection Centre (CEOP), which is a part of the United Kingdoms' law enforcement, focuses on child sex abuse crimes. In their 2007-2008 strategic report, the centre reported an increase in the number of noncommercial sources depicting babies and toddlers in child sex abuse images [4]. In addition, law enforcement has witnessed an increase in the number of images depicting children of different racial backgrounds and locations, such as South America, and South Korea. This trend in the increased number of non-White child victims was noted in the 2009 report as well [5]. Overall, the reports suggested this trend in atypical racial diversity might be related to the increase in traveling child sex offenders, who commute abroad and record their abuse [4,5].

The CEOP also noted an increase in the severity of images being posted by commercial sources. Traditionally, less sexually graphic images are posted to entice customers to provide payment for the more sexually explicit and violent images of child abuse [4]. However, according to the CEOP's Behavioral Analysis Unit, commercial sources are responding to an increased desire for images depicting sadistic sexual violence and younger victims [4]. In other words, the content of the initial images being posted to entice the consumers are depicting increased levels of graphic sexual and physical victimization of younger children.

In addition, a report by the Internet Watch Foundation supported previous research in that the images depict severe forms of sexual abuse with over half (69%) of the child victims appearing to be under the age of 10 years [6]. Of those images, 24% appear to be under the age of 6 years with 4% of the child victims appearing to be to be under the age of 2 years [6]. Lastly, over half of the images were also classified as level 4 (penetrative sexual abuse) or 5 (sadism or

bestiality), which are the two most severe levels of child sexual victimization according to the UK's Sentencing Guidelines Council [6].

Finally, the Canadian Centre for Child Protection analyzed the tips received between September 26, 2002 and March 31, 2009 from cybertip.ca, Canada's tipline for reporting Internet crime against children. Of the 35,111 websites reported to the tipline, 15,662 hosted images of child sexual abuse [7]. According to the report, 35.9% of the images depicted sexual assault and 64.1% showed the victim in sexually provocative poses [7]. Over half of the images depicted children under the age of 8 years, and some of the pictures were of babies and toddlers with roughly 1/3 of these images depicting sexual assaults. In addition, 68.5% of the sadistic sexual assault images, which includes bestiality, bondage, and torture, involved children under the age of 8 years. Finally, the majority (83%) of the children in the images were female [7].

After examining 800 commercial websites reported to cybertip.ca, three pornographic themes were identified: innocence, adult sexuality and pornography, and darkness and depravity [7]. The websites using the "innocence theme" tended to have images of younger children (toddlers to elementary school age) posing rather than being sexually abused, and the website utilized bright colors, toys, and words like "angel" and "pure." The adult sexuality and pornography themed websites used sexually explicit and slang terminology, such as "slut" and "Nymphet," and the children were depicted as promiscuous and willing sex partners. In addition, the images were likely to be of children provocatively posing and being sexually abused. Finally, the darkness and depravity pornographic theme used words such as "pedophile" and "sick," and focused on images depicting the sexual abuse of children by either other children or adults. Also, these websites sometimes posted sexually explicit abuse images of babies and toddlers [7].

2 Child Pornography Image Preference Scale

A literature review of the available scales or measurements, which assessed pornography preference, yielded scarce results regarding images of child sexual victimization. A few surveys in the area of media preference or sexual abuse attitudes included: Sexual Opinion Survey [8], Internet Sex Screening Test, Attitudes toward Sexual Abuse [9], and the Internet Behaviours and Attitudes Questionnaire [10]. However, the majority of the media preference scales only included one item, if they included one at all, related to child sex themes, thereby measuring a general genre of child pornography rather than any specific content or themes within child pornography collections [c.f., 11]. In addition, the scales assessing sexually deviant attitudes tended to focus on the relationship between violent adult pornography and violence toward women or attitudes regarding hands-on child sex abuse [c.f., 9].

Therefore, a new questionnaire was developed to assess the respondent's level of preference for various content-specific forms of child pornography, such as the

choice of victimization, age, and sex of the child. The items included for the newly developed Child Pornography Image Preference Scale (CPIPS; see Appendix) was selected based on their face validity, and the author developed the survey with the assistance of a local law enforcement agency in order to better understand the range of images routinely discovered in child pornography collections. As previously discussed, a literature review identified some recent trends in the types of child sex abuse images available on the Internet, so all of these resources were considered in the development of the survey.

The CPIPS was a likert rating scale [12], which asked the respondents to rate their level of agreement or disagreement to each statement. Since some respondents might be ambivalent regarding their preference to certain forms of child pornography, the author used an odd numbered likert scale (5-point) in order to provide an ambivalent or “neutral” item response. The CPIPS included 21 items scaled from 1 (Strongly Do Not Prefer) to 5 (Strongly Prefer).

The newly devised scale was pilot tested and reviewed by professionals in the field who provided preliminary feedback regarding the nature of the statements (clarity, simplicity, ambiguity). Feedback suggested the CPIPS had strong face validity, and the items appeared to be measuring the desired constructs.

3 Empirical Test of the CPIPS

Participants. Respondents were voluntarily recruited via the Internet by publicizing or advertising the study using various online resources including chat rooms, bulletin boards, discussion forums, and social media websites. This sampling methodology, often referred to as snowballing, allowed the author to target respondents from the “general population of Internet users.” In addition, Internet-based research designs increase the likelihood of self-disclosure since the respondents feel anonymous when completing online surveys rather than studies involving face-to-face interaction. In order to participate in the study, the respondents had to indicate on the demographics questionnaire that they were at least 18 years of age or older and were currently permanent residents of either the United States, United Kingdom, Australia, or Canada. The participants were required to provide consent, and they were able to quit the survey at anytime.

Design and Procedure. The Child Pornography Image Preference Scale was a part of a larger study assessing the personality characteristics of self-reported child pornography consumers [see 13]. However, this scale was only available online for approximately 2-weeks. Those respondents who self-reported any of the following behaviors were linked to the CPIPS: “knowingly accessing, viewing, downloading, exchanging and/or sharing pornography images of individuals under the age of 18 years.”

The online survey started with a home page, which acted as a consent form to which the respondents had to agree or decline to participate. If the prospective respondents agreed, they had to click on the “I Agree” button in order to participate.

After clicking on the “I Agree” button, the respondents were asked to complete the questionnaires, which would take approximately 20 to 30 minutes to complete in total. Once the questionnaires were completed, the participants were taken to the survey’s “Debriefing” page where the study’s true intentions were revealed, and the respondents had to decide whether to submit (opt-in) or withdraw their responses (opt-out) from the final dataset.

At no time were the respondents asked for any identifying information (e.g., name). In order to protect the respondents’ anonymity and confidentiality, they were provided with an ID number, which the database randomly assigned to the participant’s responses. Thus, the responses to the questionnaires were not linked or matched to any particular individual, which was extremely important to increase the participant’s confidence in self-disclosing criminally-sanctioned behaviors (e.g., exchanging child pornography). As for the questionnaire items, the items were forced choice; however, the respondents could respond by endorsing, “decline to answer,” to each survey item in order to meet the Institutional Review Board’s requirements.

4 Results

As stated in the methodology section, all respondents reporting some level of intentional child pornography use were linked to the CPIPS in order to assess their preference for child pornography images. For the two weeks this scale was available, two child pornography users ($n = 2$) completed and submitted their responses to the CPIPS. Due to anonymity, the two child pornography users were randomly assigned an ID number, and they will be referred to as #2488 and #297.

As shown in Table 1, child pornography user #2488 was more likely to prefer pornographic images of teens. In addition, #2488 was indifferent towards nonsexual and sexual images of children posing and did not prefer pornographic images of children perform sexual acts on themselves (e.g., self-masturbation) or pornographic images of children from different racial or ethnic backgrounds. Overall, this respondent appeared to prefer pornographic images of teens, regardless of whether the child was posing in a nonsexual or sexually provocative manner.

However, child pornography user #297 self-reported stronger preferences for a wider range of image content. As shown in Table 8, #297 strongly preferred all of the child pornography image content except pornographic images of children who are developed (e.g., pubic hair) and non-pornographic images of children posing, both of which received a rating of 1 (Strongly Do Not Prefer). Overall, this respondent preferred a wider range of image content except those images, which depicted older, post-pubescent children, and images that were less sexual or nonpornographic in nature.

Table 1. CP images preference ratings for CP User #2488 and #297

Image Content	CP User	
	ID #2488	ID #297
Infants	1	5
Toddlers	1	5
Pre-Teens	1	5
Teens	4	5
Developed	1	1
Boys	1	5
Girls	1	5
Child-Only	1	5
Adult-Child	1	5
Nonsexual Posing	3	1
Sexual Posing	3	5
Genitals	2	5
Child w/ Child	1	5
Child w/ Adult	1	5
Child w/ Self	2	5
Child w/ Animals	1	5
Power Over Child	1	5
Bondage	1	5
Violence	1	5
Racial or Ethnic	2	5
Novel or Unusual	1	5

Note. Values represent respondent's self-reported preference for certain types of child pornography images. Values scaled 1 (Strongly Do Not Prefer), 2 (Do Not Prefer), 3 (Indifferent), 4 (Prefer), and 5 (Strongly Prefer).

5 Discussion

Based on the limited data collected, this author was unable to draw statistical inferences from the results of the Child Pornography Image Preference Scale. However, there were notable descriptive differences in the types of images preferred by child pornography user #2488 and #297. In general, respondent #2488 appeared to prefer pornographic images of teens, regardless of whether the child was posing in a nonsexual or sexually provocative manner, while respondent #297 preferred a wider-range of sexually explicit image content.

Although data was collected from two self-reported consumers of child pornography, this study provides preliminary evidence that individuals are willing to self-report deviant image preferences in an online, anonymous environment. In addition, the face validity of the CPIPS appears promising since the responses to the CPIPS for #2488 and #297 yielded a different pattern of endorsements, which suggest a different pattern of interest. In future studies, this pattern of interest or preference in the child pornography images may reflect different motivations and personality characteristics.

Again, research suggests child pornography users collect sexualized images of children for a variety of reasons. Interviews with child pornography users have suggested some offenders move “through a variety of pornographies, each time accessing more extreme material” [14, p. 343] as a result of desensitization or appetite satiation, which lead to collecting and discovering other forms of deviant pornography [15]. Also, some of the consumers stated they downloaded the images simply because they were available and accessible, making the behaviors primarily a result of compulsivity rather than a specific sexual interest in children [16].

Child pornography consumers exhibit a complex array of sexual interests, which may be representative of a more general level of paraphilic tendencies rather than a specific sexual interest in children. Research suggests the majority of Internet child pornography users are collecting a wider range of deviant pornography, which may reflect a general level of sexual deviance rather than a specific paraphilia, such as pedophilia. The extent to which a child pornography user is likely to be a pedophile or a “dissident” expressing a wide range of sexual interest will best be understood through rigorous, empirical research. Overall, with future replication and empirical validation, the Child Pornography Image Preference Scale may be the first measure of people’s preferences for child pornography images.

6 Conclusion

This study demonstrated that respondents are willing to report their level of preference for various types of child pornography. Understanding the types of images preferred by child pornography users may assist in understanding their motivation for engaging in this illegal behavior. After all, previous research suggests those individuals who possess child pornography images are not at a greater risk for becoming child sex offenders [c.f., 17, 18]. Therefore, the size and content of their collections may indicate a general need or addiction to sexual stimuli, such as other

forms of deviant pornography, rather than an intense sexual arousal toward children (pedophilia). Finally, future research may be able to identify whether personality characteristics (i.e., aggressive) are predicative of image preference (i.e., violent, sadistic images) for child pornography consumers.

Overall, Internet-based research will continue to increase in popularity due to its advantages over more traditional forms of methodology, such as the accessibility of target populations with narrow interests. With regards to child pornography research, the Internet may be the best place to analyze both the users and behaviors due to the perceived anonymity and cloak of safety offered by the Internet. This type of research will continue to be a socially sensitive topic, but further empirical validation within the law enforcement and academic community is needed for the Child Pornography Image Preference Scale.

References

1. Wolak, J., Mitchell, K., Finkelhor, D.: *Internet Sex Crimes Against Minors: The Response of Law Enforcement*. National Center for Missing & Exploited Children, Washington, DC (2003)
2. Wells, M., Finkelhor, D., Wolak, J., Mitchell, K.: Defining Child Pornography: Law Enforcement Dilemmas in Investigations of Internet Child Pornography Possession. *Police Practice and Research* 8(3), 269–282 (2007)
3. Webb, L., Craissati, J., Keen, S.: Characteristics of Internet Child Pornography Offenders: A Comparison with Child Molesters. *Sex Abuse* 19, 449–465 (2007)
4. Child Exploitation and Online Protection Centre: *Strategic Overview 2007-2008*. Child Exploitation and Online Protection Centre, London (2008)
5. Child Exploitation and Online Protection Centre: *Strategic Overview 2008-2009*. Child Exploitation and Online Protection Centre, London (2009)
6. Internet Watch Foundation: *2008 Annual and Charity Report* (April 2009), <http://www.iwf.org.uk/>
7. Bunzeluk, K.: *Child Sexual Abuse Images: An Analysis of Websites by cybertip.ca*. Canadian Centre for Child Protection, Winnipeg (2009)
8. Fisher, W., Byrne, D., White, L.A., Kelly, K.: Erotophobia-erotophilia as a Dimension of Personality. *Journal of Sex Research* 26, 123–151 (1988)
9. Briere, J., Henschel, D., Smiljanich, K.: Attitudes toward Sexual Abuse: Sex Differences and Construct Validity. *Journal of Research in Personality* 26, 398–406 (1992)
10. O'Brien, M.D., Webster, S.D.: The Construction and Preliminary Validation of the Internet Behaviours and Attitudes Questionnaire (IBAQ). *Sex Abuse* 19, 237–256 (2007)
11. Bogaert, A.F.: Personality, individual differences, and preferences for the sexual media. *Archives of Sexual Behavior* 30(1), 29–53 (2001)
12. Likert, R.: A Technique for the Measurement of Attitudes. *Archives of Psychology* 22(140), 44–53 (1932)
13. Seigfried-Spellar, K.C., Rogers, M.K.: *Discriminating Self-Reported Internet Child Pornography Users by Individual Differences and Sex* (2012) (submitted for publication)
14. Quayle, E., Taylor, M.: Paedophiles, Pornography and the Internet: Assessment Issues. *British Journal of Social Work* 32, 863–875 (2002)
15. Quayle, E., Taylor, M.: Model of Problematic Internet Use in People with a Sexual Interest in Children. *CyberPsychology & Behavior* 6, 93–106 (2003)

16. Basbaum, J.P.: Sentencing for Possession of Child Pornography: A Failure to Distinguish Voyeurs from Pederasts. *Hastings Law Journal* 61, 1–24 (2010)
17. Hessick, C.B.: Disentangling Child Pornography from Child Sex Abuse. *Washington University Law Review* 88, 853–902 (2010)
18. Malamuth, N., Huppin, M.: Drawing the Line on Virtual Child Pornography: Bringing the Law in Line with the Research Evidence. *N.Y.U. Review of Law and Social Change*. 31, 773–790 (2007)

Appendix: Child Pornography Image Preference Scale (CPIPS)

Below are a number of items related to your preference for sexually explicit websites. You will probably find that you like some of the items and dislike some others and that is okay. We realize that everyone is different, so please respond as honestly as you can.

Remember, this survey is completely confidential and anonymous, meaning there is no way that your responses will be linked back to you.

STRONGLY DO NOT PREFER	2	<i>indifferent</i>	4	STRONGLY PREFER
1		3		5

1 if you *strongly do not prefer* the item

2 if you *do not prefer* the item

3 if you feel *indifferent or neutral* about the item

4 if you *prefer* the item

5 if you *strongly prefer* the item

1. Pornographic images of infants.
2. Pornographic images of toddlers.
3. Pornographic images of preteens.
4. Pornographic images of teens.
5. Pornographic images of children who are developed (e.g., pubic hair).
6. Pornographic images of boys.
7. Pornographic images of girls.
8. Pornographic images featuring only children.
9. Pornographic images featuring children with adults.
10. Non-pornographic images of children posing.
11. Images of children posing sexually or provocatively.
12. Pornographic images of children that focus on the genitals.
13. Images of children performing sexual acts on other children.
14. Images of children performing sexual acts on other adults.

15. Images of children performing sexual acts on themselves (e.g., self- masturbation).
16. Images of children performing sexual acts on or with animals.
17. Pornographic images depicting power or control over the child.
18. Pornographic images of children depicting bondage (e.g., being tied-up).
19. Pornographic images of children depicting violence (e.g., hit, kicked).
20. Pornographic images of children from different racial or ethnic backgrounds.
21. Pornographic images of children that are novel or unusual.

Cybercrime, Censorship, Perception and Bypassing Controls: An Exploratory Study

Ibrahim Baggili¹, Moza Al Shamlan², Bedoor Al Jabri², and Ayesha Al Zaabi²

¹ Tagliatela College of Engineering, Department of Electrical and Computer Engineering and Computer Science, University of New Haven, CT
Ibaggili@newhaven.edu

² Zayed University, College of Technological Innovation
Advanced Cyber Forensics Research Laboratory
m80001612@zu.ac.ae, {budur44, ayesha.alzaabi}@hotmail.com

Abstract. Countries have employed the Internet proxy as a censorship mechanism for various reasons. Concurrently, cyber criminal activities continue to rise. This research explores peoples' engagement in bypassing the Internet proxy and if it is related to cyber criminal engagement. Through an experimental design, participants were randomly assigned to three groups. Using manipulation paragraphs, in the first group (Group 1), a positive view on the Internet proxy was presented. In the second group (Group 2), a negative view on the Internet proxy was presented. The third group (Group 3) was used as the control group, where the participants' view of the Internet proxy was not manipulated. All three groups were asked to self-report their rate of proxy bypass (SRPBE) and cybercrime engagement (CCI). The results indicated a significant positive correlation between self-reported cyber criminal engagement and self-reported proxy bypass engagement. The results also showed that individuals with more knowledge in computers are more likely to bypass the Internet proxy. However, individuals with better knowledge in computers are not necessarily the ones that are more likely to commit cyber criminal activities. The results were inconclusive on whether or not the manipulation paragraphs used had an effect on the participants' view of the Internet Proxy.

Keywords: Cybercrime, psychology, censorship, Internet proxy, UAE.

1 Introduction

With time the Internet continues to grow. More users today are engaged in the World Wide Web and are actively infused with this technology. The Internet World Stats website reveals the number of increasing Internet surfers in different regions of the world. Data also reveals that there are about fifty seven million surfers in the Middle East alone. In the case of the United Arab Emirates (UAE), it was determined that it has the fifth highest number of Internet users amongst other Middle Eastern countries [1]. Furthermore, it is one out of a number of countries that applies an Internet proxy to censor Internet content.

The reasons behind the employment of an Internet proxy may vary. In the UAE, the purpose could range from religious, to social, to political reasons [2]. One of these

reasons could also be to prevent cyber criminals from accessing and downloading hacking and exploitation tools. For example, when attempting to visit the website <http://remote-exploit.org>, a website that contains software that could be used for malicious purposes, we find that the Internet Proxy in the UAE prohibits access to such a website. If a primary reason for censoring Internet content is to prohibit users that are actively engaged in cybercrime from downloading hacking tools and content, it becomes important to investigate the relationship between bypassing the Internet proxy and cybercrime engagement.

Internet censorship remains a topic of debate despite the many reasons behind why an Internet proxy is applied. In the UAE for instance, Sheikh Abdulla Bin Zayed, Minister of Information and Culture is in favor of an open Internet, for he states that the UAE's Internet Service Providers should not block access to websites because every citizen is entitled to knowledge and learning [3].

Due to the restricted Internet access in the UAE, it is hypothesized that users may engage in ways to bypass the Internet proxy. The purpose and intentions behind such user activity is yet to be empirically examined. Understanding this relationship can shed light on the effectiveness of the Internet proxy, and whether it is fulfilling the purpose of evading cyber criminals from accessing illegal content.

2 Problem Statement

One of the reasons of employing an Internet proxy is to not only censor illegal content, but also to curb cyber criminal engagement. Currently, there is no formal published research that studies the relationship between cyber criminal engagement and proxy bypass engagement. It is important to study this relationship in order to validate the productivity of the Internet proxy.

3 Research Questions and Hypotheses

In this study, the researchers attempted to answer the following questions:

- Is there a relationship between Self Reported Proxy Bypass (SRPB) and cybercrime engagement?
- Is there a relationship between the level of knowledge in computers and SRPB?
- Can respondents be manipulated using manipulation paragraphs to affect their perception of the employment of an Internet proxy?

To answer the abovementioned questions three major hypotheses were formulated:

- H1: There is a positive correlation between self-reported cybercrime (CCI) and self-reported proxy bypass engagement (SRPB).
- H2: Individuals with better knowledge in computers are more likely to bypass the Internet proxy and engage in cybercrime.
- H3: Decreasing the positive perception of the Internet proxy increases self reported cybercrime and proxy bypass engagement.

4 Literature Review

4.1 Censorship and The Internet Proxy

Censorship is a widely implemented practice. Censorship is “The restriction of what people may say, hear, write, read, or see” [4]. The Fileroom website, an archive of various materials lists censored cases in different regions of the world (thefileroom.org). This website, in partnership with the National Coalition against Censorship (NCAC) aims to fight types of censorship that may violate the right of human expression. On the other hand, there are supporters of censorship, which are those who are usually applying it, which in many cases is the government, special interest groups, or individuals. Organizations and people that are pro-censorship look at it from a different perspective mostly perceiving it as a security measure to ensure the wellbeing and morals of societies [4].

There are different types of censorship ranging from censorship in tangible material such as books and magazines, to intellectual ideas, services, and the placement of restricting rules and regulations that prohibit people from exercising a particular action. When it comes to censorship of the Internet, the United Arab Emirates (UAE) is in the top ten ranks according to a research that The National newspaper has reviewed, along with China, Iran and Saudi Arabia. This list also includes the United States, Germany, France, Canada, Tunisia, and Bahrain [5]. For instance, Chinese Internet Service Providers (ISPs) censor topics like Tibet, Taiwan and Tiananmen (for political reasons), and even high profile websites like BBC and Voice of America. Moreover, the Iranian government censors content that may include pornography, politics, religion, and anything that may have a heavy western influence like music, videos or movies; resulting in a 10+ million blocked websites ranging from Wikipedia to YouTube to Amazon. Also, the Saudi Arabian government put a ban on any anti-Islamic websites along with women’s rights topics, gambling and pornography. The above mentioned countries are only a small sample of nations that apply censorship [6].

Although the stated reasons behind applying censorship may seem to be moral, there may be a dual purpose. For example, although Internet censorship may be perceived as a way of protecting children from “dangerous or disturbing ideas and information” [7], others may think of it as an effective way of controlling people’s minds. If we look at the cases of different countries, nations propose independent reasons behind applying Internet censorship. For example, in the UAE, if you try to access a blocked website, the following message appears on the browser: “Access to this site is currently blocked. This site falls under the prohibited content categories of the UAE’s Internet Access Management Policy”. The prohibited content categories are provided, and the reasons behind blocking a website can be political, social, cultural, or religious.

Censorship is an old concept and has been around for many years. Censoring may include the destruction of a certain work, banning it, or making it illegal to produce or sell [8]. In the following part of the literature review, censorship examples are discussed with relation to criminal activities.

Looking back at the second World War, Hitler, for instance, applied censorship and banned various books for the sole purpose of implementing a “Totalitarian philosophy” [9]. He attempted to control what the people in Germany read, how they thought, and what they believed in. The employment of his philosophy shifted people’s beliefs. Although he was a powerful man and tried to indoctrinate people’s minds with his ideology, many had opposed him. In reality, many had planned conspiracies and assassinations in hopes of terminating Hitler and his regimes [10].

In the 1970’s, alcohol was prohibited in the United States by the 18th amendment. During the World War, people felt the need to become patriotic and consume their time in conserving grain, rather than drinking alcohol. This prohibition led to organized crime [11]. Large quantities of alcohol were smuggled in from Canada, overland and via the Great Lakes [12], thus indicating how the alcohol ban led to an increase in the rate of crime and illegal activities.

In 1979, the Chinese government decided to restrict the number of people in Chinese families and allow most of them to only have one child; they called it the one-child policy [13]. The population of Chinese people made up a quarter of the world’s population when they were only using seven percent of the land on earth. This rise in the population was because of Mao Zedong who led the Chinese people into giving birth to as many children as possible between 1950 and 1960 to bury the United States in a human wave [14]. Although the one-child policy prevented at least 300 million births, and boosted prosperity [14], it led to a gender disproportion in the population. Since the Chinese society tends to favor a male inheritor, this led to the abortion of many female babies [15]. The policy in place led to illegal activities such as infanticide, human trafficking, having illegal children and sending them off to isolated regions where they cannot be found, and girls being picked up by gangs to be used for banned activities [16]. Despite the primary goal that was set behind applying this ban or policy, there is an indication of the rise in illegal activity due to the restrictions that this policy has placed upon the people of China.

In the mid nineteenth century, abortion in the United States was illegal. In addition to it being a crime at the time, it was perceived to be a sin as well [17]. Although abortion was prohibited, women always found a way around that. NARAL, which is a pro-choice American Foundation mentions an excerpt of a story from the book “Women speak out about abortion”. The passage indicates how women had sought illegal abortion, even in poor hygienic conditions, in order to secretly abort unwanted children [18]. Although abortion was illegal, it was discovered that about a million illegal abortions a year were performed in the U.S. [17], thus indicating how the employment of this law had failed to serve its purpose, and resulted in the increase of secretly performed abortions.

In the U.S., the government’s interference in banning certain violent video games is questioned. It has been stated that there is weak evidence with regards to the link between video games and youth aggressiveness; even when the video game industry was booming between 1994 and 2000, a decrease in the rate of crime was witnessed, [19] (this does not necessarily eliminate the relation between those two elements, yet it sets a possible indication of their negative correlation). One research study on the relationship between violent video games and its effect on children was conducted.

This research suggested that conducted experiments have proved the following similar results: “playing violent video games can indeed cause increases in aggressive thoughts, feelings and behaviors” [20]. Despite the findings of this research there is still controversy over this topic.

Censoring violence in media has also been a topic of debate. Those supporting this type of censorship are concerned about its effect on those watching such violence on television. Scientific studies have shown the connection between violence and media violence, but they have not been able to show a causal relationship between the two [21]. Despite the uncertainty of this relationship, the research emphasizes its negative effect on children. Opponents believe that despite the large number of research conducted on violence in the media, very few studies look into this issue in real life. One study was conducted by Dr. Brandon Canterwall in three countries (Canada, U.S., and South Africa) in order to identify the connection between media violence and the rate of crime. The study was conducted from 1945 to 1974 when the television was first broadcasted in the U.S. and Canada, yet banned in South Africa. In the U.S. and Canada, homicide rates doubled when television was first introduced, yet the rates remained stable in Africa [22].

Censorship and bans on literature and media are merely a barrier for some people; a barrier which can be almost always be overcome. When it comes to books that have been banned for political or social reasons, one can merely find an online book store from which to purchase the book, whether it be a physical copy or an electronic copy. In fact, there was a website dedicated to that purpose; Banned-Books.org. This website was an online bookstore that sold banned books, audiotapes and videos [23]. There have also been events protesting the banning of books, such as Banned Books Week, which has been an annual celebration of the “freedom to read” since 1982. The organization hosts events nation-wide across the United States where bookstores welcome authors that have been subject to censorship to read their books in the bookstores and to speak about being censored [24].

Newspapers or magazines may either be banned or contain censored articles within the printed pages because they could be culturally offensive, religiously unacceptable, or harmful to the image of political or royal persons. If this is the case, then individuals seeking the non-censored versions may be able to find it on the Internet, and in most cases, newspaper or magazine websites are not censored. “And so, if the government wants to ban the Sunday Times from the newsstands, it should block its website too. Or, of course, do neither.” [25].

In certain countries, movies showing at the cinemas or aired on television that contain extremely violent or sexual scenes are usually only shown after those scenes have been censored. The same is applied to music videos. Some music videos may not be shown on television because of their highly suggestive nature. This is easily circumvented by either purchasing a DVD (which in most cases are not censored), viewing them online through an illegal video streaming website, illegally downloading the movies, and downloading the censored scenes from file sharing or peer-to-peer services.

Most of the aforementioned examples illustrate a relationship between bans and undesired activity. However, until now, no research in the UAE has been conducted

with regards to the relationship between the Internet proxy and cyber criminal engagement. In this research, the aim is to reveal the nature of this relationship, and it is hypothesized that like most of the examples illustrated above, that cyber criminal engagement is positively correlated with bypassing the Internet proxy. This hypothesis is primarily based on the “all that is banned is desired” principle, and the idea that obstructing the freedom of users will trigger illegal activity or cybercrime.

4.2 Cybercrime

The UAE is ranked second as the most vulnerable of the Gulf countries to fall victim to cybercrimes. The world is more connected than ever before, and the credit goes to technology, because with its positive use arose its misuse. Therefore, studying cybercrime in the UAE is of critical importance.

We ask ourselves: What is cybercrime? A clear-cut definition does not exist, yet we know it when we see it, or when we experience it. There have been many attempts at defining cybercrime. For example, in the book *Cybercrime: vandalizing the Information Society*, the author differentiates between computer crime and cyber crime. Computer crime is “a crime in which the perpetrator uses special knowledge about computer technology”, whereas cybercrime is “a crime in which the perpetrator uses special knowledge of cyberspace” [26]. Shinder (2002) also explains that there are different categories of cybercrime. Mainly, there are two basic categories: Violent and non-violent cybercrime. Under each category, different types or subcategories were suggested. Under non-violent cybercrime; cyber trespass, cyber theft, cyber fraud, and destructive cybercrime. On the other hand, violent cybercrime encompasses cyber terrorism, assault by threat, cyber stalking, and child pornography [26].

In 2002 Furnell further illustrates the public’s attitude and awareness of the cybercrime issue. A survey was conducted in the U.K. in order to determine the degree of the public’s understanding of cybercrime and how the media has played a role in shaping that understanding. Consequently, the survey addressed questions that revolved around three main issues: using unlicensed software, unauthorized use of IT facilities, and password sharing. The survey indicated that the participants were involved in the three activities to some extent. However, despite the respondents’ engagement in those issues, over 80% of the participants acknowledged that they understand that their actions result in cybercrime [27].

The aforementioned results of the survey conducted in the U.K. reveal people’s acknowledgement of the cybercrime issue, yet they still practice it. Why do people commit cybercrime? What is their motive or personal reasoning? Can their perceptions be affected in order to halt this activity? An article mentions various reasons of engaging in cybercrime, ranging from a user’s excitement to challenge him/her self, ease of anonymity, to holding a grudge [28]. Although individuals may engage in such activities for different reasons, manipulating perception may have an effect on users’ ideas of this activity.

4.3 Perception

Scientifically, perception is the way people interpret what they sense in the environment around them; the way they view the world [29]. Perceptions are provisional much similar in the way in which scientific hypotheses are provisional; as in peoples' perceptions of anything around them changes when they learn new information about them [30].

Any information that people gather or are given can change the way they view a certain object or idea. Peoples' perceptions are easily molded and changed both directly and indirectly. John Stauber and Sheldon Rampton wrote a book unfolding and describing what it takes to create public opinion, as well as revealing evidence of opinion manipulation from the early 20th century [31].

Edward Bernay states in his book Propaganda that "scientific manipulation of public opinion was necessary" and determined that "a relatively small number of persons pull the wires which control the public mind" [32]. He, in fact, was one of those individuals that formed peoples' opinions about numerous concepts and products in the United States. Examples of which range from promoting bacon as breakfast food, popularizing smoking cigarettes among women, to presenting the first World War as a positive concept that benefits the world [31].

The media especially plays an enormous role in influencing peoples' perceptions about ideas. Perception manipulation has been practiced in all forms of media. Some even say the type of medium chosen in order to get a message across may even be more important than the message itself [33-34]. Ultimately, it will get the desired effects.

Ball-Rokeach and DeFleur (1976) state that "audience dependency on media information resources [is] a key interactive condition for alteration of audience beliefs, behavior, or feelings as a result of mass communicated information" [35]. Today's media is no longer constrained to television, radio and print magazines. It seems to be fast-paced, unstoppable and unrestrained. The properties that make up what global media resulted in an almost involuntary reaction which is the manipulation of the public's opinion and behavior [36].

Johnson in 2007 set an example of such impact of perception manipulation once again but from a military viewpoint. The simple manipulation of a "flash of an image" such as images of dead women and children can change the public's perception of war. When war is depicted in such a way, people are influenced into considering how negative war is, and pushes back the idea of any benefits that war may bring [36].

Experimental research has grown more popular and prevalent in social psychological research. In this kind of research, components include a manipulation of at least a single independent variable, and the randomized assignment of participants of the research to the manipulation or condition [37].

For example, there are numerous examples of manipulation within medical experiments and research. Influencing patients into believing they are receiving treatment (when in reality they are not) can result in change in their behavior and in cases even cause physical change. This goes to show that one's perception can be

molded into something that may contradict one's original ideas, beliefs, and even reality.

In one study by Massachusetts Institute of Technology scientists, it was shown that magnetic fields can alter human brain operation; more specifically their moral judgment. The groups of subjects in the experiment were asked to read short stories and were then asked to decide whether the actions of the characters in the stories were morally acceptable or unacceptable. One group was then subjected to transcranial magnetic stimulation, while another was not. The temporary stimulation appears to have changed the answers of the first (manipulated) group where the results showed that the subjects were indecisive about what was morally acceptable or unacceptable, and that they focused on the outcome of the story as opposed to the intention of the characters in the stories [38].

The University of Harvard conducted research that explored how effective the placebo effect can be on people that suffered from pain, arterial hypertension and asthma. Some subjects were given the actual medication, while another group of subjects were not given any medication, and instead given a pharmacologically inactive substance, but were led to believe that they were given legitimate medication. Approximately 40% of the subjects who were administered fake medicine indicated that they felt relief from their physical pain [39].

Other instances in the medical field that experiment with the use of the placebo effect include surgical procedures as well. Hospital patients suffering from chest pain caused by chronic heart ischemia were separated into two groups: those who underwent the surgical procedure to rectify it, and those who were only led to believe that they had the procedure done (by preparing them for the operation, sedating them, and incising their skin so it appears that they have gone through surgery). Those who were operated on legitimately showed 40% improvement, while those who went through the "pretend" surgery showed 80% improvement, [39]. Again, this demonstrates the power of manipulation of people's perception.

5 Methodology

This study uses an experimental approach. Participants were randomly assigned into three groups. In the first group (Group 1), a positive view on the Internet proxy was presented. In the second group (Group 2), a negative view on the Internet proxy was presented. The third group (Group 3) was used as the control group, where the participants' view of the Internet proxy was not manipulated. All the participants in each group were asked to complete two self-reported instruments CCI which represents an index measure of cyber criminal engagement and SRPBE which aims to measure the engagement of the participants in bypassing the Internet proxy. All participants were also asked a single question about their level of knowledge of computers. The responses were analyzed in order to test the following hypotheses:

H1: There is a positive correlation between self-reported cyber crime (CCI) and self-reported proxy bypass engagement (SRPBE).

H2: Individuals with better knowledge in computers are more likely to bypass the Internet proxy and engage in cyber crime.

H3: Decreasing the positive perception of the proxy increases self reported cybercrime.

5.1 Theoretical Constructs

Figure 1 illustrates the different variables and predictors in this research. The two predictors are Proxy Bypass Engagement and Proxy Perception. The independent variable is self-reported cybercrime engagement.

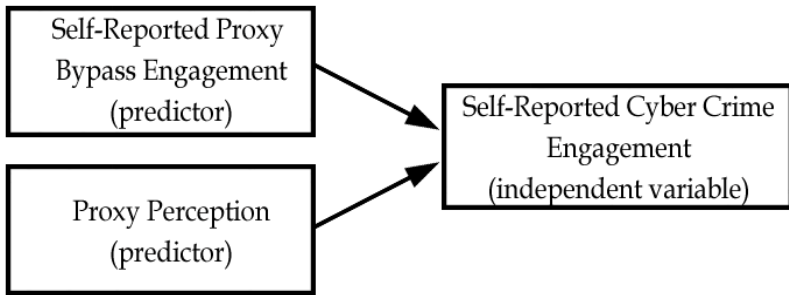


Fig. 1. Theory diagram

5.2 Instruments

5.2.1 Proxy Bypass

The researchers created this self-reported measure in order to compare its relationship with self-reported cybercrime. The survey encompasses questions around the respondents' degree of proxy bypass engagement. The types of questions used were meant to reveal the degree to which the participants were involved in proxy bypass, and the reasons behind their engagement in this activity (to challenge one's skills, access certain websites, or for malicious reasons). A measure for each respondent was then created in order to assess each participant's level of bypassing the Internet proxy.

5.2.2 Proxy Perception

This survey instrument was disseminated to three separate groups in which the "manipulation page" was phrased in a manner that either advocated the employment of the proxy (Group 1), criticized it (Group 2), or the survey was distributed without any prior effort of manipulation (Group 3, control group). The aim of the first two groups was to attempt to create a biased opinion, in a positive or negative manner, so as to find out whether the "manipulation page" would influence participants' perception of the proxy. The manipulation paragraphs used are shown in Figures 1 and 2.

The Internet proxy has been employed on websites violating the UAE's "Prohibited Content Categories". It has been placed for the purpose of preserving the country's culture, religion, and to prohibit illegal cyber activity. Since the emergence of the Internet, different types of websites have surfaced. Despite the positive progression of the Internet, its downside lies within the many existing abusive websites that need to be controlled through the implementation of the Internet proxy.

Fig. 2. Positive Manipulation Paragraph

The service providers have been blocking a large number of websites for various reasons. Unfortunately, their intervention, through the use of the Internet proxy, has gone beyond reason. Why rob users from their browsing rights? Every individual is accountable for his/her actions, and the Internet proxy will not resolve any concerns; it will only deprive users from a full learning experience.

Fig. 3. Negative Manipulation Paragraph

5.2.2.1 *Self-Reported Cyber Crime*

Following the questions around self-reported proxy bypass engagement, participants were also asked to answer questions related to self-reported cybercrime. The self-reported cybercrime survey was originally created by Dr. Marcus Rogers, a professor at Purdue University. The Index measurement was created by Dr. Rogers is an instrument that has been repeatedly used and cited in various studies [40] [41] [42].

6 Research Protocol

6.1 Participants

After seeking ethical clearance, the researchers were able to disseminate a survey to 4,473 e-mail addresses, which included students and faculty. The researchers gathered data for a period of two weeks, and the total number of participants in this study was (n=107) after eliminating 188 participants with incomplete responses 93 of which were females, and 14 of which were males. The program Gpower was used in order to determine the sample size necessary for the study.

- For a one-tailed test, with medium effect size (0.5), an alpha of (0.05) and a power (0.9), the recommended sample size is 140.
- For a two-tailed test, with medium effect size (0.5), an alpha of (0.05) and a power (0.9), the recommended sample size is 172.

It is unfortunate that the sample size the researchers received was not significantly high, becoming a limitation of this study. However, the yielded results illustrate theoretical saturation and a reasonable effect. The authors also note a high dropout rate, and this was expected due to the measurement's length.

6.2 Study Protocol

1. The following process was followed:
 - a. Ethical clearance was sought from the university research office.
 - b. A pilot survey test was conducted prior to the distribution of the survey.
 - c. The emails were randomly assigned to the three different groups. The surveys were then disseminated via email to Zayed University faculty, staff and students.
 - d. The consent form was included as the first page of the survey for all participants that would agree to contribute to the study. Participants were instructed to carefully read and agree to the pre-consent forms before beginning the survey. The survey could not be completed if participants did not agree to the consent form.

6.3 Reliability

In order to show that a set of data is internally consistent, it is generally accepted that the reliability measure Cronbach’s alpha should be greater than 0.7. In this research, both CCI and SRPBE surpassed 0.7. Consequently, the measurements show a level of acceptable internal consistency. Table 1 illustrates the reliability measure results.

Table 1. Reliability statistics of SRPBE and CCI

Cronbach’s Alpha	Variable	N of Items
.929	SRPBE	28
.803	CCI	60

6.4 Data Analysis

After analyzing the data, 188 incomplete responses were eliminated. The data was analyzed using a variety of statistics. Primarily, the data was tested for normality and outliers using Q-Q plots and box plots. It is important to note that a total of 12 responses were eliminated after closely examining the Q-Q and box plots. In order to test whether correlations existed between a set of measures, Pearson’s correlation was used. Additionally, Analysis of Variance (ANOVA) was used in order to test the effect of the included manipulation paragraphs in the surveys for groups 1 and 2. After eliminating outliers, the total number of participants was 95.

6.5 Demographics

The demographics were analyzed to gain a better understanding of the sample as shown in Table 2.

Table 2. Sample demographics

Demographic Variable	N of Items	Population %
Gender		
Females	82	86%
Males	13	14%
Age		
Less than 17	1	1%
17-20	37	39%
21-25	31	33%
26-30	4	4%
Above 30	22	23%
Education Level		
High school	8	8%
Undergraduate	59	62%
Graduate	8	8%
Postgraduate	20	21%
Academic Major/Expertise		
Business	19	20%
Education	13	14%
Liberal arts	8	8%
Health sciences	15	16%
Social sciences	5	5%
IT/ Computers	17	18%
Other	18	19%
Level of computer knowledge		
Poor	1	1%
Fair	8	8%
Average	33	35%
Good	40	42%
Excellent	13	14%

7 Results and Discussion

7.1 Hypothesis 1

H1: There is a positive relationship between self-reported cybercrime (CCI) and self-reported proxy bypass engagement (SRPBE).

In this research, it was hypothesized that a significant positive correlation would exist between self-reported cyber criminal engagement and self-reported proxy bypass engagement. Table 3 illustrates that a significant correlation does exist between those two variables. A correlation is significant at the 0.01 level, meaning that there is a 99% chance (1-0.01) that the correlation is positive and equal to 0.285.

Table 3. Pearson correlation between SRPBE and CCI

Variables	N	Significance level (2-tailed)	Pearson Correlation
SRPBE x CCI	95	0.05	0.285**

**** The correlation is significant at the 0.01 level (2-tailed).**

This study examined relationship between self-reported proxy bypass and self-reported cyber crime engagement. The results found illustrate a relationship similar to the literature review. This result can be interpreted in different ways. One way of interpreting the result is that individuals that are engaging in cyber criminal activities are also the same individuals that are bypassing the Internet proxy. Another plausible explanation for this result is similar to the notion discussed in the literature review, which indicated that “What is banned is desired”. Individuals bypassing the Internet proxy are indeed engaging in cyber criminal activities, which are activities that are banned by society. H1 is supported from the experimental results, and thus it is accepted.

7.2 Hypothesis 2

H2: Individuals with better knowledge in computers are more likely to bypass the Internet proxy and engage in cybercrime.

For the SRPBE measure (Table 4) the means illustrate that those with excellent knowledge in computers have the highest SRPBE mean. This illustrates that individuals with excellent knowledge in computers are the ones that are engaging in bypassing the Internet proxy more often. A plausible explanation for that is that certain technical skills are needed in order to bypass the Internet proxy. This part of the hypothesis is accepted – that individuals with more knowledge in computers are more likely to bypass the Internet proxy.

Table 4. Mean SRPBE for computer knowledge

Computer Knowledge	Mean SRPBE	N	St. Deviation
Average	12.30	33	14.92
Excellent	25.20	13	23.09
Fair	11.63	8	13.00
Good	13.60	40	17.46
Poor	0.00	1	
Total	14.42	95	17.42

As for the mean in the CCI measure (Table 5), those that have good computer knowledge show the highest cybercriminal engagement. This indicates individuals with at least good knowledge have engaged in cyber criminal activities. A plausible explanation for that is that most people have engaged in cyber criminal activities such as downloading illegal music, software and movies from the Internet. It is then plausible that one does not need strong technical knowledge in computing to simply

engage in cyber criminal activities. This part of the hypothesis is rejected, because the results illustrate that individuals do not need high levels of computer knowledge to engage in cyber criminal activities.

Table 5. Mean CCI for computer knowledge

Computer Knowledge	Mean CCI	N	St. Deviation
Average	11.06	33	10.97
Excellent	11.31	13	7.17
Fair	9.90	8	9.98
Good	13.10	40	10.60
Poor	0.00	1	
Total	11.74	95	10.21

7.3 Hypothesis 3

H3: Decreasing the positive perception of the proxy increases self reported cybercrime engagement and proxy bypass engagement.

The means illustrated for the control group in both SRPBE (Table 6) and CCI (Table 7) measures reveal that the manipulation paragraphs may have had an effect. This is illustrated in the increasing mean from Group 1 to Group 3 in the SRPBE and CCI. This indicates that decreasing the positive perception of the proxy increases self-reported cyber crime and proxy bypass engagement.

Table 6. Group means for SRPBE measure

Group	Mean	N	St. Deviation
1 (Positive view)	12.23	26	17.27
2 (Negative view)	15.16	31	19.32
3 (Control)	15.32	38	16.20
Total	14.42	95	17.42

Table 7. Group mean for CCI measure

Group	Mean	N	St. Deviation
1 (Positive view)	10.92	26	10.18
2 (Negative view)	11.81	31	9.86
3 (Control)	12.24	38	10.74
Total	11.74	95	10.21

When applying ANOVA in order to examine if the means in the groups were significantly different from one another in CCI and SRPBE, the results revealed that there was little significance. Reasons of this insignificance are unclear and may vary. It is possible that participants did not spend their time reading the manipulation paragraphs; therefore their perception was not significantly manipulated. Also, the

sample size could be another factor; if the sample size was larger, results of a possible significance could have been more obvious. The researchers also question the viability of the manipulation paragraphs used, perhaps the words that were used were not strong enough to manipulate the participants' perceptions.

In reference to the literature review, manipulating perception has proven to have had an effect on individuals. Although it is unclear whether H3 is accepted or rejected, we do observe a trend in the means of both CCI and SRPBE of the means increasing from Group 1, to Group 3. The researchers expected that the mean of the control group would be in the middle – between the positive and the negative, but the results indicated otherwise. It is plausible that if the participants are not provided with a manipulation paragraph, that they are more likely to reveal their true opinion, this could be the reasons that both CCI and SRPBE means are higher in the control groups. Hypothesis 3 is therefore rejected. The authors note that a more comprehensive study using various other manipulation techniques should be conducted in order to re-examine the effect of manipulation on self-reported cybercrime and proxy bypass engagement.

8 Limitations

There are some limitations in this research. Primarily, there was a significant difference between the number of male and female participants in the experiment. This was expected that given that the number of female students surpasses the number of male students at Zayed University. Moreover, due to time constraints, sending out the survey to other universities was not possible. An extended process of approval from the Human Board would have been required, and this study was set to conclude within a limited time frame. Therefore, only Zayed University students and faculty were engaged in this study. Furthermore, the number of completed surveys did not match the power calculations to witness a strong effect.

9 Future Research

This study was conducted only at Zayed University in the UAE. The UAE, as mentioned before, is one of the top ten countries when it comes to censorship of the Internet [5]. This research could be expanded in the future to include students from different universities across the UAE and different countries that employ an Internet proxy. This would be favorable to gain external validity. Future research might lead to the discovery of whether implementing a proxy is actually preventing individuals from committing cybercrime. Moreover, the results of this research can be compared to future findings of different countries.

10 Conclusion

In this research, the results indicated a significant positive correlation between self-reported cyber criminal engagement and self-reported proxy bypass engagement. The

results also showed that individuals with more knowledge in computers have a higher proxy bypass engagement. However, individuals with better knowledge in computers are not necessarily the ones that are more likely to commit cyber criminal activities. With regards to perception manipulation, the results are not conclusive on whether or not the manipulation paragraphs had an effect on people's view of bypassing the Internet proxy.

References

1. Internet World Stats: Middle East Internet Usage Statistics, Population, Facebook and Telecommunications Reports (2009),
<http://www.internetworldstats.com/stats5.htm>
2. OpenNet Initiative: Internet Filtering in the United Arab Emirates (2005),
http://opennet.net/sites/opennet.net/files/ONI_UAE_2009.pdf
3. Olsen, E.: Rare Criticism of Gulf State Internet Censorship (2002),
<http://blogcritics.org/culture/article/rare-criticism-of-gulf-state-internet/>
4. Day, N.: Censorship: Or Freedom of Expression. Learner Publishing Group (2001)
5. Kwong, M.: Reports High Website Censorship. The National Newspaper (2009),
<http://www.thenational.ae/news/uae-news/uae-reports-high-website-censorship>
6. Nick: Top 10 Countries Censoring the Web (2008),
<http://www.dailybits.com/top-10-countries-censoring-the-web/>
7. Vandergrift, K.E.: Intellectual Freedom, and Youth (1997),
<http://comminfo.rutgers.edu/professional-development/childlit/censorship.html>
8. Day, N.: Censorship: Or Freedom of Expression. Learner Publishing Group (2001)
9. Cortes, M.V.: Internet Censorship Around the World, University of Chile, Chile (2000),
http://www.isoc.org/inet2000/cdproceedings/8k/8k_4.htm
10. Schrader, P.: An Obsolete Honor: A Story of the German Resistance to Hitler. iUniverse (2008)
11. Kenney, K.: Prohibitions in the 1920s (2009),
<http://kim-kenney.suite101.com/prohibition-in-the-1920s-a90037>
12. 1920-30.com: Prohibition in the United States (2005),
<http://www.1920-30.com/prohibition/>
13. Hesketh, T., Lu, L., Xiang, Z.W.: The Effect of China's One-Child Family Policy after 25 years. The New England Journal of Medicine (2005),
<http://www.nejm.org/doi/full/10.1056/NEJMp051833>
14. Macartney, J.: Factfile: China's one-child policy. TimesOnline (2008),
<http://uyghuramerican.org/old/articles/1458/1/Factfile-Chinas-one-child-policy/index.html>
15. CNN: China to keep one-child policy. CNN (2008),
http://articles.cnn.com/2008-03-10/world/china.onechild_1_preference-for-male-heirs-traditional-preference-gender-imbalance?_s=PM:WORLD
16. Hall, A.T.: China's one policy and male surplus as a source of demand for sex trafficking in China (2010), <http://nfsacademy.org/wp-content/uploads/2011/02/Hall-Chinas-One-Child-Policy.pdf>

17. Feminist.com: History of Abortion. Touchstone Publishing (1998),
<http://www.feminist.com/resources/ourbodies/abortion.html>
18. NARAL Foundation: Choices: Women Speak Out About Abortion (2009),
<http://www.prochoiceamerica.org/media/fact-sheets/abortion-distorting-science-safety-legal-abortion.pdf>
19. Thierer, A.D.: Regulating Video Games: Parents or Uncle Sam? CATO Institute (2003),
<http://www.cato.org/publications/commentary/regulating-video-games-parents-or-uncle-sam>
20. Gentile, D.A., Anderson, C.A.: Violent Video Games: The Effects on Youth, and Public Policy Implications. Handbook of Children, Culture, and Violence. Thousand Oaks (2006),
<http://www.psychology.iastate.edu/faculty/caa/abstracts/2005-2009/05ga2.pdf>
21. Wagner, M.A., Wagner, J.: Should We Censor Violence in the Media? (2002),
http://www.yellodyno.com/pdf/Violence_in_the_media.pdf
22. Rhodes, R.: Hollow Claims About Fantasy Violence. The New York Times (2000),
<http://www.nytimes.com/2000/09/17/opinion/hollow-claims-about-fantasy-violence.html?pagewanted=all&src=pm>
23. Castillo, F.: Banned and Controversial Books (2008),
<http://banned-books.com/bblist.html>
24. Banned Books Week (2009), <http://bannedbooksweek.org/about>
25. Flanagan: The Futility of Censorship. Yahoo! News (2009),
http://business.maktoob.com/20090000404858/The_futility_of_censorship/Article.htm
26. Furnell, S.: Cybercrime: vandalizing the Information Society. Addison-Wesley Professional Publishers (2001)
27. Shinder, D.: Scene of the cybercrime: computer forensics handbook (2002),
<http://www.google.com/books?hl=ar&lr=&id=nQyucKKH6RgC&oi=fnd&pg=PR25&dq=cyber+crime+&ots=WVbXcJB81-&sig=Oszwa45V1PaTUTneqKDE9UuvX-I#v=onepage&q=cyber%20crime&f=false>
28. (Stocks, n.d.)
29. Heffner, C.: Introduction to Sensation and Perception (2004),
http://allpsych.com/psychology101/sensation_perception.html
30. Wikia: Perception: Perception and Reality (2001),
<http://psychology.wikia.com/wiki/Experimental:Perception>
31. Lacasse, M.: How the industry manipulated public opinion: Why you believe what you believe (2009), <http://www.healingdaily.com/beliefs.htm>
32. Bernay, E.L.: Propaganda. Kennikat Press (1972)
33. Budd, R.W., Ruben, B.D.: Beyond Media: New Approaches to Mass Communication. Transaction Publishers (1987)
34. Thayer, L.: On the Mass Media and Mass Communication: Notes Toward a Theory. Oxford University Press (1986)
35. Ball-Rokeach, S.J., DeFleur, M.L.: A dependency model of mass media effects. Communication Research 3, 3–21 (1976)
36. Johnson, B.K.: Dawn of the Cognetic Age: Fighting Ideological War by Putting Thought in Motion with Impact (2007),
<http://www.airpower.maxwell.af.mil/airchronicles/apj/apj07/win07/johnson.html>

37. Reis, H.T., Judd, C.M.: Handbook of Research: Methods in Social and Personality Psychology. Cambridge University Press, Cambridge (2000)
38. Bland, E.: Magnets can manipulate morality: study (2010),
<http://www.abc.net.au/science/articles/2010/03/30/2859767.htm>
39. Amaral, J.R., Sabbatini, R.M.: Placebo Effect: The Power of the Sugar Pill (1999),
http://www.cerebromente.org.br/n09/mente/placebo1_i.htm
40. Rogers, M.: A social learning theory and moral disengagement analysis of criminal computer behavior: An exploratory study. University of Manitoba, Canada (2001)
41. Baggili, I.M.: Effects of Anonymity, Pre-Employment Integrity and Antisocial Behavior on Self-Reported Cyber Crime Engagement: An Exploratory Study. Doctoral dissertation, Purdue University, USA (2009),
<http://dl.acm.org/citation.cfm?id=1834973>
42. Baggili, I.M., Rogers, M.: Self-Reported Cyber Crime: An Analysis on the Effects of Anonymity and Pre-Employment Integrity. Zayed University, UAE, Purdue University, USA (2009),
<http://www.cybercrimejournal.com/ibrahimmarcusIJCCJuly2009.pdf>

When Should Virtual Cybercrime Be Brought under the Scope of the Criminal Law?

Litska Strikwerda

University of Twente, Department of Philosophy, P.O. Box 217
7500 AE ENSCHEDE, The Netherlands
L.Strikwerda@utwente.nl

Abstract. This paper is about the question when virtual cybercrime should be brought under the scope of the criminal law. By virtual cybercrime I mean crime that involves a specific aspect of computers or computer networks: virtuality. Examples of virtual cybercrime are: virtual child pornography, theft of virtual items and the killing of an avatar (a virtual person). Drawing from philosophical ontology and legal philosophy I will establish what the necessary and sufficient conditions are for virtual cybercrime to obtain in order to count as crime under criminal law. I will also examine when virtual cybercrime meets these criteria.

Keywords: virtual (cyber-)crime, legal ontology, institutional facts, harm principle, offense principle, legal paternalism, legal moralism.

1 Introduction

The advent of computer technology has given rise to a new type of crime: cybercrime, which is crime that involves the use of computers or computer networks. Examples of cybercrime are: the spread of computer viruses, e-fraud and the distribution of child pornography by means of the Internet. The newest generation of cybercrime is virtual cybercrime. Virtual cybercrime is crime that involves a specific aspect of computers or computer networks: virtuality. For example, virtual child pornography, which does not consist of photographs or film material of real children engaged in sexually explicit conduct, but of entirely computer-simulated images of virtual children. And in the Netherlands, for instance, several minors were convicted of theft for the stealing of virtual items in the virtual worlds of the online multiplayer computer games *Habbo* and *RuneScape*. One of these cases was decided by the highest court in the Netherlands [1]. In Japan, finally, the police investigated the case of a woman who “killed” an avatar (a virtual person) in the virtual world of the online multiplayer computer game *MapleStory* [2]. But should acts like the aforementioned really be treated as crimes under criminal law? This paper aims to answer that question.

The abovementioned question belongs to the field of legal ontology. Ontology is the study of being, which is a branch of philosophy that is concerned with the questions of which kinds of things exist and how they are categorized according to

their differences and similarities. Legal ontology is an applied form of ontology that is specifically concerned with the question of how things are categorized *under law*. Legal ontology does not only study how existing things are categorized under law, but also how new things should be categorized under law [3].

This paper will study when the new phenomenon of virtual cybercrime should be categorized as crime under criminal law. This study will consist of the following three steps:

1. *Empirical exploration*: what is virtual cybercrime and how, if at all, is it treated within existing legal systems?
2. *Philosophical analysis*: what are necessary and sufficient conditions for virtual cybercrime to obtain in order to count as crime under existing law?
3. *Moral evaluation*: when does virtual cybercrime meet these criteria?¹

The first section of this paper will be concerned with the first step. It will study how cybercrime is treated within existing legal systems, provide a definition of cybercrime and determine the scope of the term. Then it will study the different meanings of the term “virtual” and define the term so that it can be explained what the new legal phenomenon of virtual cybercrime entails. At last, it will examine how virtual cybercrime is treated within existing legal systems, provide a definition of the term virtual cybercrime and determine its scope. In the second section of the paper I will establish what the necessary and sufficient conditions are for virtual cybercrime to obtain in order to count as a crime under existing law, which is the second step. I will analyze virtual cybercrime from the point of view of ontology and legal philosophy. I will establish that, in order to count as a crime under existing law, it is a necessary condition for a virtual cybercrime that it has an extravirtual consequence (a consequence outside the virtual environment). And that that is also a sufficient condition if the consequence is of such a nature that it can legitimate an interference with the liberty of citizens by means of penal law on the basis of one of Feinberg's liberty-limiting principles: the harm principle, the offense principle, legal paternalism or legal moralism. In the third section I will examine when the extravirtual consequence(s) of virtual cybercrime are of such a nature that (one of) the aforementioned liberty limiting principles can be invoked. This is the third step. Ultimately, I will come to the conclusion that virtual cybercrime should be brought under the scope of the criminal law when it results in extravirtual harm to others, offense, harm to the self or evils of other kinds.

2 Virtual Cybercrime: Legal Positioning, Definition and Scope

In this section I will examine what virtual cybercrime is and how, if at all, it is treated within existing legal systems. I will start with a description of the developing field of cybercrime. Against this background I will provide a definition of cybercrime and determine the scope of the term. Then I will study the different meanings of the term

¹ These steps are based on Koepsell [3].

“virtual” and define the term so that I can explain what the new legal phenomenon of virtual cybercrime entails. Next, I will examine how virtual cybercrime is treated within existing legal systems. At last, I will provide a definition of the term virtual cybercrime and determine its scope. Note that I will define (virtual) cybercrime in general terms so that the definition in principle applies to any country or jurisdiction worldwide.

2.1 Background: The Developing Field of Cybercrime

Crime is generally understood as a human act (or omission) prohibited by law. The prefix “cyber” refers to the use of computers or computer networks; it means “computer-mediated” [4,5,6]. Cybercrime thus consists of any computer-mediated human act that is prohibited by law.

Cybercrime poses a challenge, because the use of computers and computer networks allows for “new and different forms of (...) [human] activity that evade the reach of existing penal law” [7]. On the one hand, the use of computers or computer networks allows for new varieties of anti-social human activity that did not exist before the advent of computers and computer networks, e.g. the spread of computer viruses [5,7,8]. On the other hand, computers and computer networks can be used as a tool to commit traditional crimes, such as fraud, in different ways [5,6,7,8]. Legislators continuously need to determine which of the new and different forms of human activity that the use of computers and computer networks allows for have to be prohibited and which not. They have to enact new legal prohibitions in order to prohibit the new forms of human activity that computers or computer networks allow for or make existing legal prohibitions sufficiently broad as to include the different forms of human activity that computers and computer networks allow for. Mostly, the enactment of new penal provisions or the extension of existing penal provisions takes place at a national level. Which new and different types of human activity involving the use of computers and computer networks are outlawed precisely, varies significantly according to national legal systems, but there is a common ground [7].

The most familiar and most important international initiative to develop penal law aimed at cybercrime is the Convention on Cybercrime [6], which has been ratified by most member states of the Council of Europe and some other states, i.e. the United States of America and Japan. It is the only binding international instrument on this issue to have been adopted to date. The Convention on Cybercrime establishes “a common minimum standard of relevant offences” [6]. It defines nine types of new and different human activities involving the use of computers or computer networks and State Parties to the Convention agree to establish them as criminal offences under their domestic law, if they have not yet done so. The first five offence categories are: illegal access, illegal interception, data interference, system interference and misuse of devices. They concern new forms of human activity that did not exist before the advent of computers and computer networks. That is because they can only be carried out through the use of computers or computer networks. Since these offence categories concern new forms of human activity, they require signatory states to enact new legal prohibitions, if they did not prohibit these activities yet [7,8]. They can be

classified under the heading “computer crime” [5]. The next four offence categories are: computer-related forgery, computer-related fraud, offences related to (virtual) child pornography and offences related to infringements of copyright and related rights. They concern traditional crimes where computers or computer networks are used as a tool to commit the crime in a different way. Because states will already have criminalized these traditional crimes, these offence categories require them to make their existing laws sufficiently broad to extend to situations involving computers or computer networks if they did not do so yet [6]. They can be classified under the heading “computer-facilitated crime” [5].

Many states that have signed the Convention on Cybercrime have also signed its Additional Protocol [9], which criminalizes the following four types of human acts if committed through a computer system: the dissemination of racist and xenophobic material, racist and xenophobic motivated threat, racist and xenophobic motivated insult and denial, gross minimization, approval or justification of genocide or crimes against humanity. All of them are computer-facilitated crimes; the Additional Protocol aims to extend the penal law that already exists in most signatory states to the commission of traditional crimes through the Internet [9].

Last, the Convention on the Protection of Children against Sexual Exploitation and Sexual Abuse [10], which has been signed by most of the member states of the Council of Europe, establishes another relevant offence category. The aforementioned Convention obliges signatory states to take the necessary legislative or other measures to criminalize the solicitation of children for sexual purposes (“grooming”) through information and communication technologies. Grooming is a computer-facilitated crime: computers or computer networks are used as a tool to establish contacts that could also be established by means of non-electronic communications. Not all countries prohibit non-electronic variants of grooming, however, and the aforementioned provision explicitly does not include them either [10]. It thus differs from country to country whether the provision on grooming requires signatory states to extend an existing legal prohibition or to enact a new legal prohibition.

2.1.1 Definition and Scope of Cybercrime

Against the aforementioned background cybercrime can be defined as any new or different human act that is carried out through the use of computers or computer networks and is prohibited by the enactment of a new or the extension of an existing law. It differs from country to country which behaviors involving the use of computers or computer networks are outlawed. The Convention on Cybercrime, its Additional Protocol and the Convention on the Protection of Children against Sexual Exploitation and Sexual Abuse provide a list of new and different human acts involving the use of computers or computer networks that are commonly prohibited, i.e. illegal access, illegal interception, data interference, system interference, misuse of devices, computer-related forgery, computer-related fraud, offences related to child pornography, offences related to infringements of copyright and related rights, acts of a racist and xenophobic nature that are committed through computer systems and “grooming.”

2.2 Meaning of the Term “Virtual”

The adjective “virtual” has both a pre-computer, traditional meaning and a computer-based meaning [12]. The pre-computer, traditional meaning of the adjective “virtual” is twofold. Firstly, virtual in this sense can mean “quasi” or “pseudo” [11]. Secondly, virtual in this sense can mean “imaginary”, “make-believe” or “fake” [12].

There is no consensus on the computer-based meaning of the adjective “virtual.” There are countless definitions, each focusing on a particular context [11]. What the adjective “virtual” means precisely, seems to be dependent on its context. Below I will discuss the computer-based meaning of the term “virtual” in different contexts that will prove of importance for this paper.

In principle, the term “virtual” can refer to “anything that is created or carried by a computer and that mimics a “real” entity”, e.g. virtual memory [12]. Virtual memory is memory that is not actually built into the computer. It is a computer simulation of physical memory and can effectively function as such [12].

The term “virtual” can also be used in the specific context of a “virtual world”. A virtual world is an interactive, computer-simulated environment that is accessed by multiple users at the same time [11]. The first virtual worlds began to appear in the late 1970s. They were text-based online computer games known as MUDs (Multi-User Dungeons). MUD players created a fantasy world only using text. The next stage, graphical MUDs, started in the mid-1980s. They were image- rather than text-based fantasy worlds. In the twenty-first century graphical MUDs evolved into MMORPGs (massively multi-player online role-playing games). The increased internet access speed and the improved computer-processing power allowed for more complicated graphics, such as 3-D visuals. The vast majority of MMORPGs can still be described as fantasy worlds. But over the last decade a few virtual worlds have arisen that eschew the fantasy-based role-playing game play common to MMORPGs. They offer “an augmented version of reality” [4]. Such virtual worlds are called “metaverses” [4].

The users of virtual worlds represent themselves by means of an “avatar”. In graphical virtual worlds an avatar is a graphical object, which usually has a human-like form. In text-based virtual worlds it is a nick-name. Through their avatars users interact with each other and with virtual objects. Virtual objects are merely images that represent certain physical objects, e.g. cars.

Lastly, the term “virtual” can be used in the context of “virtual reality.” Virtual reality consists, just like a MMORPG, of an interactive, computer-simulated environment with 3-D visuals. But virtual reality differs from MMORPGs in two important aspects. First of all, users do not experience the three-dimensional, interactive, computer-simulated environment through an avatar, but through their own eyes and other senses. Secondly, virtual realities do not offer multi-access yet, at least not beyond a very limited degree, so users will mainly interact with objects instead of other users [11]. Virtual reality is designed to exploit the sensory systems of human beings so as to produce a sense of presence in those environments [13]. Virtual reality technology first emerged in the 1980s. It consists of a head-mounted display and a dataglove or datasuit attached to a computer. As the user navigates through and

interacts with the computer-simulated environment, the computer gives sensory feedback through the dataglove or datasuit [12]. Highly advanced datagloves can, for instance, make the user feel resistance when s/he grabs a computer-simulated object in the computer-simulated environment [11]. Virtual reality technologies are used to simulate both real and imaginary environments. In medicine, they are for instance used to simulate anatomical structures and medical procedures, for example for the training and education of surgeons [12].

In his dissertation, Søraker has done extensive research on the computer-based meaning of the term “virtual”. He comes to the conclusion that “computer simulation” and “interactivity” constitute the essence of the computer-based meaning of the term “virtual” [11]. Søraker provides the following generic definition of the term “virtual”: a virtual x is an “interactive, computer-simulated x (or, x , made possible by interactive computer simulation)” [11]. This definition focuses exclusively on virtual worlds and excludes from its scope things that are created or carried by a computer and mimic a real thing, such as virtual memory, because they are not interactive. Since these things should, for the purposes of this paper, be included in the scope of the definition of the term “virtual” I will make use of a generic definition of the term “virtual” that does not necessarily include interactivity. I will take “virtual” to mean computer-simulated or made possible by computer simulation. The computer simulation may or may not be interactive.

2.2.1 State of the Art: Virtual Cybercrime

Applying the above-mentioned definition of the term “virtual”, virtual cybercrime can be described as cybercrime that is carried out through the use of a specific feature of computers and computer networks, namely computer simulation. It is computer-simulated crime or crime, made possible by computer simulation. Virtual cybercrime thus consists of a computer-simulated human act or a human act made possible by computer simulation, that is prohibited by law.

The distinction between a computer-simulated human act and a human act made possible by computer simulation is an important one and should, therefore, be highlighted. A computer-simulated human act is an act that is virtual in itself. When someone performs a computer-simulated act, s/he acts in a virtual environment through an input device. An example of a computer-simulated human act is the shooting of a bear in the virtual environment of a computer game. Such a computer-simulated human act consists of three steps. First, a human being performs a bodily action, e.g. the pressing of a button. Second, the computer simulation interprets the bodily action as a particular command, e.g. “shoot the bear”. Third, the computer simulation makes the changes to the virtual environment (and possibly to the non-virtual world as well) that are required by the command, e.g. the bear in the virtual environment is death. A human act made possible by computer simulation is an act that is not virtual in itself, but that is defined in terms of a virtual object. Computer simulation is the condition of possibility for such an act and the nature of that act is partly determined by features of the computer simulation [11]. The production, possession or distribution of virtual child pornography is an example of a human act made possible by computer simulation. The aforementioned act is not virtual in itself,

but defined in terms of a virtual object: virtual child pornography. Virtual child pornographic images are child pornographic images which, although realistic, do not involve a child really engaged in sexually explicit conduct. They are either morphed pictures of real children or entirely computer-generated images [6]. Virtual child pornographic images are thus made possible by computer simulation. The nature of the act of producing, distributing and possessing them is partly determined by the features of the computer simulation, because it does not involve (the profiting from) child abuse, as opposed to the production, distribution and possession of non-virtual child pornographic images.

In fact, the production, possession or distribution of virtual child pornography is the only human act involving computer simulation that is commonly prohibited. The Convention on Cybercrime's prohibition on child pornography, as was discussed in section 1.1.1, includes the production, possession and distribution of virtual child pornography in its scope [6]. However, Iceland, Scotland and the United States of America have reserved the right not to apply the prohibition on virtual child pornography [14].

Dutch case law provides another example of a human act made possible by computer simulation that has been brought under the scope of penal law. In 2009 Dutch judges have convicted several minors of theft, because they had stolen virtual items in the virtual worlds of online multiplayer computer games. Three minors were convicted of theft for the stealing of virtual furniture in the virtual world of the online multiplayer computer game *Habbo* [15]. By means of deceit the perpetrators obtained the usernames and passwords of other *Habbo* players, so that they could access the other players' accounts and transfer their virtual furniture to their own *Habbo* accounts. In a similar case, two minors were convicted of theft for stealing a virtual amulet and a virtual mask in the virtual world of the online multiplayer computer game *RuneScape* [16]. The perpetrators had violently forced another player of *RuneScape* to give them access to his account, so that they could transfer his virtual amulet and virtual mask to their own *RuneScape* accounts. This judgement was confirmed by the Dutch Supreme Court [1]. The acts of stealing in these cases were not virtual in themselves, because they involved out-of-the-game infractions (deceit, violence). But they were defined in terms of virtual objects (the virtual items stolen). There have not yet been comparable penalties in other jurisdictions [1].²

Examples of computer-simulated crime are only found in the legal literature as opposed to in actual law [e.g. 4,5,17]. The most well-known example of a computer-simulated crime is the virtual "rape" that was described by Julian Dibbel in a much-debated 1993 paper. Dibbel describes how a user represented by an avatar named Mr. Bungle took control over other users' avatars in the virtual environment of *LambdaMOO* and forced their avatars, through his own avatar, to engage in sexual activities they did not consent to [18]. *LambdaMOO* was a text-based MOO-MUD: a MUD that mainly aimed at social interaction with other users [4]. There have not been penalties with regard to computer-simulated crime yet.

² See for a thorough analysis of this issue my paper [24].

Unlike the virtual worlds of computer games, virtual reality technologies have not yet been exploited for criminal activities, at least there have not yet been reported cases of crime instrumented by virtual reality technologies. That is because virtual realities do not yet offer multi-access or at least not beyond a very limited degree. Except for rare cases of “victimless” crimes, such as gambling or drunk-driving, crimes generally victimize another person. And thus virtual realities are not likely to provide new opportunities for crime until they become multi-accessible on a larger scale.

Finally, it is important to note that none of the virtual cybercrimes listed above concern new human activities; they are all different forms of traditional crimes. Virtual cybercrime consists either of a computer-simulated traditional crime or of a traditional crime that is defined in terms of a computer-simulated person or object. Therefore, it only requires legislators to extend existing laws and not to enact new ones.

2.2.2 Definition and Scope of Virtual Cybercrime

Against this background, virtual cybercrime can be defined as a computer-simulated human act or a human act made possible by computer simulation that is prohibited by the extension of an existing law. The scope of virtual cybercrime is unclear, however. Currently, the production, possession and distribution of virtual child pornography is the only virtual cybercrime that is commonly prohibited, although not as commonly as non-virtual child pornography. Putative virtual cybercrimes are, for example, virtual rape and theft of virtual items. This computer-simulated human act and human act made possible by computer simulation are not (commonly) prohibited yet. In the next section I will examine what the necessary and sufficient conditions are for a computer-simulated human act or a human act made possible by computer simulation to obtain in order to be prohibited under existing law so that I can ultimately determine the scope of the term “virtual cybercrime”.

3 Virtual Cybercrime: Necessary and Sufficient Conditions

It was established in the last section that the production, distribution and possession of virtual child pornography is the only virtual cybercrime that is commonly prohibited. Since it would be a fallacy to make a general statement about virtual cybercrime on the basis of one specific instance of virtual cybercrime, an empirical study of the law does not suffice to answer the question what the necessary and sufficient conditions are for a computer-simulated human act or a human act made possible by computer simulation to obtain in order to be prohibited under existing law. Therefore, I will study virtual cybercrime from a different point of view. As was stated in the introduction, the study of virtual cybercrime belongs to the field of legal ontology. Applied forms of ontology often put to use the tools of philosophical ontology in

order to categorize things within a specific domain. I will make use of this method and put to use the tools of the philosophical ontology of the American philosopher Searle in order to categorize virtual cybercrime within existing law. I choose to draw from Searle's work, because he provides the most influential recent social ontology, which is an ontology that does not focus on matters of biology and physics, but on matters of society, and pays special attention to the law. Next I will make use of legal philosophy to reflect on the outcome of the ontological analysis.

3.1 Ontological Analysis

Searle distinguishes a special class of facts: institutional facts. Institutional facts are facts that only exist by human agreement or acceptance. They come into being, because people or authorities impose status functions on things. A good example of an institutional fact is money. Money exists because we have imposed the status function of legal tender on pieces of paper and metal. Status functions are imposed by means of "constitutive rules" that have the following form: "X counts as Y (in context C)" [19]. An example of a constitutive rule is: the Euro (X) counts as a legal tender (Y) in certain EU-member countries, which are collectively known as the Eurozone (C). Penal provisions are also constitutive rules. They typically indicate that a certain human act (X) counts as a crime (Y) in a particular jurisdiction (C). Penal provisions are a special kind of constitutive rules, because they precisely specify the conditions under which the institutional fact (the crime) is created. They take the following form: for any x that satisfies a certain set of conditions p, x has status Y in C [19]. Consider, for example, the US penal prohibition on murder. This penal provision makes it the case that any act (x) that satisfies the conditions of unlawful killing of a human being with malice aforethought (p) counts as murder (Y) in the jurisdiction of the United States (C) [21].

In legal terms, the conditions that a human act needs to satisfy in order to count as a crime are called elements. The specific elements required vary depending on the crime, but there are two basic elements that are required by each crime: an *actus reus* (an unlawful act or failure to act) and a *mens rea* (a blameworthy mental state, usually it is required that the actor acts knowingly, purposely or recklessly).³ In fact, all crimes also require, implicitly or explicitly, that the *actus reus* must have a certain consequence, e.g. the death or injury of a person or a loss of property. This common element is called *causation*.

In the case of virtual cybercrime the basic elements of a crime can be satisfied "intravirtually" (within the virtual environment where the act takes place) or "extravirtually" (outside its virtual environment).⁴ The element of *actus reus* can be satisfied either intravirtually or extravirtually. A computer-simulated human act satisfies the element of *actus reus* intravirtually, because such an act is committed within a virtual environment through an input device. A human act made possible by

³ The terms "actus reus" and "mens rea" derive specifically from Anglo-American jurisprudence. But these elements are, although under a different name, also found in other legal systems.

⁴ The distinction between "intravirtual" and "extravirtual" derives from Søraker [22].

computer simulation satisfies the element of *actus reus* extravirtually, because such an act, although it is defined in terms of a virtual object, takes place outside the virtual environment. The element of *mens rea* can only be satisfied extravirtually, even when the element of *actus reus* is satisfied intravirtually. That is because the element of *mens rea* concerns the mental state of the human actor, who is necessarily extravirtual.⁵ This does not mean that, in the case of an intravirtual *actus reus*, the mental state of the actor is judged entirely independently from the virtual environment in which the act has taken place; for circumstances in the virtual environment can indicate whether s/he has acted knowingly, willingly or purposely. Like the element of *actus reus*, the element of causation can be satisfied either intravirtually or extravirtually. The element of causation is satisfied intravirtually when the *actus reus* has a consequence within the virtual environment and extravirtually when it has a consequence outside the virtual environment. It should be noted that where the element of causation is satisfied, within or outside the virtual environment, is not dependent on where the element of *actus reus* is satisfied: an intravirtual *actus reus* can have an extravirtual consequence and vice versa.

Where the element of causation is satisfied, intravirtually or extravirtually, is of crucial importance, because it determines the context (C) in which the crime status (Y) of a computer-simulated human act or human act made possible by computer simulation (X) holds. A computer-simulated human act or human act made possible by computer simulation (X) that satisfies the element of causation (p) *intravirtually* cannot count as a crime (Y) in the context of the non-virtual world (C), but may count as a crime (Y) in the context of its virtual environment (C). A computer-simulated human act or human act made possible by computer simulation (X) that satisfies the element of causation (p) *extravirtually* counts as a crime (Y) in the context of the non-virtual world (C).

The context (C) in which the crime status (Y) of a computer-simulated human act or human act made possible by computer simulation (X) holds, its virtual environment or the non-virtual world, determines whether or not the act can be included in the scope of an existing penal provision. Penal law does not apply within virtual environments. This is often explained in terms of a “magic circle”. In short, the magic circle is a metaphorical line which separates the virtual from the non-virtual realm and excludes penal law from virtual environments; regulation of conduct within these environments is left to the moderators or users.⁶ It is important to note that the rules that are set up by the moderator or users and govern the virtual environment may constitute the same crimes as we know in the non-virtual world, but they may also prohibit conduct that does not constitute a crime in the non-virtual world or allow for

⁵ In the future, the element of *mens rea* will not necessarily concern the mental state of a human actor anymore, since autonomous, learning machines, based on neural networks, genetic algorithms and agent architectures will be capable of having a *mens rea* of their own [23]. When such a machine will be part of a virtual (reality) environment, the element of *mens rea* can be satisfied intravirtually as well. Since this paper focuses on computer-simulated *human* acts and *human* acts made possible by computer simulation, the intravirtual *mens rea* is beyond its scope, however.

⁶ See for a thorough analysis of the “magic circle” my paper [24].

things that are prohibited in the non-virtual world. A computer-simulated human act or a human act made possible by computer simulation that only counts as a crime in its virtual environment thus triggers remedies within that virtual environment, but not penal law. A computer-simulated human act or human act made possible by computer simulation that counts as a crime in the non-virtual world crosses the metaphorical line of the magic circle and is, therefore, within the reach of penal law. Other authors [4,17, 20] have reached similar conclusions.

Consider the following example. Most countries prohibit various aspects of the production, trade and possession of certain drugs, because they can cause severe health problems to the people who use them. Within the virtual world of *SecondLife* users can produce, trade, possess and use a drug called “Seclimine” through their avatars [25]. The computer-simulated human act of producing, trading or possessing Seclimine in *SecondLife* satisfies the element of causation that is implicit in this actus reus intravirtually. After all, Seclimine can only be used through an avatar within the virtual world of *SecondLife* and can, therefore, not cause severe health problems to the person behind the avatar. Since the computer-simulated human act of producing, selling or possessing Seclimine within *SecondLife* (X) satisfies the element of causation (p) intravirtually, it cannot count as a crime (Y) in the context of the non-virtual world (C). If the rules of *SecondLife* prohibit the producing, selling or possessing of Seclimine, the act does count as a crime in the context of its virtual environment though.

Consider another example. Many countries legally restrict gambling. Gambling is illegal in these countries unless it complies with certain regulations made under law. In some countries, for example New Zealand, individual persons who participate in illegal gambling are held liable under criminal law. The actus reus of illegal gambling can be defined as the unlawful betting or wagering of money or something else of value. This actus reus implies a certain consequence: financial gain or loss. On the Internet one can gamble illegally in a virtual casino on a virtual slot machine with real, non-virtual money. The computer-simulated human act of illegal gambling on a virtual slot machine with real money satisfies the element of causation that is implicit in this actus reus extravirtually. After all, the money gained or lost is not virtual. Since the act of illegal gambling on a virtual slot machine with real money (X) satisfies the element of causation (p) extravirtually, it counts as a crime (Y) in the context of the non-virtual world (C) and can thus be brought under the scope of the penal prohibition on illegal gambling that some countries apply.

Sometimes a computer-simulated human act or human act made possible by computer simulation (X) can satisfy the actus reus element and the attendant element of causation of one crime intravirtually and, thereby, satisfy the actus reus element and the attendant element of causation of another crime extravirtually. Such an act counts, therefore, as crime Y in the context of its virtual environment (C) and as crime Z in the context of the non-virtual world (C). Consider the following example. Several media reported the case of a 43-year-old Japanese woman who “killed” the avatar her own avatar was married to in the virtual world of the online multiplayer computer game *MapleStory*, because it had suddenly divorced her avatar. The woman had hacked into the account of the person behind her virtual husband and deleted his

avatar. When the person found out, he called the police. The police investigated the case and even arrested the woman at her home, but she was never formally charged [2]. Provided that the deleting of an avatar is indeed considered manslaughter in the virtual environment of *MapleStory*, the act of the Japanese woman satisfies both the actus reus element (killing) and the element of causation (the death of the avatar) of that crime intravirtually. After all, both the act of killing and the death of the avatar occur within the virtual environment of *MapleStory*. But the death of the avatar in *MapleStory* also has a consequence in the non-virtual world; for the user who was represented by the avatar has lost his virtual alter ego. As was explained in section 1.1.1 countries also commonly prohibit the deterioration of computer data without right (article 4 Convention on Cybercrime). Since an avatar consists of computer data, we could say that the killing of the avatar equals the deterioration of (a set of) computer data. And since the woman illegally accessed the account of the user the avatar represented, it is also without right.⁷ By satisfying the elements of the crime of manslaughter intravirtually, the Japanese woman who killed another user's avatar in *MapleStory* thus satisfies the elements of the crime of deterioration of computer data extravirtually. In sum, the computer-simulated human act of killing an avatar (X), which counts as manslaughter (Y) in the context of its virtual environment (C), counts as deterioration of computer data (Z) in the context of the non-virtual world (C).⁸

In conclusion, a computer-simulated human act or human act made possible by computer simulation that satisfies the elements of a crime can only be brought under the scope of an existing penal provision if it counts as a crime in the non-virtual world. It does if it satisfies the element of causation extravirtually. So, in order to be brought under the scope of existing penal law, it is a *necessary* condition for a computer-simulated human act or human act made possible by computer simulation that satisfies the elements of a crime that it satisfies the element of causation extravirtually. But is that also a sufficient condition? Or are there other conditions to be met? As will be explained below, the answer to these questions depends on the stand one takes in the legal philosophical debate between legal positivists and natural law theorists.

3.2 The Debate between Legal Positivists and Natural Law Theorists

In legal philosophy there are two main, rival, theories about the content of the law: legal positivism and natural law theory. Legal positivists, like Austin, claim that laws may have any content. They would thus say that legislators and judiciaries are free to bring any computer-simulated human act or human act made possible by computer simulation that has an extravirtual consequence and satisfies the (other) elements of a crime under the scope of penal law. By contrast, natural law theorists think that the

⁷ It should probably be added that this already constitutes a crime in itself and that the woman could, therefore, also be held liable for illegal access ("hacking").

⁸ The distinction among the three above-mentioned types of virtual human acts and the different contexts in which their status function holds, derives from Brey [26].

content of laws is determined by their relation to morality. Classical natural law, which was originally developed by ancient philosophers such as Plato and Cicero and further elaborated by Thomas Aquinas, maintains that there is a necessary connection between law and morality and that an immoral law is no law. Typically, there is a particular theory of morality conjoined with that view: that the moral order is part of the natural order and that something is morally right if it is consistent with a natural purpose or end, such as survival [27]. Natural law theorists would say that legislators and judiciaries can only bring a computer-simulated human act or human act made possible by computer simulation that has an extravirtual consequence under the scope of penal law if the extravirtual consequence consists of a violation of a moral principle.

The contemporary debate on the content of the law is dominated by the legal philosophers Hart [28] and Dworkin [29] and interpretations of their work. Their theories have developed such a level of subtlety and sophistication that the traditional labels of legal positivism and natural law theory hardly apply anymore, however [27]. For the purposes of this paper only the common ground between Hart's and Dworkin's theory of law is of importance. Both Hart and Dworkin agree that the law is open to arguments that are grounded in moral principles. Taking this assumption as a starting point, Van der Burg argues that the law is most strongly open to moral argument with regard to special fields or issues that are still developing, such as biotechnology or ICT [30]. This claim can be explained as follows. As was discussed in the section 1.1, developing fields or issues such as biotechnology or ICT give rise to new and different forms of human activity that evade the reach of existing penal law, such as virtual cybercrime. It is not always clear how penal law should deal with them and this uncertainty is exhibited in the case of virtual cybercrime. Moral principles can be used to understand, analyze and evaluate arguments about how the penal law should deal with these new and different forms of human activity [30]. Yet the question arises which moral principles can help to determine how the penal law should deal with virtual cybercrime. Answering this question will be the aim of the next subsection.

3.2.1 Which Moral Principles Can Help to Determine How the Penal Law Should Deal with Virtual Cybercrime?

The general question of what moral principles are of importance to determine which human conduct should be criminalized and which not is extensively treated in Feinberg's voluminous work *The Moral Limits of the Criminal Law*, which consists of four separate books. Feinberg points out that when legislators or judiciaries bring a certain human act under the scope of a penal provision, citizens are no longer "at liberty" to perform that act [31]. According to Feinberg such an interference with the liberty of citizens by means of penal law is usually legitimated on the basis of one of the following liberty-limiting principles: the harm principle, the offense principle, legal paternalism or legal moralism [33]. I will discuss each of these liberty-limiting principles below.

The first liberty-limiting principle, the harm principle, originally derives from Mill. The harm principle entails "that the only purpose for which power can be rightfully

exercised over any member of a civilised community, against his will, is to prevent harm to others” [34]. For reasons of clarity it needs to be emphasized that Feinberg, contrary to Mill, does not believe that the harm principle is the *only* valid principle for legal coercion: after all he thinks that there are also other liberty-limiting principles [31]. Clearly, the harm principle crucially depends on what is understood by harm [35]. Mill never explicitly defined harm, but Feinberg has done so. He distinguishes between harm in a non-normative sense, which he defines as a setback to interest, and harm in a normative sense, which he defines as a wrong, that is a violation of rights caused by morally indefensible conduct [31]. Conduct is morally indefensible if it cannot be justified or excused, e.g. because the victim him- or herself voluntarily consented to a setback of his or her own interests. Feinberg claims that only setbacks to interests that are wrongs, and wrongs that are setbacks to interests can count as harms for the purposes of the harm principle. He thus defines harm, for the purposes of the harm principle, as a wrongful setback to an interest. One's interests, or more accurately, the things these interests are in, are components of one's well-being. The interests that form the basic requisites of one's well-being are called “welfare interests” and they are protected by law [31]. Welfare interests include: the interest in the continuance of one's life for a foreseeable interval, the interest in bodily integrity and the interest in the security of property. Examples of penal provisions that protect the aforementioned welfare interests are, respectively: prohibitions on murder, prohibitions on rape and prohibitions on theft. At last it should be added that harms can not only be suffered by an individual person, but also by society as a whole. Harms that are suffered by society as a whole consist of wrongful setbacks to “public” interests, such as the interest in political and economic stability or the interest in a clean environment [31]. Examples of penal provisions that protect the aforementioned public interests are, respectively: the prohibition on treason, the prohibition on counterfeiting and antipollution ordinances [31, 7].

The second liberty-limiting principle, the offense principle, is not concerned with (private or public) harm, but with offense. Like harm, offense can be defined both in a non-normative and a normative sense. The former includes in its reference all kinds of disliked mental states, such as disgust, shame, embarrassment and fear. The latter refers to those states when caused by the wrongful conduct of others. Only offense in this latter sense is intended in the offense principle. Offensive conduct of others is wrongful if it deprives “the unwilling spectators of the power to determine for themselves whether or not to undergo a certain experience”, which is a violation of the right to privacy in the sense of autonomy [33]. The offense principle should not be invoked too easily. Legislators or judiciaries who want to prohibit wrongful offensive conduct have to balance the seriousness of the offense caused (e.g. its intensity and duration) against the independent reasonableness of the offender's conduct (e.g. if wrongful offensive conduct is performed at a location where it is common and known to be common, it is less unreasonable than it would be at a location where it is rare and unexpected) [33]. Examples of penal provisions that are based on the offensive principle are: prohibitions on open lewdness, indecent exposure, solicitation and the distribution or sale of pornography [31].

The third liberty-limiting principle, legal paternalism, is concerned with harm again, like the first liberty-limiting principle: the harm principle. Contrary to the harm principle, legal paternalism is not concerned with harm to *others*, but with harm to the *self*. Legal paternalism entails that it is a good and relevant reason in support of a penal prohibition that it prevents harm to the actor him- or herself [36]. The interference with a person's liberty is justified by reasons referring exclusively to the welfare interests of the person coerced [38]. According to Feinberg there are two types of paternalism: hard (presumptively blamable) paternalism and soft (presumptively nonblamable) paternalism. Hard paternalism justifies interference with entirely voluntary self-regarding harmful behavior of people for their own good. Soft paternalism "consists of defending relatively helpless or vulnerable people from external dangers, including harm from *other* people when the protected parties have not voluntarily consented to the risk (...)" [36]. A person's self-regarding harmful behavior is substantially nonvoluntary when the choice to perform it stems from coercion, drugs or other voluntariness-vitiating factors and is, therefore, alien to him or her as the choices of someone else. Feinberg thinks that the latter type of paternalism is actually no kind of paternalism at all, because it authorizes the restraint of behavior that threatens a person with harm that, although it does not come from another person, is equally "other" from him- or herself [36]. Feinberg, therefore, focuses on hard paternalism. Examples of penal provisions that are based on legal paternalism are: prohibitions on the possession and use of psychoactive drugs and gambling as well as requirements, enforced by criminal sanctions, such as that motorcyclists wear crash helmets and that motorists use seat belts [31]. Most of these penal provisions can, however, not only be defended on the ground that the actors themselves need to be protected from the harmful consequences of their own acts (legal paternalism), but also on the ground that social harm needs to be prevented generally (the harm principle). That is because there is always a public interest involved, at least to a small extent, when people harm themselves. Think, for instance, of tax money spent on healthcare costs [36].

The last liberty-limiting principle, legal moralism, is not concerned with harm or offense, but with evils of other kinds. According to Feinberg there are two types of legal moralism: pure and impure moralism. Pure moralism entails that "it can be morally legitimate (...) to prohibit conduct on the ground that it is inherently immoral, even though it causes neither harm nor offense to the actor or to others" [39]. Impure moralism refers to the approach of some writers in legal philosophy who are called legal moralists, although the basic appeal in their arguments is to the harm or offense principle [39]. Of them Lord Devlin is the best known. Lord Devlin claims that human conduct is sometimes prohibited solely because society finds it immoral. He argues that it is legitimate for society to legislate against immorality, because society is kept together by the invisible bonds of a common morality, and would fall apart if these bonds were not protected [40]. Devlin thus thinks that immoral behaviour harms the social cohesion in society and, thereby, appeals to the harm principle. Examples of penal provisions that are based on legal moralism are: prohibitions on prostitution and bigamy [31].

No writer in legal philosophy denies the validity of the harm principle as a good and relevant reason in support of a penal provision. Most writers acknowledge the offense principle as well. But legal paternalism and legal moralism are contested [31]. Feinberg himself thinks that “harm and offense prevention are far and away the best reasons that can be produced in support of criminal prohibitions, and the only ones that frequently outweigh the case for liberty. (...) The other principles state considerations that are at most sometimes (but rarely) good reasons (...)” [39].

From an empirical point of view, it can be established that the harm principle is the most commonly and the most frequently used ground for criminalization. Although there are differences across countries and societies in how criminal behaviors are viewed and treated, the core of the criminal law, across geography and across time, consists of crimes that produce direct and serious harm to individual persons or groups. The criminal law contains everywhere and at any time penal provisions defining crimes against persons, such as murder, assault, rape and battery. Almost as non-controversial as these crimes against persons are various crimes against property, such as theft, arson and fraud [4].⁹ Penal provisions that are based on the offense principle, legal paternalism or legal moralism deviate across geography and across time.

In conclusion, the following moral principles can help to determine how the penal law should deal with virtual cybercrime: the harm principle, the offense principle, legal paternalism and legal moralism. In the last section it was established that it is a necessary condition for a computer-simulated human act or a human act made possible by computer simulation that satisfies the elements of a crime that it has an extravirtual consequence if it is to be brought under the scope of a penal provision. We can now establish that that is also a sufficient condition if the extravirtual consequence consists of harm (to another or to the self), offense or an evil of another kind. Yet the question arises when computer-simulated human acts or human acts made possible by computer simulation result in harm, offense or evils of other kinds. Answering this question will be the aim of the next section.

4 When Do Computer-Simulated Human Acts or Human Acts Made Possible by Computer Simulation Result in Extravirtual Harm, Offense or Evils of Other Kinds?

In this section I will take a so-called top-down approach¹⁰: I will apply the harm principle, the offense principle, legal paternalism and legal moralism to particular examples of computer-simulated human acts or human acts made possible by computer simulation that fall under these principles. That way I show when computer-

⁹ It should be added that the criminal law starts to focus less on harm and more on risk, however. This trend is currently merely visible in the periphery of criminal law. In the Netherlands, for example, local laws have been enacted to ban youths from the places where they hang around in order to prevent vandalism. If this trend continues, it will sooner or later also affect the core of the criminal law and make harm a less important ground for criminalization [32].

¹⁰ Beauchamp [41] describes the top-down approach as one of the models of moral reasoning in applied ethics.

simulated human acts or human acts made possible by computer simulation result in extravirtual harm (to others or to the self), offense or evils of other kinds.

4.1 Can Computer-Simulated Human Acts or Human Acts, Made Possible by Computer-Simulation Result in Extravirtual Harm to Others?

As was mentioned in the last section, Feinberg defines harm, for the purposes of the harm principle, as a wrongful setback to a (welfare) interest. This section will aim to answer the question when a computer-simulated human act or a human act made possible by computer simulation causes a wrongful setback to a welfare interest. Before answering this question, it is important to point at two supplementary principles that guide the application of the harm principle in practical contexts, however.

First, the harm principle makes sure that the criminal law does not concern itself with trivia. The harm principle can only be invoked if enough well-being is under threat [31]. But how great must the infliction upon a welfare interest be in order for the harm principle to warrant the criminal law to prevent it? According to Holtug, the harm principle involves a sliding threshold, such that the quantity of well-being that is under threat varies proportionally with the severity of the coercion in question. For example, there must be more well-being under threat to legitimate a prison sentence than a small fine [35]. If the amount of well-being that is under threat is so minor it cannot even legitimate the imposition of a small fine, the harm principle cannot be invoked at all.

Second, and this supplementary principle is closely connected to the first, the application of the harm principle requires a conception of normalcy. "It is the person of normal vulnerability whose interests are to be protected by coercive power; the person who, figuratively speaking, can be blown over by a sneeze cannot demand that other people's vigorous but normally harmless activities be suspended by government power" [31]. But what is a person of normal vulnerability? Since people and their situations differ, the amount of their well-being that is affected by a certain harmful act can vary. This problem is of crucial importance with regard to interactions in the virtual realm, because one generally does not know who the other person behind the screen is and, therefore, it is even more difficult than in the non-virtual world to estimate to which degree a certain harmful act affects the well-being of the other person.

The criminal law solves the above-mentioned problem by positing a "standard person" who is to be protected from "standard forms of harm" to "standard [welfare] interests" [31]. It was established in the last section that the core of the criminal law protects interests of personality and interests of property. According to Feinberg standard interests of personality include absence of harmful bodily contacts or the apprehension thereof, freedom from confinement and absence of emotional distress. Interests of property include the exclusive enjoyment and possession of land, chattels and other material resources and their good physical condition. Other legally protectable interests are: interests in privacy and interests in reputation. Not all countries protect the latter interests by means of the criminal law, however, some

protect them instead by compelling compensation for harm to them under civil law. Finally, as was mentioned earlier, the criminal law often does not only protect individual interests, but also public interests, such as the interest in a clean environment and the interest in economic and political stability [31].

Standard inflictions upon interests of personality consist of harm to a person's bodily health through e.g. murder or assault; harm to a person's mental health through e.g. harassment; diminutions of a person's security by the creation of threats or dangers and reductions of a person's liberty of movement through abduction or false imprisonment. Standard inflictions upon interests of property consist of depletion of a person's material resources through e.g. theft, arson or fraud. Standard inflictions upon interests in privacy consist of intrusions upon solitude e.g. through "stalking" or unpermitted disclosure of intimacies e.g. through unlawful filming. It should be added that the precise definition of "stalking" differs from country to country, but in general terms it can be described as unwanted, repeated intrusions (e.g. surveillance) and communications (e.g. phone calls, letters, gifts) that are inflicted upon a victim. Standard inflictions upon interests in reputation consist of false statements of fact about a person made in public (defamation). Defamation encompasses both libel and slander: libel refers to written statements or visual depictions, slander refers to verbal statements and gestures. Finally, standard inflictions upon public interests, such as the interest in a clean environment and the interest in economic and political stability consist of, respectively, environmental crimes (e.g. pollution); certain economic crimes (e.g. counterfeiting and smuggling) and crimes against the state (e.g. treason, rioting and obstruction of justice) [31,7]. Below it will be examined which of these standard forms of harm to standard welfare interests can be caused by computer-simulated human acts or human acts made possible by computer simulation.

Although it seems improbable at first sight, a computer-simulated human act or a human act made possible by computer simulation may result in harm to a person's bodily health. Consider the following example. In 2008 hackers intruded into the nonprofit Epilepsy Foundation's website and posted a message with a legitimate sounding-title. Users who clicked on the post were redirected to a page with a computer-generated animation that consisted of a pattern of squares rapidly flashing in different colors, which was designed to trigger seizures in both photosensitive and pattern-sensitive epileptics. Several epilepsy patients were affected [42]. This was possibly the first assault made possible by computer simulation and, to my knowledge, the only one. A computer-simulated human act could do the same type of harm if a user of a virtual environment, e.g. *SecondLife* or *MSN Messenger*, would, by the press of a button, make such a computer-generated animation designed to trigger seizures appear on the screen of another user, being a photo- and pattern-sensitive epileptic.

Much more often than harm to the bodily health of a person, computer-simulated human acts do harm to the "bodily health"¹¹ of a person's avatar. For example, a person can use his or her avatar to kill, assault, rape or torture another person's avatar. This results in (intravirtual) harm to the bodily health of the avatar, but does not do

¹¹ The term bodily health is used as a metaphor here. The bodily health of an avatar cannot literally be harmed, because an avatar does not have a physical body. But an avatar has a virtual body that can be virtually harmed within the virtual environment.

(extravirtual) harm to the bodily health of the person him- or herself. Several authors [43,44,45] argue that the computer-simulated human act of harming the bodily health of an avatar may not do harm to the bodily health of the person behind it, but can result in harm to that person's mental health. When a person is emotionally engaged in the virtual environment, because s/he is attached to and identifies with his or her avatar, bodily harm done to the avatar is felt as mental harm to the person [45]. A person whose avatar is raped, for example, can feel sexually harassed. Note that this is one of the special cases as were discussed in section 2.1.1 where a computer-simulated human act (X) satisfies the elements of one crime intravirtually and, thereby, satisfies the elements of another crime extravirtually and, therefore, counts as crime Y in the context of its virtual environment (C) and as crime Z in the context of the non-virtual world (C).

It should be added that a computer-simulated human act causing harm to a person's mental health is not necessarily aimed at the bodily health of that person's avatar; it can also be of a different nature. Consider the following example. When Ailin Graef, the woman who became a millionaire by investing in virtual real estate in *SecondLife*, appeared through her avatar on a chat show in the virtual world of *SecondLife* to talk about her success, the event was sabotaged by a group of other users. For fifteen minutes, Graef's avatar was swarmed by flying pink penises and photographs of Graef herself that were digitally altered to make her look like she was holding a giant penis. Graef felt sexually harassed [46]. It is important to note that, in this case, the sexual harassment within the virtual world of *SecondLife* spilled into the non-virtual world, because the identity of the person behind the avatar was known to the perpetrators. The harassment was not aimed at Graef's avatar (intravirtual), but at Graef herself (extravirtual). This became especially clear, because a photograph of Graef was used.

Mental harm to persons is not only done by computer-simulated human acts, but also by human acts made possible by computer simulation. For example, many virtual worlds (e.g. *SecondLife* and *World of Warcraft*) provide a chat interface, which users can abuse to send harassing messages to other users through their avatars. It should be added that harassment cannot only cause harm to the mental health of victims, it can also cause a diminution of the victim's security, if the harassment consists of threats. It is important to highlight that the harassment should be aimed at the user of the virtual world, not at the user's avatar. As became clear earlier, this can only be the case when the identity of the person behind the avatar is known to the perpetrator(s). It may be that the person behind the avatar has revealed his or her own identity, for instance in a chat conversation. It may also be that the perpetrator has unlawfully accessed the personal details of the person behind the avatar, e.g. by means of hacking.

It seems implausible that a computer-simulated human act or a human act made possible by computer simulation can cause extravirtual reductions of a person's liberty of movement through abduction or false imprisonment, at least it is not easy to think of an example. But a computer-simulated human act or a human act made possible by computer simulation can definitely cause a depletion of a person's material resources through larceny. I have extensively discussed this issue in another paper

[41]. In short, if virtual property is purchased with funds having extravirtual value (value in the non-virtual world, e.g. pecuniary value), then the extortion thereof constitutes extravirtual harm.

Computer-simulated human acts or human acts made possible by computer simulation can raise privacy issues as well. One can, for example, make one's avatar stalk another person's avatar in a virtual world. This is a computer-simulated human act. One can also think of unauthorized filming within a virtual world. In *SecondLife*, for example, it is possible to film. Films made in *SecondLife* are often put on *YouTube*. Yet one could film the private moments of an avatar, for example of the avatar having sex, put the film on *YouTube* without permission and, thereby, unpermittedly disclose the avatar's intimacies. This is a human act made possible by computer simulation. Just like with harassment, stalking or unauthorized filming in the virtual world can spill into the non-virtual world when the perpetrator knows who the person behind the avatar is. It is questionable though whether there is enough well-being under threat here to invoke the harm principle.

Computer simulation also offers new possibilities for defamation. Consider the following example. In 2010 a Dutch man was convicted for libel because he had put a digitally altered image of the then Prime Minister Balkenende online that depicted him, among other things, with a Hitler moustache and swastika's [47]. One can also think of the defamation of avatars here, for example by means of a written statement on an Internet forum. Contrary to harassment, stalking or unauthorized filming of an avatar, defamation of an avatar cannot only take effect in the non-virtual world when other users know who the person behind the avatar is. Some people make money through their avatars, think for example of the earlier mentioned case of Ailin Graef, who became a millionaire by investing in virtual real estate in *SecondLife* through her avatar Anshe Chung. If someone would make a false statement of fact about Anshe Chung, for example that she is involved in virtual real estate fraud, and because of that no one would be willing to do business with her anymore, Ailin Graef, the woman behind Anshe Chung, would suffer a non-virtual financial loss.

Finally, computer-simulated human acts or human acts made possible by computer simulation can intrude upon public interests. Counterfeiting, for example, can be made possible by computer simulation, for people can use graphics software to create false bank notes.

4.2 Can Computer-Simulated Human Acts or Human Acts, Made Possible by Computer-Simulation Result in Extravirtual Offense?

In the last section it was established that Feinberg defines offense as a disliked mental state caused by the wrongful conduct of others. Offensive conduct of others is wrongful if it deprives "the unwilling spectators of the power to determine for themselves whether or not to undergo a certain experience" [33]. According to Feinberg examples of penal provisions that are based on the offense principle are: prohibitions on open lewdness, indecent exposure, solicitation, activities and materials offensive to religious or patriotic sensibilities (e.g. blasphemous materials), racial and ethnic slurs and the distribution or sale of pornography [31]. Weckert, who has done extensive research on offense on the

internet, divides the aforementioned offensive behaviors into three categories. The first category concerns things that are not necessarily directed at any person or group. This category includes indecent exposure and solicitation. It actually also includes the sale and distribution of pornography, but Weckert has excluded pornography from his categorization, because it raises issues of its own [48]. The second category concerns the ridiculing or criticizing of beliefs and commitments. This category includes activities and materials offensive to religious or patriotic sensibilities. The last category concerns offense taken at language that is racist or sexist or denigrates people with mental or physical disabilities or the victims of accidents or crimes. This category includes racial and ethnic slurs. It may also include open lewdness insofar as the lewdness denigrates people with mental or physical disabilities or the victims of accidents or crimes [48].

Weckert claims that only the last category of offensive behaviors should be restricted on the Internet. This claim can be explained as follows. As was mentioned in the last section, Feinberg thinks that we have to balance the seriousness of the offense caused (e.g. its intensity and duration) against the independent reasonableness (avoidability) of the offender's conduct when we invoke the offense principle. Weckert points out that most offenses on the Internet can easily be avoided. If one is offended by the content of a certain website, e.g. because it contains materials that one considers blasphemy, one can simply choose not to visit that website. This would be different if one was confronted with the offensive material every time one logged on to the Internet, say by a particular welcoming message or the wording of an image or icon. And it would definitely be different if one was confronted with the offensive material on the road one has to pass on one's way to work, e.g. on a billboard. Given the high degree of avoidability of offense on the Internet, only very serious offenses can tip the scales so that the offense principle can be invoked. As Weckert explains, only offenses from the third category are serious enough to do that. They are, contrary to offenses from the first category, aimed directly at (a group of) persons. They also differ from offenses from the second category, since they offend because of characteristics over which people do not have control, such as race, gender and physical appearance, where offenses from the first category offend because of characteristics over which people have at least some control, such as political and religious beliefs. Offenses from the third category are thus the most serious types of offenses because they single out individuals or groups by characteristics which they have no power to change and, therefore, there is reason to restrict them on the Internet [48].¹²

Weckert's argument does not only make sense with regard to human acts involving the use of the Internet in general, it also applies to computer-simulated human acts and human acts, made possible by computer-simulation specifically. The degree of avoidability with regard to computer-simulated human acts or human acts made

¹² If the "unwilling spectator" is a child, there might also be reason to restrict indecent exposure, which belongs to the first category of offenses, on the Internet. That is because for children the degree of avoidability of such an offense is lower, especially when an adult persuades them to watch [37]. One could argue, however, that indecent exposure of an adult to a child does not constitute offense, but mental harm and that thus the harm principle instead of the offense principle should be invoked.

possible by computer simulation is high, because one has the choice not to participate in a certain virtual world known for its offensiveness. Of course, this argument is the strongest with regard to virtual worlds with a pre-designed content. In virtual worlds where users themselves shape the virtual world, such as *SecondLife*, it might be problematic for new users to know whether or not they will find (an area of) the virtual world offensive. But ultimately, one can always turn off the computer. So, here also only offenses from the third category are serious enough to tip the scales and invoke the offense principle. Such offenses, i.e. racial or ethnic slurs and open lewdness insofar as it denigrates people with mental or physical disabilities or the victims of accidents or crimes, are most likely to consist of comments, suggestions, requests, proposals or other communications in an environment made possible by computer simulation, e.g. a computer game with chat function. But they can also consist of computer simulated images [48]. In the United Kingdom, for instance, a man was sentenced to 300 hours of community service, because he had posted an offensive digitally altered image of a teenage shooting victim on Facebook [50]. The aforementioned acts are all human acts made possible by computer simulation. Computer-simulated human acts can produce offenses from the third category as well. Think, for instance, of a person who makes his or her avatar do the Nazi salute when it meets a black avatar in a virtual world. No matter whether the person behind the avatar is black him- or herself, he or she can take offense.

It becomes clear here that offense in the virtual realm differs in one important aspect from harm in the virtual realm: contrary to harm, we cannot distinguish between intra- and extravirtual offense. In section 2.2.1 harm was defined as a wrongful setback to an interest. As was established in section 3.1, a wrongful setback to an interest can be either intra- or extravirtual. Sometimes, an intravirtual wrongful setback to one interest counts as an extravirtual wrongful setback to another interest. As was mentioned above, offense can be defined as a disliked mental state, caused by the wrongful conduct of others. A disliked mental state can only be extravirtual, because it concerns a human being and human beings are necessarily extravirtual. An extravirtual disliked mental state can be caused either by intra- or extravirtual wrongful conduct of others, but that does not make a difference for the disliked mental state: one can be as offended by seeing an avatar doing the Nazi salute in the virtual world of a computer game (intravirtual wrongful conduct) as by being shown an offensive (digitally altered) image in the non-virtual world (extravirtual wrongful conduct).

4.3 Can Computer-Simulated Human Acts or Human Acts, Made Possible by Computer-Simulation Result in Extravirtual Harm to the Self ?

As was established in the last section, the criminal law does not only outlaw behaviors that harm others, but also behaviors that harm the *self*. Penal provisions that prohibit behaviors that inflict harm upon the self are called paternalistic. There are two kinds of paternalistic penal provisions: provisions that *prohibit* certain kinds of behavior, such as the use of psychoactive drugs and gambling, and provisions that *require* certain kinds of behavior, enforced by criminal sanctions, such as that motorcyclists

wear crash helmets and that motorists use seat belts [31]. Most of these penal provisions can, however, also be defended on the ground that social harm needs to be prevented generally, because there is always a public interest involved, at least to a small extent, when people harm themselves, e.g. the tax money spent on healthcare costs [36].

In section 3.1 we distinguished different types of harm, i.e. harm to a person's bodily or mental health; diminutions of a person's security by the creation of threats or dangers; reductions of a person's liberty of movement through abduction or false imprisonment; depletion of a person's material resources; violations of a person's privacy; defamation and inflictions upon public interests, such as the interest in a clean environment and the interest in economic and political stability. Not all of these types of harm can be inflicted upon the self. Public harms are singled out by definition. It also seems implausible that a person reduces his or her own liberty of movement through abduction or false imprisonment or that a person violates his or her own privacy. Yet the question arises which harms inflicted upon the self can constitute crimes. As will be explained below, Dworkin provides an answer to this question.

In his influential 1972 article on paternalism [38], Gerald Dworkin lists the following eleven examples of paternalistic interferences by law:

1. "Laws requiring motorcyclists to wear safety helmets when operating their machines.
2. Laws forbidding persons from swimming at a public beach when lifeguards are not on duty.
3. Laws making suicide a criminal offense.
4. Laws making it illegal for women and children to work at certain types of jobs.
5. Laws regulating certain kinds of sexual conduct, e.g. homosexuality among consenting adults in private.
6. Laws regulating the use of certain drugs which may have harmful consequences to the user but do not lead to anti-social conduct.
7. Laws requiring a license to engage in certain professions with those not receiving a license subject to fine or jail sentence if they do engage in the practice.
8. Laws compelling people to spend a specified fraction of their income on the purchase of retirement annuities. (Social Security)
9. Laws forbidding various forms of gambling (often justified on the grounds that the poor are more likely to throw away their money on such activities than the rich who can afford to).
10. Laws regulating the maximum rates of interest for loans.
11. Laws against dueling."

Not all of these examples concern the criminal law. The fourth, eighth and tenth example concern laws that are generally not part of the criminal law. With regard to the fifth example, it should be added that most countries have repealed their laws against homosexuality. The other examples all concern penal provisions that protect people from harm to their bodily health inflicted by themselves, except for laws

forbidding various forms of gambling, which protect people from depletion of material resources inflicted by themselves.

As the seventh example shows, the class of people whose welfare interests are protected does not need to be identical with the class of people being coerced. In the case of professional licensing it is the practitioner's freedom which is directly interfered with and it is the would-be patient or client whose welfare interests are presumably being served. This can be called "impure paternalism" [38]. It might be thought that it is superfluous to distinguish impure paternalism, because any such case could be brought under the scope of the harm principle. The difference between instances of impure paternalism and instances of harm to others is, however, that in the former but not in the latter cases the harm is of such a nature that it could be avoided by the individuals affected if they so choose. In the case of professional licensing, the practitioner is coerced so that the would-be patient or client cannot choose to be treated by an unlicensed practitioner, which might cause (bodily)harm.

I will now establish which of the paternalistic laws that Dworkin mentions are applicable to computer-simulated human acts or human acts, made possible by computer simulation. One can think of a computer-simulated equivalent of most of the (potentially) self-harming prohibited human activities mentioned above. One can, for example, make an avatar drive a motorcycle without a safety helmet, swim at an unguarded beach or commit suicide. And as was mentioned in section 2.1.1 people can use a drug called "Seclimine" through their avatars within the virtual world of *SecondLife*. Also, many multiplayer computer games, e.g. *World of Warcraft*, allow players to duel against each other through their avatars. But the aforementioned activities only endanger the (intravirtual) bodily health of the avatar; they do not endanger the (extravirtual) bodily health of the person behind it and can, therefore, not be brought under the scope of the paternalistic laws prohibiting these non-virtual equivalents. The only computer-simulated human act that can actually cause extravirtual harm to the self and can thus be brought under the scope of a paternalistic law is the act of gambling on a virtual slot machine. As was already discussed in section 2.1.1, the computer-simulated human act of gambling on a virtual slot machine can be brought under the scope of the prohibition on gambling because it involves real, non-virtual money and can thus cause a depletion of a person's material resources in the non-virtual world.

I can also think of an example of a human act made possible by computer simulation that can cause extravirtual bodily harm to the self and can, therefore, be brought under the scope of one of the paternalistic laws as distinguished by Dworkin. Unlicensed practice of medicine can be made possible by the Internet and, as will be explained later, also by computer simulation. People make use of the Internet as a source of health information and sometimes engage in what has been called "do-it-yourself-healthcare" [51]. Medical research shows that this can have harmful consequences [52]. That is because it is difficult to control the reliability of health information on the Internet, since there is no system of licensing or another form of authorization available online [51]. So far, one fatal case of do-it-yourself-healthcare by the use of health information on the Internet has been reported. A 55-year-old man with cancer found information on the Internet that promoted the use of a certain

medicine for cancer treatment. After self-medicating for four months with the medicine, which he had obtained from an alternative medicine website, he died. Autopsy findings suggested an adverse reaction from the use of the medicine [52]. In the metaverse of *SecondLife* one can find several virtual hospitals. In some of them users can also consult a virtual doctor through their avatars. Here, the reliability problem arises as well. After all, it is difficult to establish whether or not the person behind the virtual doctor is a licensed doctor. Thus, if a user of *SecondLife* takes a medical advice from a virtual doctor, this can be as dangerous for his or her health as relying on health information on the Internet. Therefore, the paternalistic law prohibiting unlicensed practice of medicine is in principle applicable.

The above-mentioned example of extravirtual harm to the self made possible by computer-simulation might be a little far-fetched. After all, it is about a non-virtual human being in the non-virtual world who takes a medical advice that is obtained in a virtual environment. A much clearer example of extravirtual harm to the self made possible by computer simulation would be computer and video game addiction, which seems to be a growing problem and is associated with a range of mental and bodily health problems, such as sleep deprivation, social isolation, neglect of personal hygiene and failure to eat regularly. People are not protected against the harmful consequences of excessive gaming by a paternalistic law, however: although there are laws prohibiting the sale of certain (merely violent) computer games to minors, there are no laws prohibiting people to (excessively) play computer games.

4.4 Can Computer-Simulated Human Acts or Human Acts, Made Possible by Computer-Simulation Result in Extravirtual Evils of Other Kinds?

As was established in the last section, (pure) legal moralism entails that it is legitimate to prohibit conduct on the ground that it is inherently immoral, although it causes neither harm (to the actor or to others) nor offense. Examples of penal provisions that are based on legal moralism are: prohibitions on deviant sexual activities, such as prostitution and bigamy, provided that they are “harmless (because voluntary or consented to) and unoffending (because not forced on the attention of unwilling observers)” [39]. Note that there is much inconsistency as to prohibitions that are based upon legal moralism, because they are the product of a society's values and religious principles and are, therefore, more idiosyncratic in nature [7]. In the Netherlands, for example, prostitution is legal. And in Morocco, for instance, bigamy is not prohibited.

One can find a computer-simulated variant of prostitution in the metaverse of *SecondLife*. Some people sell sex through their avatars there. They usually work for a virtual escort service or a virtual bordello. Like in the non-virtual world, they charge their clients for their services and give the owner of the escort service or bordello a percentage of their earnings. Virtual prostitution differs essentially from non-virtual prostitution, however, since no sexual activity actually occurs; it is a computer-generated animation of sex. Therefore, virtual prostitution can better be described as pornography than as prostitution [4]. Virtual prostitution is thus one of the special cases as were discussed in section 2.1.1 where a computer-simulated human act (X)

satisfies the elements of one crime intravirtually and, thereby, satisfies the elements of another crime extravirtually and, therefore, counts as crime Y in the context of its virtual environment (C) and as crime Z in the context of the non-virtual world (C). Because virtual prostitution counts as pornography in the non-virtual world the traditional concerns about morality that historically gave rise to the criminalization of prostitution do not apply [4]. The offense principle, which generally offers ground to prohibit pornography, cannot be invoked either, however. As was established in section 3.2, we have to balance the seriousness of the offense caused against the independent reasonableness (avoidability) of the offender's conduct when we invoke the offense principle. In the virtual realm, the degree of avoidability is generally high. Therefore, only the most serious offenses can tip the scales so that the offense principle can be invoked. In section 3.2 it was explained that pornography is not a serious enough offense that is to do that.

Bigamy can also occur in *SecondLife*. Although the ceremonies are not legally binding, people can marry each other through their avatars there. People who are already married in the non-virtual world can, through their avatars, marry the avatar of a person who is not their spouse. They find themselves engaged in "cross-world bigamy" [4]. People can also marry more than one avatar, which constitutes intravirtual bigamy. Neither cross-world bigamy, nor intravirtual bigamy can be brought under the scope of the prohibition on bigamy, however, since the law does not recognize *SecondLife* unions [4]. And, therefore, the underlying traditional concerns about morality that historically gave rise to the criminalization of bigamy do not apply either.

Prostitution or bigamy cannot be made possible by computer simulation, at least it is difficult to think of examples. Thus, neither of Feinberg's examples of penal provisions that are based upon legal moralism are applicable to the virtual realm. Nevertheless, I can think of two examples of prohibitions on human acts made possible by computer simulation that do seem to be based on legal moralism. The first example is the prohibition on the production, distribution and possession of virtual pornography involving sex with animals that some countries (e.g. the Netherlands) apply. Just like the virtual child pornographic images that were discussed in section 1.2.1 they are either morphed pictures or entirely computer-generated images. Since neither animals nor humans of flesh and blood are involved in its production, virtual pornography involving sex with animals cannot constitute harm to either of them. I have not found evidence that it would constitute harm to people who willingly choose to watch these images themselves either. And as long as it is distributed among individuals, which is the case, and is not made publicly accessible, there are no unwilling spectators who can be offended by it. However, Feinberg distinguishes a special class of cases where we are offended at the "bare thought" that the conduct occurs [39]. I think that the production, distribution and possession of virtual pornography involving sex with animals belongs to this special class of cases. According to Feinberg, conduct that offends at bare thought is found offensive, because it is judged to be immoral. Therefore, it should not be brought under the scope of the offense principle, but under the scope of legal moralism instead [39]. The second example is the prohibition on the production, distribution and possession of

virtual child pornography. This example is more controversial than the first. I have written extensively on this topic in another paper [49]. In this paper I argue, in short, that virtual child pornography does not do harm to others, because, contrary to non-virtual child pornography, no actual children are involved in the production.¹³ It does not do harm to the self either. Except for very rare cases in which virtual child pornography is aimed at children and instructs them how to perform certain sexual activities, there is not enough evidence that it would encourage or seduce children into participating in sexual contacts with adults, neither is there is enough evidence that it would encourage or seduce pedophiles to commit child abuse [49]. And virtual child pornography cannot be brought under the scope of the offense principle, because it is not traded in public, but secretly among pedophiles, and, therefore, there are no unwilling spectators who are deprived of the power to determine for themselves whether or not to see these images. I think that the production, distribution and possession of virtual child pornography offends at bare thought, because it is judged to be immoral. In my paper I have claimed that virtual child pornographic images are generally judged to be immoral, because they flout our sexual mentality, which is based on the equality norm, for sex between adults and children is per definition unequal [49]. The production, distribution and possession of virtual child pornography thus results in an evil of another kind than harm (to others or to the self) or offense.

4.5 Some Short Comments on What the Future Holds

In the sections 1.2.1 and 3.1 it was noted that virtual reality technologies will probably allow for new possibilities to do harm to others when they become multi-accessible in the future. In this subsection I will first describe what kind of new possibilities for human action virtual reality technologies might allow for in the future. Then I will establish how they can be harmful to others. Next I will examine whether or not virtual reality technologies could also increase the possibilities to give offense, do harm to the self or to act inherently immoral.

Philip Zhai has written a “philosophical adventure” in which he explores, from a theoretical point of view, what kind of human experiences virtual reality technologies might allow for in the future [53]. Zhai explains that state-of-the-art virtual reality technologies entail the following. One wears a helmet or goggles and earphones so that one is not able to see anything except 3-D animated video images on two small screens in front of one's eyes; nor does one hear anything except sounds from the earphones. One also wears a bodysuit, including gloves, that gives different amounts of pressure against different parts of one's body that are in accordance with one's changing video and audio sensations. Moreover one is situated in a motion-tracker that detects one's movements and feeds the signals into the computer that also

¹³ Note that child pornography differs essentially from adult pornography because children cannot consent to sex. Sex with children is, therefore, always considered abuse or rape. Child pornography is thus a recording of abuse and rape and is prohibited on the ground that it harms children and not on the ground that it is offensive.

processes all the visual and audio information so that the computer can coordinate one's movements with the images one sees and the sounds one hears. This way one is fully immersed in a virtual world, where the goggles are equivalent to one's eyes and the body suit is equivalent to one's skin [53].

In the virtual world one can encounter all kinds of virtual things that are the result of digital programming. One can perceive rocks, trees, animals etc., with which one can interact. One can, for example, pet an animal and the glove one wears will give sensory feedback so that it feels like one is really petting an animal. The virtual rocks, trees and animals one perceives may be equal to the rocks, trees and animals one has seen before in the non-virtual world, but they may also be different. It may be, for instance, that if one lifts one of the rocks it feels like it weighs as much as a rock would weigh in the non-virtual world, but it may also be that it feels like the rock is weightless. In the virtual world one can also meet other human beings. They may be virtual human beings whose behavior is totally programmed by the computer. But they may also be the virtual representations of persons who are wired to the same computer as one is oneself. When one interacts with them, one does not only get the sensory feedback belonging to the act oneself, but they also get the sensory feedback from the bodysuit and gloves they are wearing. One can, for example, shake hands with the virtual representation of another person wired to the same computer and this information is transformed and transmitted to (the glove worn by) the other person so that s/he feels like his or her hand is shaken. And much more complicated interactions are possible. Zhai, for example, describes how two persons wired to the same computer could have sex through "a seamless combination of digital simulation, sensory immersion, and functional teleoperation" [53].

Zhai does not think that human interactions mediated by virtual reality technologies can be harmful. He states: "(...) in the virtual world, nobody can physically affect us in a way our self-managed program does not allow. We set the limit in the infrastructure to prevent any serious injury." [53]. But what if a user hacks the program of another user and changes the settings? Then one could hit, kick or otherwise physically hurt the virtual representation of the other person wired to the same computer as oneself and the other person would get painful sensory feedback through his or her bodysuit. One would even be able to kill the other person when one would, for example, be able to impose an electric shock on him or her through the bodysuit. Bodily harm to the other person could also be done without being wired to the same computer oneself: one could hack into the program of a user of a virtual reality technology and add to it a virtual human being that hits, kicks or does another kind of bodily harm. To sum up, virtual reality technologies could allow for increased possibilities to do bodily harm to others through computer-simulated human acts or human acts made possible by computer simulation in the future. Yet the question arises whether or not virtual reality technologies could also allow for new possibilities to give offense, to inflict harm upon the self or to act inherently immoral.

It seems implausible that virtual reality technologies would allow for possibilities to give offense in the future that differ essentially from the possibilities that computer

simulation offers already. It was established in section 3.2 that offense in the virtual realm differs in one important aspect from harm in the virtual realm, because, contrary to harm, we cannot distinguish between intra- and extravirtual offense. It was explained that offense is a disliked mental state, caused by the wrongful conduct of others. And that a disliked mental state can only be extravirtual, because it concerns a human being and human beings are necessarily extravirtual. An extravirtual disliked mental state can be caused either by intra- or extravirtual wrongful conduct of others, but that does not make a difference for the disliked mental state. Virtual reality technologies increase the possibilities for intravirtual human acts to have extravirtual consequences. But since in the case of offense the consequence, a disliked mental state, is necessarily extravirtual, virtual reality technologies do not increase the possibilities to give offense.

Virtual reality technologies could allow for new possibilities to do harm to the self though. As was established above they could offer their users possibilities for hitting, kicking or otherwise physically hurting each other. Virtual reality technologies might, therefore, be used for dueling. They could also provide new ways to commit suicide, e.g. by imposing a fatal electric shock on oneself through one's body suit. Virtual reality technologies might be used for unlicensed practice of medicine as well. But I do not think that they will offer possibilities that differ essentially from the possibilities that computer simulation offers already. The same goes for gambling. It seems implausible that virtual reality technologies could increase the possibilities for other types of harm to the self. They may give one the impression that one, for example, drives on a motorcycle without a safety helmet, swims at an unguarded beach or is under the influence of drugs. But such impressions do not pose real risks to one's bodily health and there is thus no reason to bring them under the scope of the criminal law.

Finally, virtual reality technologies could also allow for new possibilities for inherently immoral behavior. In section 3.4 it was stated that neither prostitution nor bigamy, Feinberg's examples of inherently immoral behavior, can currently be made possible by computer simulation. Virtual reality technology could make both possible in the future. As was mentioned above, Zhai claims that people might be able to have sex in the virtual world in the future. If so, they can also sell sex and thus prostitute themselves in the virtual world. And bigamy could also be made possible by virtual reality technologies in the future. In several countries, including the Netherlands, it is allowed to marry by proxy. One can marry someone who has consented to the marriage, but is not able to attend the ceremony, for instance because s/he is far abroad and not able to come over for the marriage. In other words, one marries at a distance. Virtual reality technologies could be used for marriage by proxy. Wearing the goggles, earphones, body suit and glove two persons wired to the same computer could say yes to, exchange a ring with and kiss a virtual representation of each other and the devices would make them hear "yes", make them feel like they have a ring put around their finger and make them sense like they are kissed. Once virtual reality technologies will be used for marriage by proxy, bigamy through virtual reality technology will also be possible.

5 Conclusion

In this paper I have studied the question when virtual cybercrime should be brought under the scope of the criminal law. The paper consists of three parts. The first part of the paper is an empirical exploration; in this part I have examined what virtual cybercrime is and how, if at all, it is treated within existing legal systems. The second part of the paper is a philosophical analysis; in this part I have established, drawing from ontology and legal philosophy, what the necessary and sufficient conditions are for virtual cybercrime to obtain in order to count as crime under existing law. The third part of the paper is a moral evaluation; in this part I have studied when virtual cybercrime meets the aforementioned criteria.

In the first part of the paper I have defined cybercrime as any new or different human act that is carried out through the use of computers or computer networks and is prohibited by the enactment of a new or the extension of an existing law. I have pointed out that it differs from country to country which behaviors involving the use of computers or computer networks are outlawed, but that the Convention on Cybercrime, its Additional Protocol and the Convention on the Protection of Children against Sexual Exploitation and Sexual Abuse provide a list of new and different human acts involving the use of computers or computer networks that are commonly prohibited. This list includes: illegal access, illegal interception, data interference, system interference, misuse of devices, computer-related forgery, computer-related fraud, offences related to child pornography, offences related to infringements of copyright and related rights, acts of a racist and xenophobic nature that are committed through computer systems and “grooming.” The first five offence categories concern new forms of human activity that did not exist before the advent of computers and computer networks. That is because they can only be carried out through the use of computers or computer networks. The next offence categories concern traditional crimes where computers or computer networks are used as a tool to commit the crime in a different way.

Subsequently, I have described virtual cybercrime as cybercrime that is carried out through the use of a specific feature of computers and computer networks, namely computer simulation. It consists of a computer-simulated human act or a human act made possible by computer simulation. Contrary to ordinary cybercrime, virtual cybercrime does not concern new human activities; only different human activities. Therefore, it requires legislators to extend existing laws, but not to enact new ones. In sum, virtual cybercrime can be defined as a computer-simulated human act or a human act made possible by computer simulation that is prohibited by the extension of an existing law. It was established that the scope of virtual cybercrime is unclear, however. Currently, the production, possession and distribution of virtual child pornography is the only virtual cybercrime that is commonly prohibited, although not as commonly as non-virtual child pornography. Putative virtual cybercrimes are, for example, virtual rape, virtual killing and theft of virtual items.

In the second part of the paper I have explained that an empirical study of the law does not suffice to answer the question what the necessary and sufficient conditions are for a computer-simulated human act or a human act made possible by computer

simulation to obtain in order to be prohibited under existing law, since the production, distribution and possession of virtual child pornography is the only virtual cybercrime that is commonly prohibited and it would be a fallacy to make a general statement about virtual cybercrime on the basis of one specific instance of virtual cybercrime. Therefore, I have studied virtual cybercrime from a different point of view. As was stated in the introduction, the study of virtual cybercrime belongs to the field of legal ontology. Applied forms of ontology often put to use the tools of philosophical ontology in order to categorize things within a specific domain. I made use of this method and put to use the tools of the philosophical ontology of the American philosopher Searle in order to categorize virtual cybercrime within existing law.

Searle claims that penal provisions generally take the following form: for any x that satisfies a certain set of conditions p , x has status Y in C . So, following Searle, a particular human act (X) counts as a crime (Y) in the jurisdiction of a particular country (C) when the set of conditions (p) for that crime has been satisfied. I have explained that in legal terms the conditions that a human act needs to satisfy in order to count as a crime are called elements. The specific elements required vary depending on the crime, but there are two basic elements that are required by each crime: an *actus reus* (an unlawful act or failure to act) and a *mens rea* (a blameworthy mental state, usually it is required that the actor acts knowingly, purposely or recklessly). In fact, all crimes also require, implicitly or explicitly, that the *actus reus* must have a certain consequence, e.g. the death or injury of a person or a loss of property. This common element is called *causation*.

I have argued that, in the case of virtual cybercrime, the basic elements of a crime can be satisfied intravirtually (within the virtual environment where the act takes place) or extravirtually (outside its virtual environment), except for the element of *mens rea*, which can only be satisfied extravirtually, since it concerns the human actor, who is necessarily extravirtual. I have established that it is of crucial importance where the element of causation is satisfied, intravirtually or extravirtually, because it determines the context (C) in which the crime status (Y) of a computer-simulated human act or human act made possible by computer simulation (X) holds. A computer-simulated human act or human act made possible by computer simulation (X) that satisfies the element of causation (p) *intravirtually* counts as a crime (Y) only in the context of its virtual environment (C); a computer-simulated human act or human act made possible by computer simulation (X) that satisfies the element of causation (p) *extravirtually* counts as a crime (Y) in the context of the non-virtual world (C). In special cases a computer-simulated human act or human act made possible by computer simulation (X) can satisfy the elements of one crime intravirtually and, thereby, satisfy the elements of another crime extravirtually. Such an act counts, therefore, as crime Y in the context of its virtual environment (C) and as crime Z in the context of the non-virtual world (C).

Subsequently I have claimed that the context (C) in which the crime status (Y) of a computer-simulated human act or human act made possible by computer simulation (X) holds, its virtual environment or the non-virtual world, determines whether or not the act can be included in the scope of an existing penal provision. A computer-simulated human act or a human act made possible by computer simulation that only

counts as a crime in its virtual environment triggers remedies within that virtual environment, but not penal law. A computer-simulated human act or human act made possible by computer simulation that counts as a crime in the non-virtual world is within the reach of penal law.

To sum up, I think that it is a *necessary* condition for a computer-simulated human act or a human act made possible by computer simulation in order to be brought under the scope of the criminal law that it has an extravirtual consequence, so that it can count as a crime in the non-virtual world, provided that it also satisfies the (other) elements of a crime. I have explained that it depends on the stand one takes in the legal philosophical debate between legal positivists and natural law theorists, whether or not that is a sufficient condition as well. Legal positivists claim that laws may have any content. They would thus say that legislators and judiciaries are free to bring any computer-simulated human act or human act made possible by computer simulation that has an extravirtual consequence and also satisfies the (other) elements of a crime under the scope of penal law. Natural law theorists would say that legislators and judiciaries can only bring a computer-simulated human act or human act made possible by computer simulation that has an extravirtual consequence under the scope of penal law if the extravirtual consequence consists of a violation of a moral principle. The contemporary debate on the content of the law is dominated by the legal philosophers Hart and Dworkin and interpretations of their work. Their theories have developed such a level of subtlety and sophistication that the traditional labels of legal positivism and natural law theory hardly apply any more. Most legal philosophers would nowadays agree that the law is open to arguments that are grounded in moral principles, especially with regard to special fields or issues that are still developing, such as ICT. Taking this assumption as a starting point, I have argued that Feinberg's liberty-limiting (moral) principles, i.e. the harm principle, the offense principle, legal paternalism and legal moralism, can help to determine how the penal law should deal with virtual cybercrime.

In the third part of the paper I have first established that computer-simulated human acts or human acts made possible by computer simulation can result in several types of extravirtual harm to others and that they can, therefore, be brought under the scope of the harm principle. Then I have argued that computer-simulated human acts or human acts made possible by computer simulation can result in extravirtual offense and that they can, therefore, be brought under the scope of the offense principle. Next I have claimed that computer-simulated human acts or human acts made possible by computer simulation can result in a couple of forms of extravirtual harm to the self and that they can, therefore, be brought under the scope of legal paternalism. Subsequently I have established that computer-simulated human acts or human acts made possible by computer simulation can result in extravirtual evils of other kinds and that they can, therefore, be brought under the scope of legal moralism. Last I have argued that, in the future, virtual reality technologies might allow for new possibilities to do harm (to others or to the self) or to act inherently immoral, but that it seems implausible that virtual reality technologies would allow for possibilities to give offense that differ essentially from the possibilities that computer simulation offers already. That is because virtual reality technologies increase the possibilities

for intravirtual human acts to have extravirtual consequences. But since in the case of offense the consequence, a disliked mental state, is necessarily extravirtual, virtual reality technologies do not increase the possibilities to give offense.

Acknowledgements. I would like to thank Philip Brey, Johnny Søraker and Bert-Jaap Koops for their valuable comments and suggestions.

References

1. Hoge Raad, LJN: BQ9251 (January 31 (2012), <http://www.rechtspraak.nl>
2. Jilted woman 'Murdered Avatar', <http://news.sky.com/home/world-news/article/15127170>
3. Koepsell, D.R.: *The ontology of cyberspace: philosophy, law, and intellectual property*. Open Court Publishing Company, Peru (2003)
4. Brenner, S.W.: *Fantasy Crime: The Role of Criminal Law in Virtual Worlds*. Vanderbilt Journal of Entertainment And Technology Law 11(1), 1–97 (2008)
5. Clough, J.: *Principles of Cybercrime*. Cambridge UP, Cambridge (2010)
6. Council of Europe, *Convention on Cybercrime and Explanatory Report*, Budapest, CETS No.185 (November 23, 2001), <http://conventions.coe.int>
7. Goodman, M.D., Brenner, S.W.: *The Emerging Consensus on Criminal Conduct in Cyberspace*. International Journal of Law and Information Technology 10(2), 139–223 (2002)
8. Tavani, H.T.: *Ethics & Technology. Ethical Issues in an Age of Information and Communication Technology*. John Wiley & Sons, Hoboken (2007)
9. Council of Europe, *Additional Protocol to the Convention on Cybercrime, concerning the criminalisation of acts of a racist and xenophobic nature committed through computer systems*, Strasbourg, CETS No.189 (January 28, 2003), <http://conventions.coe.int>
10. Council of Europe, *Convention on the Protection of Children against Sexual Exploitation and Sexual Abuse and Explanatory Report*, Lanzarote, CETS No. 201 (October 25, 2007), <http://conventions.coe.int>
11. Søraker, J.H.: *The value of virtual worlds. A philosophical analysis of virtual worlds and their potential impact on well-being (doctoral dissertation)*. Ipskamp, Enschede (2010)
12. Brey, P.: *Virtual Reality and Computer Simulation*. In: Himma, K.E., Tavani, H.T. (eds.) *The Handbook of Information and Computer Ethics*, pp. 361–384. John Wiley and Sons, Hoboken (2008)
13. Allen, C.: *Artificial life, artificial agents, virtual realities: technologies of autonomous agency*. In: Floridi, L. (ed.) *The Cambridge Handbook of Information and Computer Ethics*, pp. 219–233. Cambridge UP, Cambridge (2010)
14. *List of declarations made with respect to treaty No. 185 Convention on Cybercrime*, <http://conventions.coe.int>
15. Rechtbank Amsterdam, LJN: BH9789, BH9790, BH9791 (April 2, 2009), <http://www.rechtspraak.nl>
16. Gerechtshof Leeuwarden, LJN: BK2773, BK2764 (November 10, 2009), <http://www.rechtspraak.nl>
17. Kerr, O.S.: *Criminal Law in Virtual Worlds*. University of Chicago Legal Forum; GWU Law School Public Law Research Paper No. 391, SSRN, <http://ssrn.com/abstract=1097392>

18. Dibbell, J.: A rape in cyberspace. How an Evil Clown, a Haitian Trickster Spirit, Two Wizards, and a Cast of Dozens Turned a Database Into a Society. *The Village Voice* (December 23, 1993)
19. Searle, J.R.: *Making the Social World. The Structure of Human Civilization*. Oxford UP, New York (2010)
20. Lastowka, G., Hunter, D.: Virtual Crime. 49 *New York Law School Review* 293 (2004), Available at SSRN: <http://ssrn.com/abstract=564801>
21. 18 USC § 1111, <http://www.law.cornell.edu/uscode/text>
22. Søraker, J.H.: Virtual worlds and their challenge to philosophy: understanding the “intravirtual” and the “extravirtual”. *Metaphilosophy* 43(4), 499–512 (2012)
23. Matthias, A.: The responsibility gap: Ascribing responsibility for the actions of learning automata. *Ethics and Information Technology* 6, 175–183 (2004)
24. Strikwerda, L.: Theft of virtual items in online multiplayer computer games: an ontological and moral analysis. *Ethics and Information Technology* 14(2), 89–97 (2012)
25. Second Life Seclimine VB Sample, <http://www.youtube.com/watch?v=OQvgWros7TY>
26. Brey, P.: The Physical and Social Reality of Virtual Worlds. In: Grimshaw, M. (ed.) *The Oxford Handbook of Virtuality* (forthcoming)
27. Murphy, J.G., Coleman, J.L.: *Philosophy of Law: An Introduction to Jurisprudence*. Westview Press, Boulder (1990)
28. Hart, H.L.A.: *The Concept of Law*. Clarendon Press, Oxford (1961)
29. Dworkin, R.M.: Is Law a System of Rules? In: Summers, R.S. (ed.) *Essays in Legal Philosophy*, pp. 25–60. University of California Press, Berkeley (1976)
30. Van der Burg, W.: Law and Ethics: The Twin Disciplines. *Erasmus Working Paper Series on Jurisprudence and Socio-Legal Studies* No. 10-02, SSRN, <http://ssrn.com/abstract=1631720>
31. Feinberg, J.: *The Moral Limits of the Criminal Law, Harm to Others*, vol. 1. Oxford UP, Oxford (1984)
32. Koops, B.J.: *Technology and the Crime Society: Rethinking Legal Protection*. TILT Law & Technology Working Paper No. 010/2009 and Tilburg University Legal Studies Working Paper No. 006/2009. Available at SSRN: <http://ssrn.com/abstract=1367189>
33. Feinberg, J.: *The Moral Limits of the Criminal Law, Offense to Others*, vol. 2. Oxford UP, Oxford (1985)
34. Mill, J.S.: *On Liberty*. Longmans, Green and Co., London (1865)
35. Holtug, N.: The Harm Principle. *Ethical Theory and Moral Practice* 5, 357–389 (2002)
36. Feinberg, J.: *The Moral Limits of the Criminal Law, Harm to Self*, vol. 3. Oxford UP, Oxford (1986)
37. Koops, B.J.: Sex, Kids and Crime in Cyberspace: Some Reflections on Crossing Boundaries. In: Lodder, A.R., Oskamp, A. (eds.) *Caught in the Cyber Crime Act*, pp. 63–76. Kluwer, Deventer (2009), Available at SSRN: <http://ssrn.com/abstract=1365986>
38. Dworkin, G.: Paternalism. *The Monist* 56(1), 64–84 (1972)
39. Feinberg, J.: *The Moral Limits of the Criminal Law*, vol. 4, *Harmless Wrongdoing*. Oxford UP, Oxford (1988)
40. Devlin, P.: *The Enforcement of Morals*. Oxford UP, Oxford (1965)
41. Beauchamp, T.L.: The Nature of Applied Ethics. In: Frey, R.G., Wellman, C.H. (eds.) *A Companion to Applied Ethics*, pp. 1–16. Blackwell Publishers, Malden (2003)
42. Hackers Assault Epilepsy Patients via Computer, <http://www.wired.com/politics/security/news/2008/03/epilepsy>

43. Huff, C., Johnson, D.G., Miller, K.: Virtual Harms and Real Responsibility. *IEEE Technology and Society Magazine*, 12–19 (2003)
44. Powers, T.M.: Real wrongs in virtual communities. *Ethics and Information Technology* 5, 191–198 (2003)
45. Wolfendale, J.: My avatar, my self: Virtual harm and attachment. *Ethics and Information Technology* 9, 111–119 (2007)
46. The legal rights to your 'Second Life' avatar,
<http://news.cnet.com/2100-1047-6147700.html>
47. Gerechtshof 's-Gravenhage, LJN: BO4035 (November 16, 2010),
<http://www.rechtspraak.nl>
48. Weckert, J.: Offence on the Internet. In: Collste, G. (ed.) *Ethics in the Age of Information Technology*, pp. 104–118. Centre for Applied Ethics, Linköping (2000)
49. Strikwerda, L.: Virtual Child Pornography Why Images Do Harm from a Moral Perspective. In: Ess, C., Thorseth, M. (eds.) *Trust and Virtual Worlds Contemporary Perspectives*, pp. 139–161. Peter Lang Publishing, New York (2011)
50. Internet ban for offensive image,
<http://www.independent.co.uk/news/uk/crime/internet-ban-for-offensive-image-7575915.html>
51. Collste, G.: The Internet-Doctor. In: Collste, G. (ed.) *Ethics in the Age of Information Technology*, pp. 119–129. Centre for Applied Ethics, Linköping (2000)
52. Crocco, A.G., Villasis-Keever, M., Jadad, A.R.: Analysis of Cases of Harm Associated With Use of Health Information on the Internet. *JAMA* 287(21), 2869–2871 (2002)
53. Zhai, P.: *Get Real. A Philosophical Adventure in Virtual Reality*. Rowman & Littlefield Publishers, Lanham (1998)

Research Trends in Digital Forensic Science: An Empirical Analysis of Published Research

Ibrahim Baggili¹, Afrah BaAbdallah², Deena Al-Safi², and Andrew Marrington²

¹ Tagliatela College of Engineering, Department of Electrical and Computer Engineering
and Computer Science, University of New Haven, CT

Ibaggili@newhaven.edu

² Zayed University, Advanced Cyber Forensics Research Laboratory
Abu Dhabi, United Arab Emirates, P.O. Box 4783

Andrew.Marrington@zu.ac.ae

Abstract. Digital forensic science is a new discipline. In order to advance and improve this science, stakeholders should stay abreast over the research trends in this domain. This research studied, categorized and analyzed a sample of five-hundred publications (n=500) from this discipline. The results indicated that the rate of publication in this domain continues to increase over time. Additionally, results showed an overall lack of anti-forensics research where only 2% of the sampled papers dealt with anti-forensics. In terms of research methodology, the results indicated that 17% of the sampled publications were secondary research, 36% were exploratory studies, 33% were constructive and 31% were empirical. The results also indicated a lack of basic research in this scientific discipline where most of the research (81%) was applied, and that only 19% of the sample was categorized as basic research. Additionally, results exemplified a lack of quantitative research in the discipline, with only 20% of the research papers using quantitative methods, and 80% using qualitative methods. Furthermore, results showed that the largest portion of the research (42.9%) from the examined sample originated from the United States. The findings also showed a lack of cooperative research between academia and industry, where only 10% of the research studies examined where a collaborative effort between industry and academia. Lastly, the findings indicated an increase in the disparity between the number of published articles and the number of cited articles over the years possibly indicating isolation amongst researchers in this domain.

Keywords: Digital forensic science, research trends, research methodologies, challenges in digital forensics science.

1 Introduction

Cybercrime initially emerged as a threat to computer users and businesses; it now impacts entire nations. Internet usage continues to rise and so does this threat [1]. Yet, most computer users remain unconscious of the drastic impact it has on their daily lives. The statement “The Internet is the crime scene of the 21st century” as written in the Wall Street Journal, is a realistic indicator of the current times [2].

Rogers and Seigfried in 2004 reported that cybercrime is constantly on the rise, spurring a massive progress in digital forensic science (DFS) [3]. This has consequently lured the attention of scientists towards a subset of DFS – computer forensics, establishing it as a recognized scientific discipline [4][5].

Patzakis in 2003 described computer forensics as a process of collecting, preserving, analyzing, and presenting electronic evidence where a computer has been an instrument to committing a crime [6]. This investigative methodology is used to reconstruct computer evidence as well as examine digital media storage devices in order to find electronic evidence which could lead to the source of the crime and its perpetrator(s). Furthermore, computer forensics is recommended whenever the security of an organization or company has been breached. In such a scenario, system administrators begin investigations by acquiring and analyzing the collected digital evidence.

Research has been conducted and articles published discussing various topics in DFS. Some researchers have illustrated specific definitions and processes in digital forensics [7], whereas others have published studies addressing anti-forensics [8]. Additionally, certain researchers have focused their attention to incident response and best practices when a computer crime occurs [9]. It is beyond this research paper's scope to provide a complete overview of all the research conducted under the DFS umbrella. Nonetheless, it is critical for scientists as well as practitioners to keep up with research trends associated with the science of digital forensics to acknowledge and further investigate gaps in the domain.

This research provides a strong primary contribution to this new scientific discipline, as it empirically studies research trends in the field. The primary goal is to empirically explore the path that DFS is moving towards through the categorization and analysis of a sample of five-hundred (n=500) publications issued between 1992 and 2011.

2 Literature Review

DFS is at its infancy and continues to be of utmost importance. Governmental agencies are obliged to depend on the scientific and private communities to derive novel methods and tools that allow the extraction and preservation of digital evidence in a scientific and law-abiding manner. Given the importance of this field and its impact, it is essential to collect, analyze, and categorize research in this scientific domain. This can help shed light on the discipline, aiding in a more appropriate response to cybercrime while contributing to the development of the science and professional practice in this field.

Garfinkel in 2010 argued that there is a genuine need for a well defined and collaborative approach to be undertaken by the researchers and institutions in digital forensics [10]. Garfinkel stated that “Without a clear strategy for enabling research efforts that build upon one another, forensic research will fall behind the market, tools will become increasingly obsolete, and law enforcement, military and other users of computer forensics products will be unable to rely on the results of forensic analysis” [10].

In order to combat challenges mentioned in academic literature, it is critical to consistently and empirically study research trends in DFS under a framework where research in the discipline is collected, categorized, and analyzed. The results can aid researchers and practitioners in keeping abreast over the trends in the scientific domain, as well as ensuring that they are on target with any intended scientific goals. The concept of research trends includes creating a trend map from research papers and patents and enabling the discipline's stakeholders to grasp the outline of technical trends in a particular field [11]. This concept is not new and has already been used in various disciplines such as Psychology [12], Biology [13], and Sociology. Furthermore, research trends guide the scientific community in solving challenges and potential obstacles that hinder the process of the discipline's development.

Some scientists have illustrated interest in DFS research trends reflected by their research on the future of the discipline. Rogers and Seigfried in 2004 disseminated a survey to study and characterize the top five issues in computer forensics. In their paper, they addressed the main challenges in the field, as well as issues pertaining to having a defined standardization and modular approach for data representation and forensic processing.

Moreover, in 2007, Chichao, Wenyuan, and Weiping [14] presented results in their study which aimed at exploring trends in computer crime and cybercrime research from 1974 to 2006. In their research, two-hundred and ninety two ($n=292$) papers on computer crime and cybercrime publications were drawn from the ISI Web of Science, the Science Citation Index (SCI), and the Social Science Citation Index (SSCI). Their results indicated that many papers were written in English, and most articles came from the U.S.A.

The purpose of this study was to explore the trends in DFS research from past till present. Publications for this analysis were drawn from scientific and professional publications such as Springer, Elsevier, Digital Forensics Research Conference (DFRWS), Journal of Digital Forensics Security and Law (JDFSL), National Institute of Standards and Technology (NIST), Small Scale Digital Device Forensics Journal, International Journal of Digital Evidence (IJDE), Journal of Digital Forensic Practice, International Journal of Electronic Security and Digital Forensics (IJESDF).

What made this research study unique is that the researchers did not disseminate a survey; rather, they studied, categorized and analyzed the existing literature in DFS to extrapolate an overview of the scientific discipline and the research trends associated with it over the years.

3 Methodology

The procedures followed during the data collection phase were empirical. First, the authors depended on credible publication venues to collect a sample of publications. The International Journal of Digital Evidence, Digital Forensics Research Conference, and Springer and Elsevier were powerful resources for collecting the data needed for the study. Using the collected articles, the authors built a database containing a sample of five hundred ($n=500$) research papers related to DFS. The breadth in the publications helped cover a wide range of research topics in the discipline across different time periods.

The process of categorizing the data spanned over two months. It started in the middle of June 2011 and carried on until the middle of August 2011. Here, the authors note that the categorization process was manual. Because the process was manual, bias could have possibly been introduced into the methodology due to human error. The authors note that this is a limitation in this study and that the researchers strived to remain accurate throughout the categorization phase.

During the categorization of the papers, each paper that was added to the database was examined and classified using the following categories:

- Publication year
- Forensic type (Forensic/Anti-Forensic)
- Research type (Primary, Secondary)
- Research methodology type (Exploratory/Constructive/Empirical)
- Research category (Basic/Applied)
- Research method (Qualitative/Quantitative)
- Location/Country of the research
- The originator of the research (Academic/Business or company/ Co-operation of both)
- Cited/Not cited

Based on the abovementioned categories, the authors objectively classified each paper and documented that categorization accordingly.

4 Findings and Analysis

The final database contained a sample of five hundred publications (n=500). The data for each classification category was then analyzed and graphs were created to extract general trends. The findings for each of the categories are shown in the sections that follow.

4.1 Publication Year

In this category, the authors examined the percentage of publications produced over time, as shown in Figure 1. Figure 1 illustrates how the number of research publications increased over the years. Starting in 1992, the number of published papers was insignificant compared to the number of papers that were published in 2010. This trend indicates that the number of studies in DFS has steadily increased throughout the years, though there was a slight decrease in the number of published research between 2002 and 2003 then a steady output of publications between 2007 and 2008.

The research findings also highlight a dramatic decrease in the number of research publications published between 2010 and 2011. A reason for this drop could be that data collection ceased before the end of August 2011 and that publication houses typically issue papers that were presented in 2011 in 2012 editions of journals or conference proceedings. Moreover, it is important to take into consideration the

general length of time required to submit, accept, approve, and publish peer-reviewed research papers. The authors note that this is a potential limitation in the sample of papers collected. Hence, a prediction can be made that the annual increase of papers will progress as DFS continues to capture the attention of more researchers and organizations.

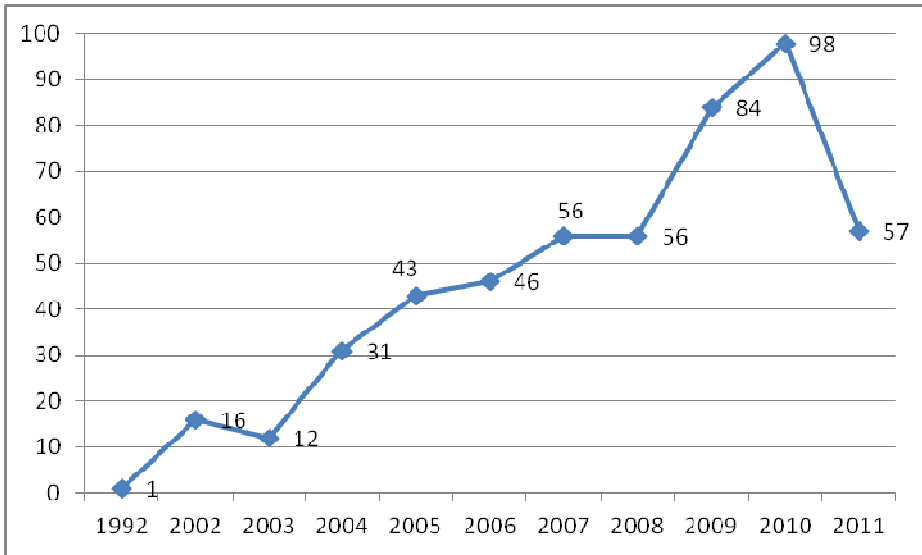


Fig. 1. Digital forensic science publications over time

4.2 Forensic Type

From the collected data, a conspicuous trend was noticed. After categorizing the research papers into forensic and anti-forensic related research papers, the results showed that only 2% of the studies discussed anti-forensics, while 98% of the publications discussed forensics.

One plausible explanation for this is that most scientific research aims at improving the effectiveness of forensic examination, whereas anti-forensics has the opposite focus. It is likely that much of the anti-forensics innovation occurs outside of the academic community altogether. Consequently, a relatively low proportion of development in anti-forensics appears in the peer-reviewed scientific literature. Irrespective of the reasons, what can be observed is that, overall, anti-forensics is neglected as a research topic in DFS.

4.3 Research Type

The results in this category illustrate that 83% of the analyzed publications were categorized as primary research studies, and 17% were secondary research studies. These results signify that DFS is being driven by primary research, reflecting the novelty and infancy of this science.

Another pattern the authors analyzed was research output type over time. Figure 2 illustrates the results obtained from that analysis.

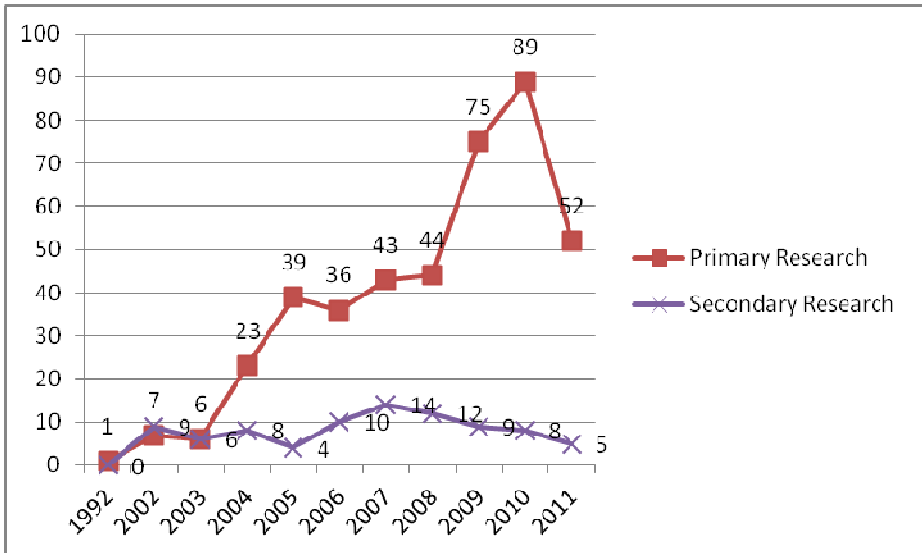


Fig. 2. Primary and secondary research over time

From Figure 2, one can clearly observe that primary research studies continued to increase from 1992 until 2010 with some minor fluctuations. In 2010 it reached the peak with 89 publications. Again, a plausible explanation for the decrease from 89 to 52 publications in 2011 is that the publication sample used in this research was not representative of all the research studies that were conducted in 2011.

As for secondary research studies, Figure 2 demonstrates fluctuations in this type of research. Overall, there has been a slight decline in secondary research in recent years.

4.4 Research Methodology

The authors examined the research methodologies used in each of the sampled publications. The three types of research methodologies used in the categorization process were: constructive, empirical and exploratory. Table 1 shows the results obtained from the data analysis process.

As shown in Table 1, the largest percentage of papers, 36%, used an empirical methodology. The reason for this may be due to the fact that this field is still new. Therefore, additional knowledge can be gained by using methods such as direct and indirect observation or experience. Furthermore, a reason for the high percentage of utilization of the empirical research methodology could be linked to the high percentage of primary research, which depends on collecting original data after gaining knowledge from a direct observation [15].

Table 1. Research methodologies used

Research methodology	% of sample papers
Empirical	36 %
Exploratory	33 %
Constructive	31 %

The results also indicate that 33% of the research papers used an exploratory methodology. The plausible explanation for that could be that since DFS is new, many exploratory studies are being pursued to gain a deeper understanding of the science.

Finally, 31% of the research papers used a constructive research methodology. Constructive research is highly linked to the computer sciences, and so is DFS. A plausible explanation for this finding is that most researchers in this domain come from a computer science background, thus many are trained to use constructive research methodologies. To examine the research methodology over time, data was analyzed further, as shown in Figure 3.

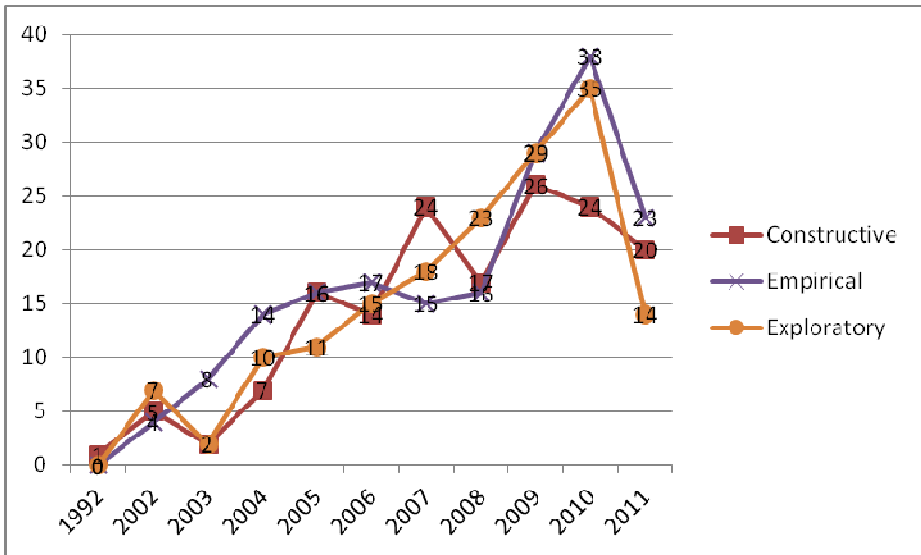


Fig. 3. Research methodologies over time

Speaking generally, there has been consistent growth in the number of exploratory studies, with some fluctuation in recent years in the rates of publication of constructive and empirical studies. The authors note that the figures for 2011 are inaccurate because of the sample’s misrepresentation of the literature that was published in 2011.

4.5 Research Category

Research can be categorized as applied or basic research. The collected data demonstrates that 81% of research studies are applied, and only 19% are basic. This is shown in Table 2.

Table 2. Research category (Applied and Basic)

Research category	% of sample papers
Applied	81 %
Basic	19 %

Applied research deals with solving practical problems and generally employs empirical methodologies [16]. This sustains the previous trends described. In contrast, basic research tends to expand the knowledge and understanding of essential principles that might not add any direct benefit or conclusions. This might be a cause of worry to the field of DFS, since it is new. The authors speculate on whether or not more basic research should be pursued by scientists in this domain to strengthen the foundational elements of this discipline.

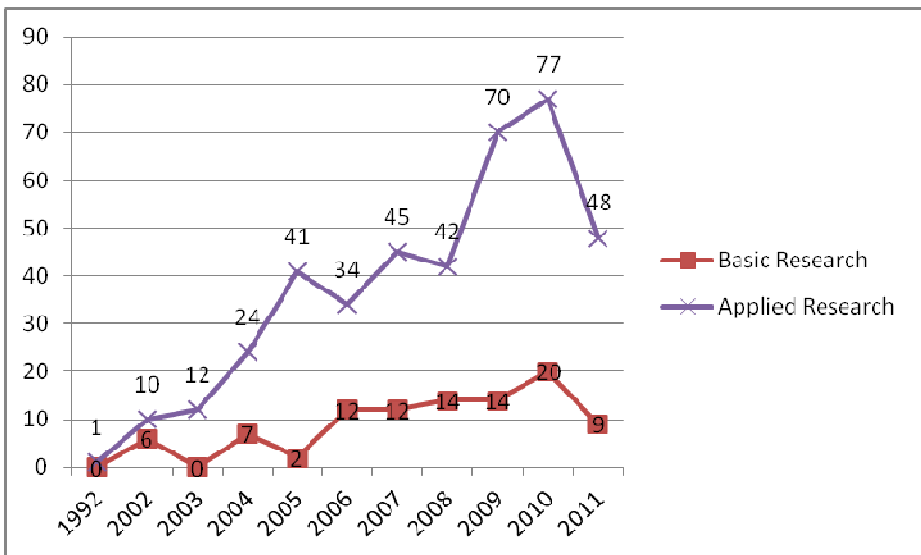


Fig. 4. Research category (Applied and Basic) over time

To gain a more thorough understanding, the authors analyzed the category of research over time as shown in Figure 4. The gap between the number of applied research publications and basic research publications is presently wide (57 in 2010, 56 in 2009). From year 2006 to 2011, the rate of basic research publication was reasonably steady (with a slight increase in 2010 followed by a decrease in 2011).

The authors speculate that these results could be attributed to the fact that DFS is a discipline that has been driven by experienced and applied practitioners in the field that may not have had traditional academic research training. Many stakeholders in DFS argue that the nature of the field is applied and therefore the amount of applied research in this domain reflects that notion.

4.6 Research Method

Table 3 shows another significant finding in DFS research trends. It illustrates that 80% of the research studies were categorized as qualitative and only 20% were quantitative. The fact that the qualitative method investigates the why and how of decision making makes it understandable as to why there is such a high percentage of qualitative research [16]. In order to further investigate the research methods used, the authors analyzed the research method used over time as shown in Figure 5.

Table 3. Research methods used

Research method	% of sample papers
Quantitative Research	20 %
Qualitative Research	80 %

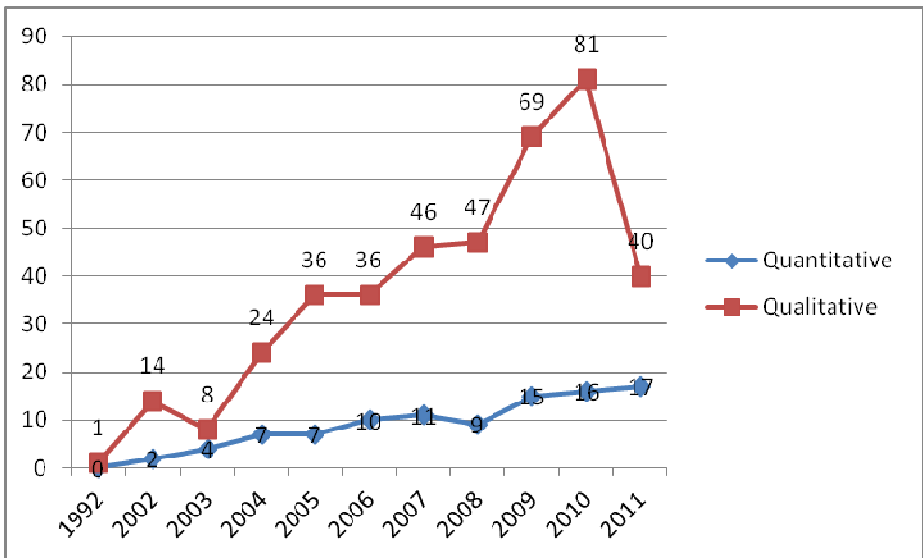


Fig. 5. Quantitative and qualitative research over time

Figure 5 clarifies the relationship between the research methods and time. The volume of quantitative research (as measured by published research papers) has steadily increased over the years. Qualitative research, however, experienced two sharp spikes, one in 2004 (24 from 8 in 2003), and in 2009 (69 from 2008, and 81 in 2010).

Overall, these graphs illustrate that both research methods are increasing. The significant drop in qualitative research in 2011 could also be attributed to the aforementioned sample problem; the data was collected before all the 2011 research studies were published.

4.7 Location (Country of Origin) of Research

It is important to highlight the location of research publications because it leads to the discovery of the countries that are pursuing research initiatives in DFS. Therefore, the data collected was classified based on the institution and/or organization's country that issued the study. Some of the publications were issued in one country, yet a few were issued in co-operation between international universities and communities. Table 4 shows each country and the number of published articles released from that specific country.

Table 4. Publications by country

Country	# of Papers	% of sample
USA	228	42.9
UK	49	9.2
Australia	37	7.0
China	23	4.3
Korea	22	4.1
India	17	3.2
Germany	16	3.0
Ireland	15	2.8
Italy	13	2.4
Taiwan	10	1.9
Canada	9	1.7
France	7	1.3
Japan	7	1.3
Malaysia	7	1.3
Hong Kong	6	1.1
UAE	6	1.1
Netherlands	6	1.1
Singapore	6	1.1
Sweden	5	0.9
Norway	5	0.9
South Africa	5	0.9
Belgium	3	0.6
New Zealand	3	0.6
Poland	3	0.6
Greece	2	0.4
Brazil	2	0.4
Finland	2	0.4

Table 4. (Continued.)

Iran	2	0.4
Switzerland	2	0.4
Saudi Arabia	2	0.4
Turkey	2	0.4
Algeria	1	0.2
Croatia	1	0.2
Romania	1	0.2
Luxembourg	1	0.2
Indonesia	1	0.2
Mexico	1	0.2
Pakistan	1	0.2
Uganda	1	0.2
Spain	1	0.2
Qatar	1	0.2

Table 4 illustrates that the United States of America holds the highest number of publications at 42.9% of the total sample. The United Kingdom comes in second place with a significant difference in percentage at 9.2%. China, Korea, India, Germany, Ireland, Italy, Taiwan, and Canada follow with different percentage variations at 4.3%, 4.1%, 3.2%, 3.0%, 2.8%, 2.4%, 1.9%, and 1.7% respectively.

4.8 Research Originator

One of the categories used in this study to classify research articles was the originator of the research studies. Some of the papers were published by professors and academic experts, whereas others were prepared by digital forensic practitioners. Additionally, some of the publications were a cooperative effort between academia and private sector organizations. There have been continuous deliberations amongst experts in DFS regarding a stronger collaboration between academia and private sector with regards to DFS research. The authors thought it would be interesting to explore how much of the research originated from academic institutions, how much originated from companies, and lastly, the amount of publications in which companies and academic institutions jointly collaborated on. Table 5 shows the percentage of papers categorized by the originator.

Table 5. Research originator

Originator	% of sample papers
Academic	60 %
Industry	29 %
Joint	10 %
N/A	1 %

Table 5 depicts that 60% of the publications were issued by academics. This high percentage indicates that universities and academics are the most productive in terms of research in DFS. Furthermore, 29% of the publications were issued by companies or organizations that were either interested or invested in DFS. Lastly, only 10% of the research papers stemmed from a cooperative effort between academics and organizations. The authors couldn't trace the origin of 1% of the collected publications.

These results illustrate a clear dichotomy between academia and organizations when it comes to DFS research. The authors understand the importance of collaboration between academics and private organizations since the science of digital forensics is new and concurrently in practice.

4.9 Cited Papers

It is accepted practice to regard impact publications to have a significant number of citations. Generally a large number of citations for a publication indicates that it is useful, effective, and in demand. From the five-hundred (n=500) publications that comprised the study, some were cited, and some were not. During this process, Google Scholar was used in order to check if an article was cited or not. Table 6 shows the percentage of cited and non-cited papers.

Table 6. Cited and non-cited articles

Cited/non-cited	% of sample papers
Cited articles	66 %
Non-cited articles	34 %

The percentage of cited articles was 66%, unlike the percentage of non-cited articles which was 34%. This may indicate that DFS is gaining more attention by academics and organizations. Of course, the number of cited papers will continue to increase. Perhaps the more interesting metric is the proportion of papers which are cited – as this may indicate the proportion of the literature which is relevant and useful to other authors (and perhaps, indirectly, to industry and law enforcement). The analysis of the number of cited publications over time is illustrated in Figure 6.

Figure 6 also compares the number of published articles to the number of articles cited by publication year. The data shows that the number of publications in the sample continues to increase. A trend can be noticed as a significant increase in disparity between the number of articles published and the number cited articles starting 2009-2011. The authors speculate that this is most likely because newer articles have not been sufficiently exposed to other researchers for further work to be built on top of them yet, although it may also indicate a decrease in the proportion of published articles which may be regarded as seminal to DFS.

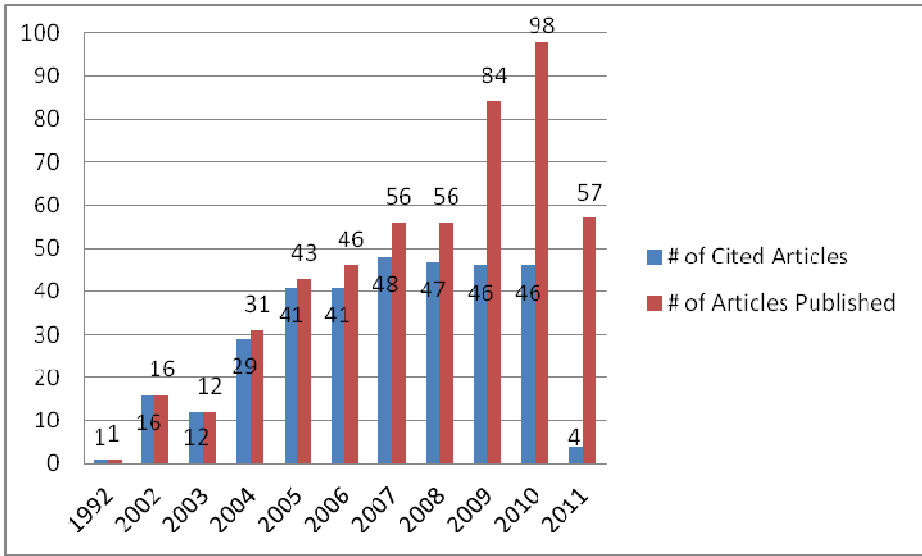


Fig. 6. Articles cited and published by year

5 Conclusions and Future Work

DFS has captured the attention of the scientific community. This, in turn, has led to the increase in the number of studies and research in the discipline. Yet, it is still a new scientific discipline, which makes it difficult to clearly predict where this field is heading. Therefore, this research paper attempted to highlight the path that this new field of science is leaning towards; by collecting a random sample of papers (n=500) and classifying them into different categories in an attempt to arrive at research trend patterns.

There were a few limitations that affected the research in this paper. The main one was the difficulty in finding ontological research topics where research papers could be classified under. Due to this limitation, the authors were not able to classify the research publications into research topics such as media forensics, small scale device forensics, network forensics etc. This challenge also points to the idea that perhaps a more accepted ontology of DFS research topics should be researched and accepted within the scientific community.

The authors believe that the database of this primary research needs to be constantly updated for more accurate results, allowing DFS stakeholders to stay abreast over research trends across time. As the database grows, the sample size of the categorized research publications will increase as well. Constantly increasing the sample size will lead to better and more accurate future results.

This was the first empirical step in defining the path that this new discipline is taking. The authors hope that researchers will continue to expand on this research topic as the discipline of DFS continues to mature.

References

1. Wolf, S.: A click away from a mugging. *The Australian* (2004), https://intranet.stjohns.sa.edu.au/curriculum/infotech/ITissues/pdf/A_click_away_from_a_mugging.pdf (retrieved)
2. Solms, P.B.V.: S.Africa: The crime scene of the 21st century. *Wall Street Journal* (2011)
3. Rogers, M.K., Seigfried, K.: The future of computer forensics: a needs analysis survey. Center for Education and Research in Information Assurance and Security, Purdue University, 656 Oval, West Lafayette, IN 47907, USA (2004)
4. Whitcomb, C.M.: An Historical Perspective of Digital Evidence: A Forensic Scientist's View. *International Journal of Digital Evidence* 1(1) (Spring 2002)
5. Rogers, M.: The role of criminal profiling in the computer forensics process. Elsevier Ltd. Center for Education and Research in Information Assurance and Security (CERIAS), Purdue University (2003)
6. Patzakis, J.: Computer Forensics as an Integral Component of the Information Security Enterprise G. Software, California (2003)
7. Oseles, L.: Computer Forensics: The Key to Solving the Crime. INSS 690 (2001)
8. Peron, C.S.J., Legary, M.: Digital Anti-Forensics: Emerging trends in data transformation techniques (2005)
9. Al-Zarouni, M., Al-Hajri, H.: A Proof-of-Concept Project for Utilizing U3 Technology in Incident Response. School of Computer and Information Science, Edith Cowan University (2007)
10. Garfinkel, S.L.: Digital forensics research: The next 10 years. A Naval Postgraduate School, Monterey, USA (2010)
11. Nanba, H., Kondo, T., et al.: Automatic creation of a technical trend map from research papers and patents. In: Proceedings of the 3rd International Workshop on Patent Information Retrieval, pp. 11–16. ACM, Toronto (2010)
12. Gauvin, L., Spence, J.C.: Psychology research on exercise and fitness: Current research trends and future challenges (1995)
13. Marzluff, J.M., Bowman, R., et al.: A historical perspective on urban bird research: trends, terms, and approaches. *Avian Ecology and Conservation in an Urbanizing World*. Kluwer Academic, Boston (2001)
14. Lu, C., Jen, W., Chang, W.: Trends in Computer Crime and Cybercrime Research During the Period 1974-2006: A Bibliometric Approach. In: Yang, C.C., et al. (eds.) PAISI 2007. LNCS, vol. 4430, pp. 244–250. Springer, Heidelberg (2007)
15. Teller, P.: Whither Constructive Empiricism? *Philosophical Studies* 106(1-2), 123–150 (2001)
16. Gruman, J.C.: Basic vs. Applied Research: Finding a Balance. *The Chronicle of Higher Education* 49(29), B.20 (2003)

Face Recognition Based on Wavelet Transform and Adaptive Local Binary Pattern

Abdallah Mohamed^{1,2} and Roman V. Yampolskiy¹

¹ Computer Engineering and Computer Science,
University of Louisville, Louisville, KY, 40292, USA

² Department of Mathematics, Menoufia University,
Shebin El-Koom, Menoufia, 32511, Egypt
{aamoha04, roman.yampolskiy}@louisville.edu

Abstract. Local Binary Pattern (LBP) is a very efficient local descriptor for describing image texture. In this paper, we propose a novel face recognition technique based on wavelet transform and the least square estimator to enhance the classical LBP. First, Wavelet transform is used to decompose a given image into four kinds of frequency images from which the features of that image can be extracted. Then, the least square estimation of local difference between each image pixel and its neighborhoods is used to build the adaptive LBP. Finally, the classification accuracy is computed using a nearest neighbor classifier with Chi-square as a dissimilarity measure. Experiments conducted on three face image datasets (ORL dataset and two avatar face image datasets); show that the proposed technique performs better than traditional methods (single scale) LBP and PCA, Wavelet Local Binary Pattern (WLBP) and Adaptive Local Binary Pattern (ALBP) in terms of accuracy.

Keywords: Face recognition, avatar, Adaptive Local Binary Pattern (ALBP), wavelet transform.

1 Introduction

Face recognition has become the center of attention of many researchers during the last few decades because of its wide range of practical applications, including access control, surveillance systems and biometric identification. However, after all these years of research to find out proper human face recognition techniques, identification of avatars in virtual worlds is still an open problem [1].

Face recognition techniques can be divided into two categories [2]: holistic matching and local feature-based methods. Local Binary Pattern (LBP) is one of the most popular local feature-based methods.

LBP, first proposed by Ojala et al. [3], is a powerful way for texture description and it was applied to face recognition for the first time by Ahonen et al. [4]. Later, some methods further developed LBP for either recognizing human faces or avatar faces. For example, Yang et al. [5] applied LBP for face recognition with Hamming distance constraint. Chen et al. [2] used Statistical LBP for face recognition. Mohamed et al. [6] applied hierarchical multi-scale LBP with wavelet transform to

recognize avatar faces. In this paper, we propose a novel face recognition approach combining wavelet transform with a new LBP type (adaptive LBP) to recognize both human and avatar faces. The efficacy of the new method is demonstrated by the experiments on ORL dataset and two avatars datasets from Second Life and Entropia virtual worlds.

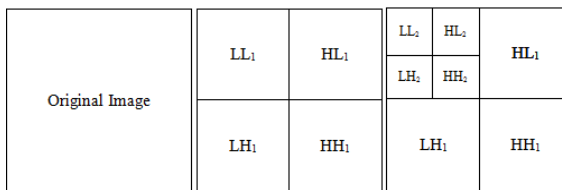
The remaining of this paper is organized as follows; Section 2 provides an introduction to wavelet decomposition. In Section 3, an overview of the LBP is presented. Section 4, presents the proposed method, Wavelet Adaptive LBP (WALBP). In Section 5, experimental results are presented followed by conclusions in Section 6.

2 Review of Wavelet Transform

Wavelet Transform (WT) is a popular tool for image analysis. It provides multi-resolution analysis of the image by using coefficient matrices [7]. It has many applications in signal and image processing including multi-resolution analysis, computer vision and graphics. Many articles have discussed its mathematical background and advantages [8]. WT can be applied in image decomposition for many reasons [8]:

- Using WT to decompose an image reduces the resolution of the sub-images and then the computational complexity will also be reduced.
- WT decomposes an image into sub-images corresponding to different frequency ranges and this can lead to minimize the computational overhead.
- Using WT allows obtaining the local information in different domains (space and frequency).

Decomposing an image with the first level of WT provides four sub-bands LL₁, HL₁, LH₁ and HH₁ (see Fig 1.a.).



(a)



(b)

Fig. 1. a) Structure of one-level and two-level wavelet decomposition b) an example of decomposing an image using one-level and two-level wavelet decomposition

The sub-band LL represents the approximation coefficient of the wavelet decomposition and it has the low frequency information of the face image [7]. This information includes the common features of the same class. The other sub-bands represent the detailed coefficients of the wavelet decomposition and they have most of the high frequency information of the face image. This information includes local changes of face image such as illumination and facial expression. To improve recognition performance we have to enhance the common features of the same class and remove changes. So, during our experiments we considered only the approximation images.

Decomposing an image with two scales will give us seven sub-bands [8]: LL2, HL2, LH2, HH2, HL1, LH1 and HH1 as in Fig. 1.

3 Local Binary Pattern (LBP)

3.1 LBP Operator

The local binary pattern (LBP) operator was proposed by Ojala et al. [3], to describe local textural patterns. It works by thresholding the pixels in a certain block of an image with its center, multiplied by powers of two and then added together to form the new value (label) for the center pixel [9]. The output value of the LBP operator for a block of 3×3 pixels can be defined as follows [9]:

$$LBP(x_c, y_c) = \sum_{i=0}^7 2^i S(g_i - g_c) \quad (1)$$

where g_c corresponds to the gray value of the central pixel, (x_c, y_c) are its coordinates, g_i ($i = 0, 1, 2, \dots, 7$) are the gray values of its surrounding 8 pixels and $S(g_i - g_c)$ can be defined as follows:

$$S(g_i - g_c) = \begin{cases} 1, & g_i \geq g_c \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

The LBP operator was extended to use neighborhoods of different sizes to be able to deal with large scale structures that may be the representative features of some types of textures [4, 10]. In the following the notation (P, R) will be used as indication of neighborhood configurations. P represents the number of pixels in the neighborhood and R represents the radius of the neighborhood. The neighborhood can be either in a circular or square pattern (Fig. 2 gives an example of a circular neighborhood for the same neighbor set of pixels but with different values of the radius).

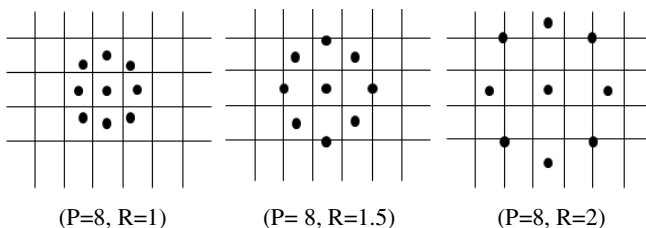


Fig. 2. Three different LBP operators [4, 6]

LBP operator can also be extended to other definitions and patterns. One of the most important and successful extensions to the basic LBP operator is called uniform LBP (ULBP). An LBP is called uniform if the binary pattern contains at most two different conversions from 0 to 1 or 1 to 0 when the binary string is viewed as a circular bit string [4]. For example, 11000011, 00111110 and 10000011 are uniform patterns. The results of statistical analysis indicated that most of patterns in images are uniform patterns. Ojala reported that with (8, 1) neighborhood, uniform patterns account for a little less than 90% of all patterns and with (16, 2) neighborhood, uniform patterns account for around 70% of all patterns [4].

3.2 LBP Histogram

After labeling an image with the LBP the histogram of the labeled image can be defined as follows [10]:

$$H_i = \sum_{x,y} I(f(x, y) = i), i = 0, 1, \dots, n - 1 \quad (3)$$

where 'n' is the number of different labels produced by the LBP operator, $f(x, y)$ is the labeled image and $I(A)$ is a decision function with value 1 if the event A is true and 0 otherwise.

To form the LBP histogram, the image has to be divided into sub-regions. Then, the LBP histogram for each sub-region has to be computed and then all sub-regions histograms have to be combined to form the feature histogram of the whole image [11].

4 Wavelet Adaptive LBP (WALBP)

We propose an algorithm to work with gray scale images. These images can be either from the real world (human images) or from virtual worlds (e.g. Second Life and Entropia). Our algorithm has three steps: preprocessing datasets, extracting features and classifying each image to its subject.

4.1 Preprocessing Datasets

For the two virtual world datasets (Second Life and Entropia datasets), we have to get rid of the background in each image if it is present. The presence of the background of an image has an effect of identifying that image. To remove the background of an image we manually cropped the facial portion of that image on the bases that the new facial image should have two eyes, nose and mouth in each image.

During our experiments we decomposed all facial images using the first level of decomposition and the low frequency coefficient of decomposition is used in the next step to extract the facial image features.

4.2 Adaptive Local Binary Pattern (ALBP)

In an image, to improve the classification performance using the LBP by reducing the estimation error of local difference between each pixel and its neighbors a new parameter called weight (w_p) is defined in the LBP equation. We call this new approach Adaptive LBP (ALBP).

So the new definition of the LBP equation will have the following form [12, 13]:

$$ALBP_{p,R} = \sum_{p=0}^{P-1} 2^p S(g_p * w_p - g_c) \quad (4)$$

where the weight w_p can be computed using:

$$w_p = \bar{g}_p^T \bar{g}_c / (\bar{g}_p^T \bar{g}_p) \quad (5)$$

where $\bar{g}_c = [g_c(I,1); g_c(I,2); \dots; g_c(N,M)]$ is a column vector that contains all possible values of any pixel $g_c(i,j)$, $N \times M$ is the size of an image and $\bar{g}_p = [g_p(I,1); g_p(I,2); \dots; g_p(N,M)]$ is the corresponding vector for all $g_p(i,j)$ pixels. We have to note that each weight w_p is computed along one orientation $2\pi p/P$ for the whole image.

4.3 Classification

The last step in our algorithm is to classify each face image to its class. We have to build the distance matrix of the training images and the testing ones using the ALBP definition and the definition of the Chi-Square distance. The Chi-Square distance has the following form [4]:

$$D(X, Y) = \sum_{n=1}^N \frac{(X_n - Y_n)^2}{X_n + Y_n} \quad (6)$$

where X is the testing images and Y is the training images.

The distance matrix is used by the definition of the nearest neighbor classifier to compute the accuracy rate.

5 Experiments

In this section, we verify the performance of the proposed algorithm on three different datasets: one real world well known human dataset (ORL) and two virtual world agents (avatar) datasets (see Fig. 3). The proposed method is compared with well-known methods of face recognition, PCA, single scale LBP and wavelet LBP.

5.1 Experimental Setup

Three facial image datasets were used to evaluate the proposed WALBP method. The first one is the ORL dataset. The ORL dataset contains 400 images representing 40 distinct subjects [14]. Each subject has 10 different images. These images were taken at different times, with varying lighting, pose angle, facial expressions (open eyes, closed eyes, smiling, not smiling) and facial details (wearing glasses or no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position and each is grayscale image with a resolution of 92 x 112 pixels [14]. We have used all images in this dataset during our experiments without cropping the facial portion from each image but we used them as they were in the original dataset. After applying the first level of wavelet decomposition the resolution of each image in the ORL dataset was changed from 92 x 112 to 46 x 56.

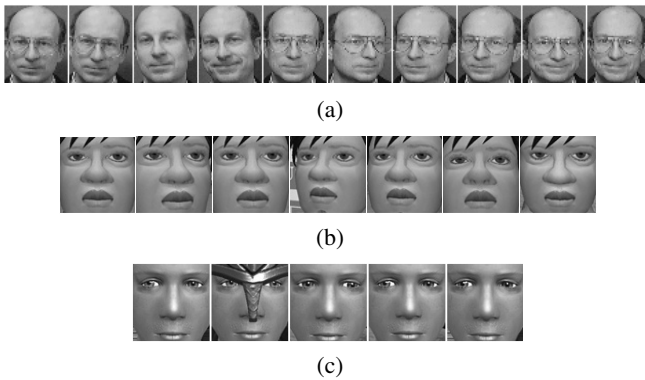


Fig. 3. Samples of one subject of facial images from: a) ORL dataset b) Second Life dataset c) Entropia dataset

For the other two datasets, the first one was collected from the Second Life (SL) virtual world [15]. This dataset contains 581 gray scale images with size 1280 x 1024 each to represent 83 different avatars. Each avatar subject has different 7 images for the same avatar with different frontal pose angle (front, far left, mid left, far right, mid right, top and bottom) and facial expression.

The second virtual world dataset was collected from Entropia (ENT) Universe virtual world [16] and contains 490 frontal images of 98 subjects or avatars (5 images per avatar) with size 407 x 549 pixels each. Each avatar subject's images have different frontal angle and details (wearing a mask or no). The facial part of each virtual world image used in our experiments was manually cropped from the original images (for the second Life dataset the size will be 260 x 260) based on the location of the two eyes, mouth and the nose. Each cropped Entropia facial image was rescaled to the size of 180 x 180 pixels. After applying the first level of wavelet decomposition the resolution of each face image in the Second Life dataset will be reduced to be 130 x 130 and for Entropia dataset the new resolution for each face image becomes 90 x 90.

The intensity of all images used in all experiments is normalized to reduce the variance of illumination.

5.2 Experimental Results

We performed many experiments to proof the superiority of our algorithm over the other methods used in experiments.

In the first one we compared ALBP with our proposed method WALBP, in this experiment the first 5 images from each subject in the ORL were used for training and the rest were used for testing and then the training and testing sets were swapped. The average of the two experiments was used as the final accuracy rate. We followed the same protocol with the other two datasets but with different number of training and testing images. In SL dataset the first 4 images were used for training and the rest were used for testing and then the training and testing images were swapped. In ENT dataset the first 3 images were used for training and the rest for testing and then training and testing images were swapped. The result of this experiment using different LBP operators can be seen in Fig. 4.

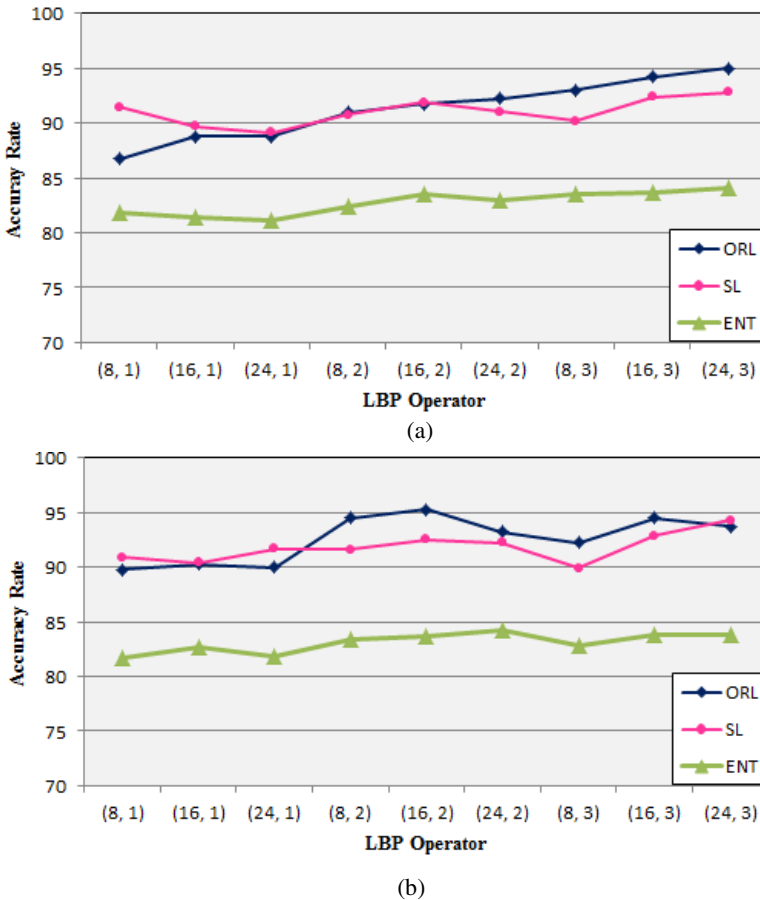


Fig. 4. Recognition rate average using: a) ALBP algorithm b) WALBP algorithm

Table 1. Accuracy rates for SL dataset

Algorithm	The number of training images		
	1	3	5
PCA	76.71% [17]	84.29%	87.35%
LBP	76.50% [17]	86.37%	89.23%
WLBP	79.55% [17]	88.19%	90.56%
WALBP	84.96%	90.66%	93.77%

From Fig 4 we can recognize that in most of the cases the accuracy rate of using WALBP is better than that of using ALBP. Also the processing time of using WALBP is less than that of using ALBP.

We also compared WALBP with PCA, Traditional LBP and WLBP using different number of training images from each subject and the result are shown in table 1. We did this experiment using only the SL dataset.

It is very clear from Fig 4 and table I that our proposed method can achieve better result than the other algorithms in terms of accuracy.

6 Conclusions

In this paper, a novel LBP face recognition approach (WALBP) is proposed based on a new definition of the LBP operator and wavelet transform to increase the recognition rate of facial images. Experimental results show the effectiveness of the WALBP in recognizing faces from both real and virtual worlds. In the future work, we will add statistical features such as mean and standard deviation to the multi-scale version of the adaptive LBP to increase the recognition rate of faces and facial expressions.

References

1. Gavrilova, M.L., Yampolskiy, R.V.: Applying Biometric Principles to Avatar Recognition. In: International Conference on Cyberworlds, Singapore, pp. 179–186 (2010)
2. Chen, L., Wang, Y.H., Wang, Y.D., Huang, D.: Face Recognition with Statistical Local Binary Patterns. In: 8th International Conference on Machine Learning and Cybernetics, Baoding, pp. 2433–2438 (2009)
3. Ojala, T., Pietikainen, M., Harwood, D.: A comparative Study of Texture Measures with Classification Based on Feature Distributions. *Pattern Recognition* 29, 51–59 (1996)
4. Ahonen, T., Hadid, A., Pietikainen, M.: Face Description with Local Binary Patterns: Application to Face Recognition. *IEEE Transaction on Pattern Analysis and Machine Intelligence* 28, 2037–2041 (2006)
5. Yang, H., Wang, Y.D.: A LBP-based Face Recognition Method with Hamming Distance Constraint. In: 4th International Conference on Image and Graphics, Sichuan, pp. 645–649 (2007)

6. Mohamed, A.A., D'Souza, D., Baili, N., Yampolskiy, R.V.: Avatar Face Recognition Using Wavelet Transform and Hierarchical Multi-scale LBP. In: 10th IEEE International Conference on Machine Learning and Applications, Honolulu, Hawaii, pp. 194–199 (2011)
7. Garcia, C., Zikos, G., Tziritas, G.: A Wavelet-based Framework for Face Recognition. In: 5th European Conference on Computer Vision, Freiburg, Allemagne, pp. 84–92 (1998)
8. Mazloom, M., Ayat, S.: Combinational Method for Face Recognition: Wavelet, PCA and ANN. In: International Conference on Digital Image Computing: Techniques and Applications, Canberra, pp. 90–95 (2008)
9. Meng, J., Gao, Y., Wang, X., Lin, T., Zhang, J.: Face Recognition Based on Local Binary Patterns with Threshold. In: IEEE International Conference on Granular Computing, San Jose, CA, pp. 352–356 (2010)
10. Wang, W., Chang, F., Zhao, J., Chen, Z.: Automatic Facial Expression Recognition Using Local Binary Pattern. In: 8th World Congress on Intelligent Control and Automation, Jinan, China, pp. 6375–6378 (2010)
11. Liu, X., Du, M., Jin, L.: Face Features Extraction Based on Multi-scale LBP. In: 2nd International Conference on Signal Processing Systems, pp. v2 438- v2 441 (2010)
12. Guo, Z., Zhang, L., Zhang, D., Zhang, S.: Rotation Invariant Texture Classification Using Adaptive LBP with Directional Statistical Features. In: 17th IEEE International Conference on Image Processing, Hong Kong, pp. 285–288 (2010)
13. Mohamed, A.A., Gavrilova, M.L., Yampolskiy, R.V.: Artificial Face Recognition using Wavelet Adaptive LBP with Directional Statistical Features. In: 12th International Conference on Cyberworlds, Darmstadt, Germany (2012)
14. The ORL Database, <http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html>
15. Second Life, <http://www.secondlife.com>
16. Entropia Universe, <http://www.entropiauniverse.com>
17. Mohamed, A.A., Yampolskiy, R.V.: An Improved LBP Algorithm for Avatar Face Recognition. In: 23th International Symposium on Information, Communication and Automation Technologies, Sarajevo, Bosnia & Herzegovina, pp. 1–5 (2011)

Similarity Preserving Hashing: Eligible Properties and a New Algorithm MRSH-v2

Frank Breitinger and Harald Baier

da/sec Biometrics and Internet Security Research Group
Hochschule Darmstadt, Darmstadt, Germany
{frank.breitinger,harald.baier}@h-da.de

Abstract. Hash functions are a widespread class of functions in computer science and used in several applications, e.g. in computer forensics to identify known files. One basic property of cryptographic Hash Functions is the avalanche effect that causes a significantly different output if an input is changed slightly. As some applications also need to identify similar files (e.g. spam/virus detection) this raised the need for *Similarity Preserving Hashing*. In recent years, several approaches came up, all with different namings, properties, strengths and weaknesses which is due to a missing definition.

Based on the properties and use cases of traditional Hash Functions this paper discusses a uniform naming and properties which is a first step towards a suitable definition of Similarity Preserving Hashing. Additionally, we extend the algorithm MRSH for Similarity Preserving Hashing to its successor MRSH-v2, which has three specialties. First, it fulfills all our proposed defining properties, second, it outperforms existing approaches especially with respect to run time performance and third it has two detections modes. The regular mode of MRSH-v2 is used to identify similar files whereas the **f**-mode is optimal for fragment detection, i.e. to identify similar parts of a file.

Keywords: Digital forensics, Similarity Preserving Hashing, fuzzy hashing, MRSH-v2, properties of Similarity Preserving Hashing.

1 Introduction

Within the area of computer forensics investigators are overwhelmed with digital data. Traditional books, photos, letters and long-playing records (LPs) turned into ebooks, digital photos, email and mp3. In order to handle this amount of data, investigators need methods to automatically identify suspect files (e.g., images of child abuses). Normally the proceeding is quite simple: the investigator computes hash values (fingerprints) of all files which he finds on a storage medium and performs database lookups, e.g., within the widespread National Software Reference Library (NSRL, [1]). Besides finding exact duplicates using a cryptographic Hash Function, it is also necessary to uncover similar files using *Similarity Preserving Hashing*.

Cryptographic Hash Functions are well established and thus a clear definition exists. This is in contrast to Similarity Preserving Hashing where the (preferable) properties are unclear although there are existing approaches like `ssdeep` from Kornblum ([2], 2006), `sdfhash` from Roussev ([3], 2010) or `bbHash` from Breitinger et al. ([4], 2012).

There are two main contributions of this paper. On the one hand it discusses foundations for *Similarity Preserving Hashing* as a first step towards a definition of Similarity Preserving Hashing. Our discussion comprises a uniform naming, five important properties, which we consider to be eligible to be part of a definition of Similarity Preserving Hashing and two use cases to understand our proposals. On the other hand, we present a new version of the existing Similarity Preserving Hashing algorithm `MRSH` ([5]). We show that our version `MRSH-v2` is compliant with the proposed defining properties. Moreover, practical tests and a theoretical analysis reveal that `MRSH-v2` outperforms existing approaches with respect to run time performance and allows to detect similar files and file fragments.

The rest of the paper is organized as follows: At first we present properties (Sec. 2.1), use cases (Sec. 2.2) and Bloom filters (Sec. 2.3) for Similarity Preserving Hashing in the section on foundations (Sec. 2). Next, we shortly review related work in Sec. 3. Sec. 4.1 introduces the concepts of the existing algorithm `MRSH`, while Sec. 4.2 deals with our improved variant `MRSH-v2`. Based on the foundations and the new algorithm, Sec. 5 evaluates `MRSH-v2` and presents experimental results. Sec. 6 concludes our paper.

2 Foundations of Similarity Preserving Hashing

The topic uses the term Similarity Preserving Hashing that is also known as similarity digest, fuzzy Hash Function or similarity preserving Hash Function. Within a first step we like to come up with a uniform naming.

Each existing approach consists of two sub-functions: one for generating hash values / fingerprints¹ and one for comparing them. Thus, Similarity Preserving Hashing² (abbreviated SPH) consists of a

similarity preserving hash function, (abbreviated SPHF) which is a function / algorithm to create a hash value / fingerprint and a **comparison function**, (abbreviated CP) that outputs a similarity score for two hash values / fingerprints.

In contrast to cryptographic Hash Functions, we do not expect a fixed-length hash value (more in Sec. 2.1) and therefore the term *hashing* might be a little bit confusing. Nevertheless, as some uses cases are almost identical to traditional Hash Functions, we agreed on this term.

Talking about the similarity of files, one usually distinguishes between byte level similarity and semantic similarity. In what follows we treat each input as a

¹ The term fingerprint or hash value is due to cryptographic Hash Functions.

² We also use the long term *Approach for Similarity Preserving Hashing*.

byte sequence and consider byte level similarity. Thus, when talking about the similarity of two files, we generally talk about the similarity of the underlying byte sequences.

2.1 Properties of Similarity Preserving Hash Functions

We bring five properties for Similarity Preserving Hashing into being that are discussed in the following and later used as a benchmark. This results from the necessity that we do not have a clear definition right now. Inspired by cryptographic Hash Functions we distinguish between *general properties* (P1-P3) and *security properties* (P4-P5). Finally this sections briefly discusses the impact of the properties to the existing algorithms: `ssdeep` ([2]), `sdhash` ([3]), and `bbhash` ([4]).

General Properties for SPH

- P1 - **Compression.** The output (hash value) of a SPHF is much smaller than the input (the shorter the better). In contrast to traditional Hash Functions we do not expect a fixed-length hash value. The reason for compression is two-spread. First, a short hash value is space-saving and second, the comparison of small hash values is faster.
- P2 - **Ease of Computation.** Generating a hash value is ‘fast’ in practice for all kinds of inputs. This is comparable to the property of a classical hash function like SHA-1. It is obvious that ease of computation is a prerequisite for a SPHF to be usable in practice.
- P3 - **Similarity Score.** In order to compare two hash values we need a ‘comparison function’³. Input of the comparison function are two hash values, its output is a value from 0 to X , where X is the maximum match score. A match score of X indicates that the hash values are identical or almost identical, which implies that the input files are identical or almost identical, too. Preferably the similarity score is between 0 and 100 and represents a percentage value. If the comparison function is linear, it is easy to map the match score in $[0, X]$ to the corresponding value in $[0, 100]$.

Security Properties for SPH

- P4 - **Coverage.** Every byte of an input is expected to influence the hash value. We remark that this property is formulated in a statistical way. It means that given a certain byte of the input the probability that this byte does not influence the input’s digest is insignificant. Otherwise it is possible that small changes will be uncovered. This property is in conformance with the corresponding characteristic of classical hash functions.
- P5 - **Obfuscation Resistance.** It is the difficulty to achieve a false negative / non-match. For instance, let f be a file e.g., a suspect file. Then it should be difficult to manipulate f to f' so that a comparison yield a non-match but they are still very similar.

³ In most cases the comparison of similarity preserving hash values is more complex than for traditional hashes where we can use the Hamming distance.

2.2 Use Cases

This section demonstrates that within the area of Similarity Preserving Hashing both mentioned security properties are sufficient. Assuming the applications computer forensics, malware or junk mail detection, which are reasonable in our mind, we identified two common aspects: *file identification* and *fragment detection*, which are explained in the following.

File Identification. The mentioned applications mostly use databases containing hash values of known inputs e.g., it stores fingerprints of known malware or files from previous investigations. Later on, if the application is faced with an unknown input, it generates the fingerprint and performs database lookups. Depending on the underlying database, this processing categorizes files into the categories: known-to-be-good, known-to-be-bad and unknown input.

Blacklisting. The main challenge for an active adversary is to hide suspect (=known-to-be-bad) files from an automatic identification through a 3rd party e.g., investigators, anti-virus software or junk mail scanner. As this is easily feasible for cryptographic Hash Functions by flipping a single bit, it should not be possible within the area of SPH. This concludes *P5 - obfuscation resistance*.

Whitelisting. Within the area of whitelisting we believe that cryptographic Hash Functions are the mean of choice. For instance, an active adversary is able to manipulate the *ssh daemon* of an operation system and include a backdoor. Thus, the original file and the modified file are still very similar although it is a malicious ssh daemon.

As whitelisting is out of scope we argue that traditional security properties like preimage-resistance, second preimage-resistance and collision resistance are not necessary for SPH - no one likes to manipulate a file to look like a suspect file.

Fragment Detection. Another opportunity for SPH on the binary level is its ability to identify file fragments e.g., 200kiB out of 1MiB. One possible scenario is the computer forensics. For instance, an investigator receives a hard disk which is formatted in quick-mode. Thus he is only able to analyze the low level hdd blocks. If a match is identified, a known-to-be-bad files were present before the deletion. In the best case he even may recover the file.

2.3 Bloom Filters and the Comparison Function

A very promising way to represent hash values for SPH are Bloom filters because they allow a fast comparison using the Hamming distance. According to this, we briefly describe Bloom filters in general followed by a possible *comparison function* (CP) as mentioned in Sec. 2.

Bloom Filters. A Bloom filter is an array of m bits (all set to zero) and used to represent a set S of n elements. In order to ‘insert’ an element s into the filter, k independent Hash Functions are used where each Hash Function outputs a value

between 0 and $m-1$. For instance, to insert s we compute $h_0(s), h_1(s), \dots, h_{k-1}(s)$ where each h outputs a value between 0 and $m-1$. Thus, each Hash Function sets the corresponding bit within the Bloom filter.

To answer the question if s' is in S , we compute $h_0(s'), h_1(s'), \dots, h_{k-1}(s')$ and look if the bits at the corresponding positions are set to one. If all bits are set to one, s' is assumed to be within S with a high probability. Otherwise, if at least one bit is set to zero, we know that s' is not within S .

Comparison Function for Bloom Filters. This paragraph explains Roussev's idea ([3,6]) for a CP. The proceeding how to obtain a set S out of the input is explained later in Sec. 4.1. Hence, the rest of this section only explains how we compare Bloom filters.

Let bf, bf' be two Bloom filters, let $|bf|$ denote the number of bits set to one within a Bloom filter and let e be the amount of bits in common ($e = |bf \cap bf'|$). To define the similarity of two Bloom filters, we have to make some assumptions of the minimum and maximum overlapping bits by chance wherefore Roussev introduces a cutoff point C . If $e \leq C$, then the similarity score is set to zero.

C is determined as follows

$$C = \alpha \cdot (E_{max} - E_{min}) + E_{min} \quad (1)$$

where α is set to 0.3⁴, E_{min} is the minimum number of overlapping bits due to chance and E_{max} the maximum number of possible overlapping bits. Thus E_{max} is defined as

$$E_{max} = \min(|bf|, |bf'|). \quad (2)$$

As described in Sec. 2.3, k denotes the amount of hash functions and m the size of a Bloom filter in bits. Furthermore, let \bar{bf} denote the amount of elements within a Bloom filter and $p = 1 - 1/m$ the probability that a certain bit isn't set to one when inserting a bit. Thus

$$E_{min} = m \cdot (1 - p^{k \cdot \bar{bf}} - p^{k \cdot \bar{bf}'} + p^{k \cdot (\bar{bf} + \bar{bf}')}) \quad (3)$$

is an estimation of the amount of expected common bits set to one in the two Bloom filters bf, bf' by chance. In order to receive a similarity score we use

$$SF_{score}(bf, bf') = \begin{cases} 0, & \text{if } e \leq C \\ \lceil 100 \frac{e-C}{E_{max}-C} \rceil, & \text{otherwise.} \end{cases} \quad (4)$$

Due to different file sizes, it might be possible that $|S|$ is very large and all bits within bf ⁵ would be set to one. To overcome this issue, we create a new Bloom filter if $\bar{bf} = BF_{max}$. Hence, the final hash value is not a single but a list of Bloom filters. If we'd like to compare them, it is an all-against-all comparison of Bloom filter sequences.

⁴ This is done by best practice.

⁵ The size m of a Bloom filter is fixed.

Let $SD_1 = \{bf_1, bf_2, \dots, bf_s\}$ and $SD_2 = \{bf'_1, bf'_2, \dots, bf'_r\}$ the Bloom filter sequences (hash values) of two inputs and $s \leq r$. If $\overline{bf_1} < 6$ or $\overline{bf'_1} < 6$ then the original input does not contain enough features and the similarity score is -1 , not comparable. Otherwise the similarity score is the mean value of the best matches of an all-against-all comparison of the Bloom filters, formally defined as

$$SD_{score}(SD_1, SD_2) = \frac{1}{s} \sum_{i=1}^s \max_{1 \leq j \leq r} SF_{score}(bf_i, bf'_j). \quad (5)$$

3 Related Work

The beginning of similarity preserving hashing was in 2002 by Harbour who developed `dcfldd`⁶ which extends the well-known disk dump tool `dd`. `dcfldd` is also called *block based hashing* as it divides an input into fixed-size blocks, hash each block separately and concatenate all hash values. In order to overcome this approach it is sufficient to insert / remove one byte in the beginning. Thus the offset of each block shifts and the resulting hash value is completely different.

Context triggered piecewise hashing (abbreviated CTPH) can be considered as an advancement of `dcfldd` which fixes the alignment weakness. It was presented in [2] by Kornblum in 2006 and is based on a spam detection algorithm of [7]. The basic idea is equal to the aforementioned block based hashing but instead of dividing an input into blocks of a fixed length, an input is divided based on the current context of 7 bytes.

As CTPH was the first Approach for Similarity Preserving Hashing, it was improved in the upcoming years by [8,5,9,10] with respect to both, efficiency and security. In 2011 [11,12] did a security analysis of CTPH where the authors focused on blacklisting and whitelisting and came to the conclusion that `ssdeep` fails in case of an active adversary.

Similarity Digest Hashing is a completely different Approach for Similarity Preserving Hashing and was presented in 2010 by Roussev ([3]) including a prototype called `sdhash`. Instead of dividing an input into pieces, `sdhash` identifies “statistically-improbable features” ([13]) using an entropy calculation.

These characteristic features, a sequence of length 64 bytes, are then hashed using the cryptographic Hash Function SHA-1 ([14]) and inserted into a Bloom filter ([15]). Hence, files are similar if they share identical features.

Comparison [16] provides a comparison of `ssdeep` and `sdhash` and shows that the latter “approach significantly outperforms in terms of recall and precision in all tested scenarios and demonstrates robust and scalable behavior”. A security analysis ([17]) approved this statement but also showed some peculiarities and weaknesses of `sdhash`.

⁶ <http://dcfldd.sourceforge.net>; visited 02.05.2012

4 Multi-Resolution Similarity Hashing (MRSH)

Roussev et al. ([5]) present a powerful variation of `ssdeep` called *multi-resolution similarity hashing* (abbreviated MRSH) that slid into obscurity. Therefore Sec. 4.1 explains the concept of the original algorithm and Sec. 4.2 shows changes to increase the performance.

4.1 Foundations of MRSH

As briefly described within Sec. 3 the main idea of `ssdeep` is to divide an input in several chunks based on the current context of 7 bytes where an input is a byte sequence. As MRSH is based on `ssdeep`, this algorithm is explained first.

Let an input IN of length L be given as a byte sequence $b_0b_1 \dots b_{L-1}$. In order to identify the end of a chunk (i.e., to divide the input into blocks), `ssdeep` uses a window of size 7 bytes that moves through the whole input, byte for byte. At each position p ($0 \leq p < L$) within IN the window contains a byte sequence $BS_p = b_{p-6}b_{p-5} \dots b_p$ which serves as input for a pseudo random function PRF . We denote this by $PRF(BS_p)$. If $PRF(BS_p)$ hits a certain value, the end of the current chunk is identified and b_p is called a *trigger point*. The subsequent chunk starts at byte b_{p+1} and ends at the next trigger point or EOF.

In order to define a hit for $PRF(BS_p)$, MRSH uses a fixed modulus called *blocksize* b e.g., 256⁷. Thus, if $PRF(BS_p) \equiv -1 \pmod{b}$ then b_p is a trigger point and the algorithm identified the end of the chunk. If PRF outputs equally distributed values, the probability of a hit is reciprocally proportional to b and therefore the average chunk size should be b bytes⁸.

In contrast to `ssdeep` which uses an algorithm called `rolling_hash`⁹ for PRF , MRSH uses the polynomial Hash Function `djb2`¹⁰ over the 7 byte window which is shown in Algorithm 1. For each window (at each position p within IN) the window needs to be computed.

Algorithm 1. Polynomial Hash Function `djb2`

```

unsigned long hash = 5381
int c
for i = 0 → 6 do                ▷ Run through all bytes within the window
    c = BS[i]
    hash = ((hash << 5) + hash) + c;           ▷ 33 · hash + c
end for
return hash

```

⁷ `ssdeep` used a variable modulus based on the file size.

⁸ Therefore this modulus is called *blocksize*.

⁹ This function is a variation of Adler-32; <http://en.wikipedia.org/wiki/Adler-32>; visited 04.06.2012

¹⁰ <http://www.cse.yorku.ca/~oz/hash.html>; visited 21.05.2012

The biggest difference between `ssdeep` and MRSH is the hash value representation. `ssdeep` uses the non-cryptographic Hash Function FNV ([18]) to hash each chunk. For each chunk it uses the least significant 6 bits of the FNV hash and concatenates all of them. Thus the final hash value is a Base64 sequence.

MRSH works completely different. All identified chunks build the set S which is used as basis for the hash value generation using Bloom filters (see Sec. 2.3). Instead of using k different Hash Functions, MRSH “take[s] the MD5 hash and split[s] it into four 32-bit numbers and take[s] the least significant 11 bits from each part” ([5]). For instance, imagine the least significant 11 bits are 010 1000 1010 = $0x28A = 650$, thus the bit at position 650 within the Bloom filter is set to one. Having 4 sub-hashes, each chunk sets 4 bits within the Bloom filter. After inserting $BF_{max}(=256)$ chunks into a Bloom filter, it reaches its maximum and a new filter is created. Hence, the final hash value is a list of Bloom filters.

As stated before, b is the approximate length of a chunk. In comparison to `ssdeep`, MRSH uses a *minimum chunk size* which is $\frac{1}{4}$ of the chunk size b . Thus, whenever a trigger point is discovered the next $\frac{b}{4}$ bytes are skipped for *PRF*, so the chunk is guaranteed a minimum size of $\frac{b}{4}$.

4.2 MRSH Version 2

In the following we present an updated version of MRSH called `MRSH-v2`. Generally speaking `MRSH-v2` uses its precursor as a base frame but with some accommodations based on the aforementioned properties.

PRF. We impose two important requirements on a pseudo random function (PRF). First, it has to be very efficient with respect to its computation time as it is invoked for roughly every byte of the input. Second its output should behave pseudo randomly.

In his version of MRSH Roussev changed the PRF from `rolling_hash` to `djb2` which he motivates with respect to performance. As shown in Algorithm 1, `djb2` should be quite fast. However, our tests presented in Sec. 5.2 show different results. The point is, although `djb2` looks less complex, it needs to compute the hash value over the whole window at each time (7 loops per window) whereas the original version (`rolling_hash`) is able to remove the last byte and add the new one to the hash value (only one loop per window).

[5, Sec. 3] compares the randomness of `djb2` with MD5 and concludes that `djb2` totally fulfills the expectations of a fast PRF. However, [11, Sec. V] shows that `rolling_hash` is suitable for `MRSH-v2`, too.

As outcome of both requirements we decided to make use of the original rolling hash as PRF in our algorithm `MRSH-v2`.

Chunk Hash Function. The motivation to change the chunk Hash Function from FNV to MD5 is that “FNV is not a collision-resistant function and has some known collision issues [...] especially for inputs with lower entropy which would present a serious problem for simple hashes” ([5]).

The latter argument is in contrast to [18] where it says that “the high dispersion of the FNV hashes makes them well suited for hashing nearly identical strings”. Furthermore we argue that collision resistance is not necessary as discussed in Sec. 2.1. Moreover MRSH reduces the MD5 hash value from 128 bits to 44 bits in order to insert it into the Bloom filter. Thus, the hash loses its cryptographic properties.

Due to these facts our version uses FNV-1a (64 bit) and is therefore faster (some measurement results are given in Sec. 5.1).

Minimum Chunk Size. A minimum chunk size comes with two improvements. First, it overcomes one of the main attacks on `ssdeep` presented in [11] called ‘adding trigger points’. Second, it increases the performance as the PRF needs not to be computed at each offset within the input sequence. A drawback is that some details may be lost. This is the case if two subsequent trigger points have a distance of at most $\frac{b}{4} - 1$.

We illustrate this characteristics on base of an extreme example. We assume that the input byte sequence has a trigger point every $(\frac{b}{4} - 1)$ -th byte. They are denoted by t_0, t_1, t_2, \dots . Then every second trigger point is skipped (only trigger points with an even index are used). Removing the first trigger point t_0 from the input results in considering the trigger points t_1, t_3, \dots yielding a fundamental different hash value.

However, for performance reasons we agree on the same minimum chunk size as used in MRSH, $\frac{b}{4}$.

Bloom Filters. MRSH uses Bloom filters of size $m = 2048$ and inserted $BF_{max} = 256$ chunks each setting 4 bits within the Bloom filter which is in contrast to our implementation. Within MRSH a maximum of 1024 bits could be set and each Bloom filter could represent approximately 65,536 byte (using the blocksize $b = 256$).

The final hash value of MRSH-v2 is mostly based on the settings identified for `sdhash` in [3,6]. Thus the Bloom filter size is still $m = 2048$ bits but we changed $BF_{max} = 256$ and $k = 5$ (five sub-hashes). The maximum is therefore 800 bits and one Bloom filter could represent approximately 40,960 byte (more see Sec. 5.1). Also MRSH has a better compression, MRSH-v2 has a better false positive rate as shown in Sec. 5.6.

In order to insert the chunk hash value into a Bloom filter, we use the least significant $k \cdot \log_2(m)$ bits (MRSH divides the chunk hash values). As a consequence our chunk Hash Function needs at least so many bits which is fulfilled by FNV-1a using the default setting $k = 5$, $m = 2048$. A performant proceeding is given in Algorithm 2. In addition the design of MRSH-v2 allows to change the parameters like k, m or the chunk Hash Function.

5 Experimental Results and Evaluation

The following sections discuss the properties from Sec. 2.1 with respect to our algorithm. Furthermore we compare MRSH-v2 to `sdhash`, `bbHash` and `ssdeep`.

Algorithm 2. Insertion of a chunk hash into a Bloom filter h is the chunk hash value $k = 5$

▷ Amount of sub-hashes

 $m = 0x7FF$ ▷ $m = 2048 - 1$

shiftOps = 11

▷ Calculated by $\log_2(m + 1)$ **for** $i = 0 \rightarrow k - 1$ **do**▷ Create k sub hashesbit = $(h \gg (\text{shiftOps} \cdot i)) \& m$

setBitInBloomFilter(bit)

end for

All experimental tests were performed on a 64Bit Mac OS X with a 2.4 GHz Intel Core 2 Duo processor.

5.1 P1 - Compression

Due to the design of `ssdeep` and thus of `MRSH-v2`, the hash value length depends on the blocksize b , the amount of chunks per Bloom filter BF_{max} and the size of a Bloom filter m (in bits). Each Bloom filter represents approximately $BF_{max} \cdot b$ bytes of a given input and thus the compression ratio is $\frac{m}{8} \cdot \frac{1}{BF_{max} \cdot b}$.

As discussed in Sec. 4.2 we use $m = 2048$ bits and $BF_{max} = 160$. Assuming these values, the compression ratio is $\frac{2048}{8} \cdot \frac{1}{160 \cdot b} = \frac{8}{5 \cdot b}$ for $b > 0$ and therefore adjustable by changing b . Table 1 shows the proportion between blocksize b and the expected hash value length. For instance, by default we set $b = 320$ and thus the compression ratio is at 0.5%.

Table 1. Proportion between blocksize b and the hash value length in percent

b	128	160	256	320	512
expected length in %	1.250	1.000	0.625	0.500	0.313

`ssdeep` produces outputs having at most 100 Base64 characters. This rather good compression implies a security drawback as discussed in [11]. Put simply, if there are too many chunks, the last chunks are combined into one large one. Due to the poor result in the security analysis we neglect `ssdeep` and focus on two other approaches.

The hash values of `bbHash` and `sdhash` are proportional to the input length, where the proportionality factor is 0.5% and 3.3%, respectively. However, the performance of `bbHash` isn't acceptable wherefore we come up with the following classification: 1. `MRSH-v2` (0.5%), 2. `sdhash` (3.3%), 3. `bbHash` (0.5%) and `ssdeep`.

5.2 P2 - Ease of Computation

This section is roughly divided into two parts. First, we analyze MRSH-v2 itself as the performance of MRSH-v2 is based on two issues: the pseudo random function (PRF) and the chunk hash function. Second, we compared our implementation against other existing algorithms.

All tests are based on a 500MiB file from `/dev/urandom`.

PRF. In the following we show that the `rolling_hash` is faster than `djb2`. In order to test both algorithms we separated them, run them ‘stand-alone’ and used all optimizations modes of the `gcc` compiler¹¹. Of course, both versions are improved for performance, e.g., the struct of the `rolling_hash` from `ssdeep` was removed. The result is given in Table 2¹².

Table 2. Performance of two possible pseudo random function (PRF)

optimization mode	-	O1	O2	O3
djb2	23.620s	11.021s	1.236s	1.241s
rolling hash	9.835s	4.315s	1.138s	1.085s
djb2 / rolling hash	2.402	2.554	1.086	1.143

Actually we cannot explain these serious differences. We recognized them by comparing

- `djb2` (8.532s) and `rolling_hash` (3.808s) within MRSH-v2 and
- `djb2` (1.241s) and `rolling_hash` (1.085s) as ‘stand-alone’.

Chunk Hash Function. As discussed in Sec. 4.2 we decided for FNV-1a instead of the cryptographic Hash Function MD5¹³. To test MD5 we took a library provided by OpenSSL and used an optimized version of FNV-1a. The test is focused on the algorithm time (read-in time is neglected) and solved using the `clock()`-function from C++. The result is 10^{-6} s from FNV vs. 1.354s of MD5. Using these functions within MRSH-v2 it is 5.235s vs. 6.569s.

The second part of this section is the comparison against other existing algorithms. We skipped `bbHash` as its performance is not acceptable and focused on `ssdeep` and `sdbhash`. Furthermore we also included SHA-1 as a reference time. All times were measured using the `time` command and the algorithm CPU-time (`time` denotes this by `user-time`). The results are shown in Table 3.

As expected SHA-1 outperforms every similarity preserving Hash Function and `sdbhash` is the slowest one due to the high complexity. The difference between MRSH-v2 and `ssdeep` relies on the *minimum chunk size* which allows to skip some calculations and an improved implementation of the `rolling_hash`.

¹¹ <http://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html>; visited 21.6.2012

¹² We do not measure the time it takes to read the file into a buffer.

¹³ Due to the minimum hash value length of 55 bits (see Sec. 4.2, Bloom filters) it is not possible to use `djb2` which would be even faster.

¹⁴ We used `sdbhash` with 2 threads.

Table 3. Performance comparison of similarity preserving Hash Functions and SHA-1

	SHA-1	MRSH-v2	sdhash 2.0	sdhash 2.0 ¹⁴	ssdeep 2.8
runtime	2.549s	5.235s	28.641s	28.493s	7.131s
algorithm / SHA-1	1.000	2.054	11.236	11.178	2.798

5.3 P3 - Similarity Score

Our algorithm MRSH-v2 makes use of the Bloom filter comparison algorithm from Sec. 2.3 and findings from [17, Sec. 4.3] including an own improvement. In a first step we show that the existing algorithm is well suited for fragment detection, but has drawbacks for file similarity detection. As a result we recommend to use the original comparison function of `sdhash` for fragment detection. However, we modify the algorithm to decide about file similarity, too.

Fragment Detection. We discussed the original comparison algorithm in Sec. 2.3. We explain its shortcomings in what follows based on an example (a generalization is easy). As a result MRSH-v2 makes use of this comparison algorithm for fragment detection only.

Let f and f' be two files where f' is a fragment of f , e.g. the first 25% of f . Let $SD = \{bf_1, bf_2\}$ be the hash value of f and let $SD' = \{bf_1^*\}$ be the hash value of f' where $|bf_1^*| < |bf_1|$.

To receive the similarity score we first have to identify the best matching Bloom filters where the filter similarity is identified by Eq. (4). Recall,

$$SF_{score}(bf, bf') = \begin{cases} 0, & \text{if } e \leq C \\ \lceil 100 \frac{e-C}{E_{max}-C} \rceil, & \text{otherwise.} \end{cases} \quad (4)$$

In case of fragments we have $e = E_{max}$ as $e = |bf \cap bf'| = |bf'|$ and $E_{max} = \min(|bf|, |bf'|) = |bf'|$. Thus the $SF_{score}(bf_1^*, bf_1) = 100$.

Knowing the best matching Bloom filters, the final similarity score is generated using Eq. (5). Recall,

$$SD_{score}(SD_1, SD_2) = \frac{1}{s} \sum_{i=1}^s \max_{1 \leq j \leq r} SF_{score}(bf_i, bf_j'), \quad (5)$$

where $s = |SD'|$, $r = |SD|$ (s needs to be smaller). As $s = 1$, the SD_{score} is $\frac{100}{1}$. To sum it up, we compared two obviously different hash values and resulted in a 100% match score. As f' is a fragment of f this algorithm is perfect for *fragment detection*.

File Similarity Detection. Besides fragments we also interested in identifying similar files. Taking the aforementioned example concerning f, f' we expect a file similarity of 25%, if $|bf_1|, |bf_2|$ are at their maximum and $|bf_1^*| = \frac{bf_1}{2}$. In order to achieve file similarity, there are two adaptations:

1. As proposed in [17] MRSB-v2 makes use of a new function $E'_{max} = \max(|bf|, |bf'|)$ in Eq. (4) (the min function is replaced by the max function).
2. Additionally we replace $\frac{1}{s}$ by $\frac{1}{r}$ within Eq. (5). As $s \leq r$ all Bloom filters are considered (in contrast to [17], where only the first s Bloom filters are relevant).

To receive the final similarity score, we first have to generate the SF_{score} defined by $SF_{score} = \lceil 100 \frac{e-C}{E'_{max}-C} \rceil$. Thus we need to determine e , E'_{max} and C where

- $e = |bf_1 \cap bf_1^*| = |bf_1^*|$,
- $E'_{max} = \max(|bf_1|, |bf_1^*|) = |bf_1|$ and
- $C = \alpha \cdot (E_{max} - E_{min}) + E_{min}$ where
 - $E_{max} = \min(|bf_1|, |bf_1^*|) = |bf_1^*|$ and
 - $E_{min} = m \cdot (1 - p^{k \cdot \overline{bf_1}} - p^{k \cdot \overline{bf_1^*}} + p^{k \cdot (\overline{bf_1} + \overline{bf_1^*})}) = 117.548$

We first estimate the amount of bits set to be one using the following

$$|bf_1| = m \cdot \left(1 - \left(1 - \frac{1}{m} \right)^{k \cdot |bf_1|} \right) = 2048 \cdot (1 - 0.99951172^{5 \cdot 160}) = 662.386$$

$$|bf_1^*| = m \cdot \left(1 - \left(1 - \frac{1}{m} \right)^{k \cdot |bf_1^*|} \right) = 2048 \cdot (1 - 0.99951172^{5 \cdot 80}) = 363.442$$

and calculate C by

$$C = 0.3(363.442 - 117.548) + 117.548 = 191.316$$

To sum it up, we result in

$$SF_{score} = \lceil 100 \frac{e-C}{E'_{max}-C} \rceil = 100 \cdot \frac{363.442 - 191.316}{662.386 - 191.316} = 100 \cdot \frac{172.126}{471.0698} = 36.539.$$

In the very last step we use the adopted version Eq. (5) (instead of $\frac{1}{s}$ we use $\frac{1}{r}$). Thus we have to divide SF_{score} by 2 and result in a final similarity score of 18.270.

Implementation. These properties allow to extend our algorithm to have two modes as listed in Fig. 1.

Regular mode (default setting) is used to identify the similarity between two files.

Fragment mode (use `-f` option) is the fragment mode and used to find smaller parts of a file.

```

$ dd if=/dev/urandom of=2MiB bs=1m count=2
$ split -b 512k 2MiB

$ ./mrsh-v2 2MiB xaa
Similarity of files 2MiB and xaa is: 27.113835

$ ./mrsh-v2 -f 2MiB xaa
Similarity of files 2MiB and xaa is: 99.417396

```

Fig. 1. A sample for fragment and similar file detection

5.4 P4 - Coverage

Full coverage means that every byte of an input should influence the output. By design all bytes influence the final hash value and therefore especially some greater random changes influence the final hash value. Recall, a high similarity score (e.g., 100) means that two inputs are very similar but it does not imply that they are completely identical.

`ssdeep` and `MRSH-v2` have a better (full) coverage compared to `sdhash`, as [17] shows that there are bytes which don't influence the similarity digest at all. `bbHash` claims to have a full coverage but this is not attest.

5.5 P5 - Obfuscation Resistance

Obfuscation resistance is the difficulty to achieve a non-match. Thus we roughly analyze the amount of changes an active adversary has to do in order to overcome this approach. However, this section does not replace a comprehensive security analysis.

The most obvious attack is to change one byte within each chunk which will change all chunk hash values. Recall, the chunk size in bytes is approximately the blocksize b . Let $b = 320$. Assuming a file of 1048576 bytes (=1 MiB), this result in $\frac{1,048,576}{320} = 3276.8$ changes. Due to the comparison algorithm it is not necessary to have changes within each chunk.

[17] showed the possibility of 'Bloom filter shifts'. It is possible to reduce the similarity score down to approximately 25 by inserting data at the beginning of a file. However, the authors also present a first idea to solve this issue which will be analyzed for the next upcoming version of `MRSH-v2`.

Nevertheless the possibility to make changes within a specific file depends on the file type. Generally we classify files in one of the following categories.

Locally sensitive file types (e.g., jpg, pdf, zip, exe) nearly impossible to manipulate at each position (e.g., [11] showed that the jpg-header allows changes).

A flipped bit can have 'global' consequences such that the file is not readable

anymore. We believe that an active adversary will not overcome MRSH-v2 for these kind of types¹⁵.

Locally non sensitive file types (e.g., txt, doc, bmp) are mostly small (e.g., doc, txt) and sometimes not so wide-spread (e.g., bmp). Manipulations only influence the local area e.g., changing a letter within a txt file. For small files, reducing b increases granularity of the hash value and force an attacker to do more changes. Of course there are also large doc-files but they mostly contain images (which give them their unique characteristic).

5.6 False Positive Rate

Within 4.2 we explained that we have to find a good trade-off between compression and false positive rate. Due to the changes from $[k = 4, BF_{max} = 256]$ to $[k = 5, BF_{max} = 160]$ we reduced the false positive rate

$$\left(1 - \left(1 - \frac{1}{m}\right)^{k \cdot BF_{max}}\right)^k = \left(1 - \left(1 - \frac{1}{2048}\right)^{4 \cdot 256}\right)^4 = 0.0240 \quad (6)$$

down to

$$\left(1 - \left(1 - \frac{1}{m}\right)^{k \cdot BF_{max}}\right)^k = \left(1 - \left(1 - \frac{1}{2048}\right)^{5 \cdot 160}\right)^5 = 0.0035 . \quad (7)$$

which is a factor of approximately 7.

6 Conclusion

Currently there are no constant naming, definition or properties for Similarity Preserving Hashing which are necessary to classify them. But due to the increasing amount of data, it is necessary to rate different approaches. Thus this paper at hand presents 5 properties: 3 general properties and 2 security related properties. As a conclusion, the identified properties coincide only partially with traditional Hash Functions which comes due to the different use cases.

Additionally we improved an existing Approach for Similarity Preserving Hashing from 2007 with respect to performance and introduced MRSH-v2. We briefly compared it against other algorithms based on the properties. As a result MRSH-v2 outperforms existing algorithms with respect to performance. The hash value length is at 0.5% and only surpassed by `ssdeep` which failed a security analysis. As a highlight MRSH-v2 is the first algorithm that has two modes: fragment detection and similar file detection.

There are three next steps: First, we like to complete the functions of our implementation (e.g., read directory) . Second, a detailed security analysis of MRSH-v2 is needed. And third, we would like to implement `sdhash` using FNV and analyze the performance.

¹⁵ We focused on the binary level and not on a semantic level where it is possible to rotate an image.

Acknowledgments. This work was partly funded by the EU (integrated project FIDELITY, grant number 284862) and supported by CASED (Center for Advanced Security Research Darmstadt).

We thank Mustafa Karabat for supporting us with programming and testing.

References

1. NIST, “National Software Reference Library” (May 2012),
<http://www.nsr1.nist.gov>
2. Kornblum, J.: Identifying almost identical files using context triggered piecewise hashing. In: Digital Forensic Research Workshop (DFRWS), vol. 3S, pp. 91–97 (2006)
3. Roussev, V.: Data fingerprinting with similarity digests. In: Chow, K.-P., Sheno, S. (eds.) *Advances in Digital Forensics VI. IFIP AICT*, vol. 337, pp. 207–226. Springer, Heidelberg (2010)
4. Breitinger, F., Baier, H.: A Fuzzy Hashing Approach based on Random Sequences and Hamming Distance. In: ADFSL Conference on Digital Forensics, Security and Law, pp. 89–101 (May 2012)
5. Roussev, V., Richard, G.G., Marziale, L.: Multi-resolution similarity hashing. In: Digital Forensic Research Workshop (DFRWS), pp. 105–113 (2007)
6. Roussev, V.: Scalable data correlation. International Conference on Digital Forensics (IFIP WG 11.9) (January 2012)
7. Tridgell, A.: Spamsun. Readme (2002),
<http://samba.org/ftp/unpacked/junkcode/spamsun/README>
8. Chen, L., Wang, G.: An Efficient Piecewise Hashing Method for Computer Forensics. In: Workshop on Knowledge Discovery and Data Mining, pp. 635–638 (2008)
9. Seo, K., Lim, K., Choi, J., Chang, K., Lee, S.: Detecting Similar Files Based on Hash and Statistical Analysis for Digital Forensic Investigation. In: Computer Science and its Applications (CSA 2009), pp. 1–6 (December 2009)
10. Breitinger, F., Baier, H.: Performance Issues About Context-Triggered Piecewise Hashing. In: Gladyshev, P., Rogers, M.K. (eds.) *ICDF2C 2011. LNICST*, vol. 88, pp. 141–155. Springer, Heidelberg (2012)
11. Baier, H., Breitinger, F.: Security Aspects of Piecewise Hashing in Computer Forensics. In: *IT Security Incident Management & IT Forensics (IMF)*, 21–36 (May 2011)
12. Breitinger, F.: Security Aspects of Fuzzy Hashing. Master’s thesis, Hochschule Darmstadt (February 2011), <https://www.dasec.h-da.de/offerings/theses/>
13. Roussev, V.: Building a Better Similarity Trap with Statistically Improbable Features. In: 42nd Hawaii International Conference on System Sciences, pp. 1–10 (2009)
14. SHS, “Secure Hash Standard” (1995)
15. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM* 13, 422–426 (1970)
16. Roussev, V.: An evaluation of forensic similarity hashes. In: Digital Forensic Research Workshop, vol. 8, pp. 34–41 (2011)
17. Breitinger, F., Baier, H., Beckingham, J.: Security and Implementation Analysis of the Similarity Digest sdhash. In: First International Baltic Conference on Network Security & Forensics (NeSeFo) (August 2012)
18. Noll, L.C.: Fowler / Noll / Vo (FNV) Hash (2001),
<http://www.isthe.com/chongo/tech/comp/fnv/index.html>

Investigating File Encrypted Material Using NTFS \$logfile

Niall McGrath and Pavel Gladyshev

Digital Forensic Investigation Research Group
University College Dublin

Abstract. When an encrypted file is discovered during a digital investigation and the investigator cannot decrypt the file then s/he is faced with the problem of how to determine evidential value from it. This research is proposing a methodology for locating the original plaintext file that was encrypted on a hard disk drive. The technique also incorporates a method of determining the associated plaintext contents of the encrypted file. This is achieved by characterising the file-encryption process as a series of file I/O operations and correlating those operations with the corresponding events in the NTFS \$logfile file. The occurrence of these events has been modelled and generalised to investigate file-encryption. This resulted in the automated analysis of \$logfile in *FindTheFile* software.

Keywords: NTFS \$logfile file, MAC Times, Encryption.

1 Introduction

Law enforcement agencies (LEA) encounter encryption in relation to many crimes. The distribution of illegal material [1] [2] is an example of the many offences associated with file encryption. The use of encryption in general has been cited [3] as a major hurdle in digital investigations. When file-encrypted material is investigated and the file cannot be decrypted, cracked nor bruteforced; there is no formal method or technique to extract evidential value. As a result this research presents a methodology which identifies the original plaintext filename that was encrypted, while also displaying the plaintext contents of the file. This is irrespective of the file being “deleted” or not. A typical scenario that occurs is where an encrypted bundle is transmitted to a buyer or intended recipient of illegal material. Encryption software has a common feature of giving the option of deleting the original plaintext file after encryption. This naturally increases the complexity of a digital investigation but does not restrict it; how to recover deleted files is outlined in [4]. The NTFS \$logfile file (\$logfile) is the fundamental evidence artefact upon which the proposed methodology here is based on.

1.1 Problem Description

The main problem with investigating encrypted material is not being able to establish an evidential link between the encrypted file and the original plaintext file and also

not being able to view the plaintext contents. The approach taken here to solve the problem is to observe the process of encryption and then characterise the sequence of events. Extracting event information from \$logfile is central to the approach. This leads to formulating a methodology, where it is modelled, generalised, automated and then applied formally in a case-study.

1.2 Related Work

Cryptopometry methodology can only be used to investigate illegal material when it is encrypted and exchanged using public-private key encryption like OpenPGP or X.509 [5]. *Cryptopometry* also does not reveal the plaintext contents of the encrypted file. It is however elaborated on how the computer forensic investigator can use *Cryptopometry* to identify encrypted material, examine it and extract evidential value from it in [5]. Typically this scenario is where a distributor encrypts the illegal material and posts it into a newsgroup or interest group via anonymous re-mailer or via an instant messenger system. The accomplice who is subscribed to that group receives encrypted material and can decrypt it. The anonymity of all involved parties is preserved and the content cannot be decrypted by bystanders [5].

2 Background Information

In order for an application to encrypt a file's contents the underlying actions that take place can be categorised according to 1) type of I/O event i.e. Read or Write, 2) the processes and the sequence of threads that govern execution and 3) where in the stack does this executable file get called and in what mode i.e. user or kernel. In addition there are numerous NTFS design goals outlined in [6] but the specific components that are of interest in this paper are: \$logfile and how it is updated by the Log file service and Master File Table (MFT). ObjectId and how the Distributed Link Tracking (DLT) service and how they facilitate forensic examinations are also of interest.

2.1 I/O File Processing

The steps of the I/O file open process along with the principles of I/O request packet (IRP) processing are detailed in [6]. It is outlined that the runtime library function calls the CreateFile function, and then the kernel32.dll-windows subsystem is called which in turn calls the native NtFileCreate function in Ntdll.dll. The transition into kernel mode (where NtCreateFile in Ntoskrnl.exe is called) and the subsequent commands to the Object Manager and the I/O manager and finally the transition back to user mode are listed.

2.2 NTFS

The journal file for the windows operating system is called \$logfile. The \$logfile is used to recover from system crashes and unexpected conditions. It has the standard

file attributes and stores the log data in the \$DATA attribute. The file is organised into 4,096 byte pages consisting of two parts: the restart and the logging area. The restart area contains information on how to start the recovery after a system failure [7]. There are two types of information recorded here. These are “Redo” and “Undo” information. Redo information is how to reapply one sub-operation of a fully logged (“committed”) transaction to the volume if a system failure occurs before the transaction is flushed from the cache. Undo information is how to reverse one sub-operation of a transaction that was only partially logged (“not committed”) at the time of a system failure [9]. These “Redo” and “Undo” operation codes are used in a composite manner to form the series of log records that are written to the \$logfile when a file operation is performed. The hexadecimal (0x) composite operation codes are used such as 0x0E/0x0F, 0x02/0x00, 0x08/0x00, and 0x14/0x14 for file creation, delete, extending, truncation, information setting and renaming [9]. NTFS guarantees that the transaction will appear on the volume, even if the operating system subsequently fails. A table of values that represent update records for each of the following transactions is presented in [9]: Initializing (0x02), de-allocating (0x03) file record segments, writing the end of file record segments (0x04), creating (0x05) and deleting (0x06) attributes, updating resident (0x07) and non-resident (0x08), setting attribute sizes (0x0B) adding (0x0E) and deleting (0x0F) index entry allocation and setting (0x15) and clearing bits in \$bitmap (0x16).

In NTFS the primary data structure is the MFT and every file will have at least one entry in the MFT. The MFT holds information about the files and directories in MFT entries. These MFT entries store attributes; where an attribute is a data structure containing a specific type of information such as a file's filename. NTFS take the form of reading and writing attributes for a given file e.g. the \$DATA attribute which is common to every file in the file system [6]. The \$STANDARD_INFORMATION attribute contains the timestamp information for each file. This attribute determines the MAC times for a file when the properties of a file are viewed. There is also a \$FILE_NAME attribute that contains the MAC time information as it relates to the filename for a given file. Link files are created when a file is opened [8]. Also an ObjectID is described as an attribute that uniquely identifies a file or directory on a volume. This is listed as the location of a file at some point in time; it is made up of a VolumeID and an ObjectID. The ObjectID of a file can be queried using the command line tool *fsutil* [7].

3 Observation towards a Framework

Using three different encryption packages a file was encrypted. The encryption process was monitored using *Process Monitor*. *Process Monitor* is a system activity monitoring tool which monitors the flow of IRPs between various applications and the NTFS driver. It is an example of a passive filter driver. The output of *Process Monitor* while encryption is taking place is followed. Since three encryption packages were observed, it would be superfluous to illustrate all three here as the events are repetitive; therefore the events of one package (PrivateFile) are illustrated below. The first operation of note in Fig 1 is a file system *QueryOpen* executed on the plaintext file to be encrypted. The *QueryOpen* is initiated by the encryption software *exe*

process. A file handle is created and results in a successful retrieval of file attributes like the MAC times along with allocated size. The stack trace of this event originates from Kernel mode (ntkrnlpa.exe) to user mode (msvbvm60.dll). Next there is a file system *CreateFile* call for read access to the file plaintext file. This results in an *opened* status. Similarly the stack trace originates in kernel mode and traverses to user mode. Finally there is a *CloseFile* instruction whose job is to close down and free up the previous IRP associated resources. The file is now ready to be read.

Time of Day	PID	Operation	Path	Detail
21:57:27.4657361	6888	FASTIO_NETWORK_QUERY_OPEN	C:\Case Study\Secret.txt	CreationTime: 14/01/2012 21:55:49, LastAccessTime: 14/01/2012 21:56:53
21:57:27.4680278	6888	IRP_MJ_CREATE	C:\Case Study\Secret.txt	DesiredAccess: GenericRead, Disposition: Open, Options: Synchronous I/O
21:57:27.4681714	6888	IRP_MJ_CLEANUP	C:\Case Study\Secret.txt	

Fig. 1. Initial File System Events with Plaintext file

Time of Day	PID	Operation	Path
21:57:27.4909309	6888	FASTIO_NETWORK_QUERY_O...	C:\Case Study\Ciphertext\Secret.txt.pfs
21:57:27.4910374	6888	IRP_MJ_CREATE	C:\Case Study\Ciphertext\Secret.txt.pfs
21:57:27.4915480	6888	FASTIO_QUERY_INFORMATION	C:\Case Study\Ciphertext\Secret.txt.pfs
21:57:27.4915860	6888	IRP_MJ_CLEANUP	C:\Case Study\Ciphertext\Secret.txt.pfs

Fig. 2. File System Events with Ciphertext file

Similarly there are file system calls (*QueryOpen*, *CreateFile* and *CloseFile*) executed on the ciphertext file. The name of this file is inputted by the user or the software automatically populates the filename field by just appending the new file extension to the original plaintext file name, Fig 2. There is also a *QueryStandardInformationFile* call to query allocation size and determine if the entry is a directory or not; the file is now prepared to be written to. There are file system calls (*CreateFile* & *CloseFile*) but in addition there are calls to read the contents of the plaintext file and also to write the plaintext contents to the first temporary file, Fig 3.

21:57:27.9949550	6888	FASTIO_READ	C:\Case Study\Secret.txt
21:57:27.9949922	6888	IRP_MJ_CLEANUP	C:\Case Study\Secret.txt
21:57:28.4912877	6888	IRP_MJ_READ	C:\Documents and Settings\Local Settings\Temp\pfi20.tmp
21:57:28.4913444	6888	IRP_MJ_CLEANUP	C:\Documents and Settings\Local Settings\Temp\pfi20.tmp
21:57:28.4925778	6888	IRP_MJ_CREATE	C:\Documents and Settings\Local Settings\Temp\pfi20.tmp
21:57:28.4934603	6888	IRP_MJ_WRITE	C:\Documents and Settings\Local Settings\Temp\pfi20.tmp

Fig. 3. Temporary file1 system events

Time of Day	PID	Operation	Path
21:57:27.9939323	6888	FASTIO_NETWORK_QUERY_OPEN	C:\Documents and Settings\Local Settings\Temp\zia1136f
21:57:27.9941278	6888	IRP_MJ_CREATE	C:\Documents and Settings\Local Settings\Temp\zia1136f
21:57:27.9953140	6888	IRP_MJ_WRITE	C:\Documents and Settings\Local Settings\Temp\zia1136f

Fig. 4. Temporary file2 system events

Subsequently the plaintext contents are re-written to a second temporary file where the contents of this are encrypted, in Fig 4. Next there is a new handle created to the ciphertext file and then the encrypted contents of temporary file 2 are written into the designated ciphertext file. Please see Fig 5. The two temporary files during the encryption process are deleted. This is achieved by the file system calling a

setDispositionInformationFile call, while passing a boolean variable *Delete* set to true. There is a file system *QueryOpen* called on the plaintext file and this returns the MAC times of the file. Then there is also a *QueryInformationVolume* where the volume- create time and volume serial number are returned. There is also a flag (*SupportObjects*) returned to indicate whether *Objects* are supported or not, this is a reference to the DLT service mentioned earlier. Since the value returned here is *true* there is a subsequent file system control call to retrieve the *ObjectID*. This is the objectid of the birth volume i.e. volume id of where the plaintext file was originally created.

Time of Day	PID	Operation	Path
21:58:31.0374602	3484	IRP_MJ_CREATE	C:\Case Study\Ciphertext\Secret.txt.pls
21:58:31.0375198	3484	IRP_MJ_QUERY_INFORMATION	C:\Case Study\Ciphertext\Secret.txt.pls
21:58:31.0375702	3484	IRP_MJ_QUERY_INFORMATION	C:\Case Study\Ciphertext\Secret.txt.pls
21:58:31.0487339	3484	IRP_MJ_WRITE	C:\Case Study\Ciphertext\Secret.txt.pls
21:58:31.0489284	3484	IRP_MJ_CLEANUP	C:\Case Study\Ciphertext\Secret.txt.pls

Fig. 5. Encrypted material written to ciphertext file

3484	IRP_MJ_LHWRITE	L:\Case Study\Ciphertext\secret.txt.pls	Type: QueryBasicInformationFile
3484	IRP_MJ_QUERY_INFORMATION	C:\Case Study\Ciphertext\Secret.txt.pls	CreationTime: 14/01/2012 21:57:27
3484	IRP_MJ_QUERY_INFORMATION	C:\Case Study\Ciphertext\Secret.txt.pls	LastAccessTime: 14/01/2012 21:58:31
3484	IRP_MJ_READ	C:\Case Study\Ciphertext\Secret.txt.pls	LastWriteTime: 14/01/2012 21:57:29
3484	IRP_MJ_CLEANUP	C:\Case Study\Ciphertext\Secret.txt.pls	ChangeTime: 14/01/2012 21:57:29
3484	IRP_MJ_CLEANUP	C:\Case Study\Ciphertext\Secret.txt.pls	FileAttributes: A

Fig. 6. Timelines and timestamps

When a file is accessed or read for file encryption purposes, the last access time attribute of the plaintext file will indicate the approximate creation time of the encrypted file. The approximation is caused by the difference in time or lag between the plaintext file contents being read, buffered in a temporary file then written to a second temporary file where the encryption is carried out. Once encryption is completed, the ciphertext data is written to the output file. The timestamp for when the IRP_MJ_CLOSE (fileClose) executes on the ciphertext file indicates the last access timestamp, as can be seen in Fig 6. The ciphertext file create timestamp is later than the last access timestamp for the plaintext file, in addition the ciphertext file last access timestamp is later than that of the plaintext file.

4 Characterise the Encryption Process

As can be seen from the observations above the flow of the encryption process can be characterised as a series of file I/O actions. The flow of data through the encryption process is depicted in Fig 7. The overall result of observing *Privatefile* encryption is that four files are created (when plaintext file is not deleted); two temporary files, a lnk file and the ciphertext file. The different interfaces (files) that the data flows through can be seen and it is at these “touch-points” along the process flow where evidence can be retrieved. This is because particular events, as indicated in [7] that occur at each of these points are recorded chronologically in the \$logfile e.g. each mft update is recorded in \$logfile and its entry is preceded with the following string “FILE0”.

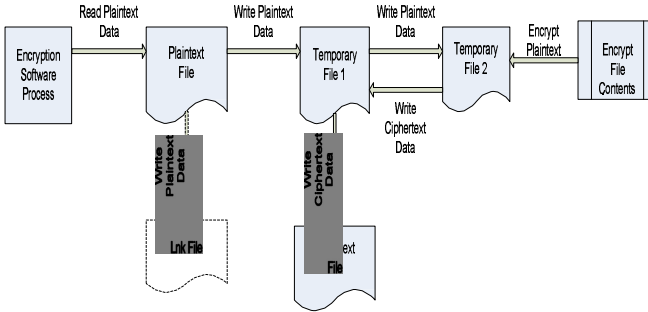


Fig. 7. Flow of data through

4.1 Sequence of Events That Constitute the Encryption Process

The sequence model in Fig 8 displays the order in which events take place. This is important in understanding the contents of the \$logfile because a forensic picture of events can be constructed. The individual events listed here can be classified into various groups of event-types see table 1. This is needed in forming event sequence signatures.

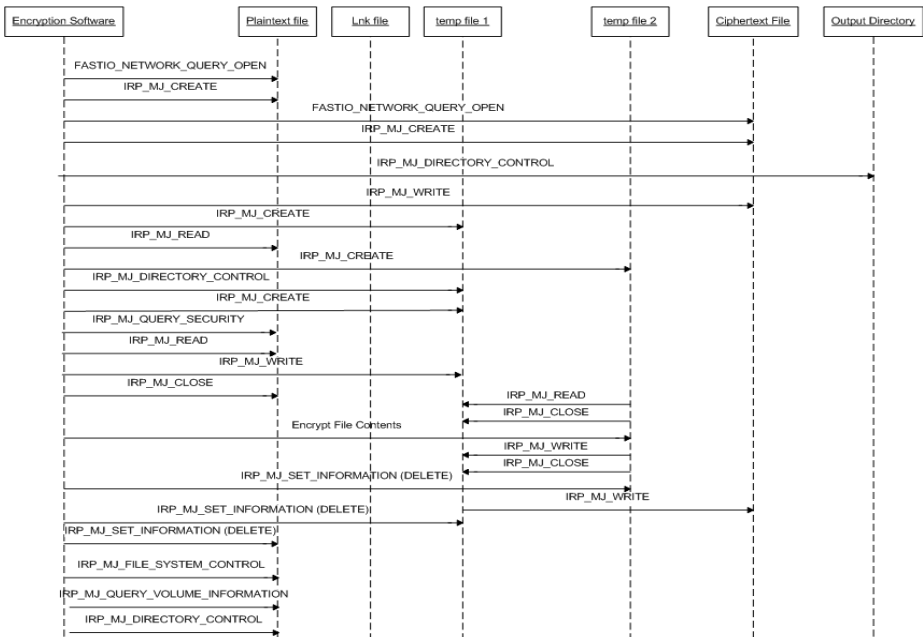


Fig. 8. UML Sequence Model (plaintext file not deleted)

4.2 Establishing an Event Sequence Signature of the Encryption Process

Three encryption software packages were studied here. It was found that in all three cases that the events which led to the creation of an encrypted file were consistent with each other. However it was noted that the frequency of certain events and their sequence varied slightly. Please see table 1 below where the individual event-types of the three encryption software packages are summarised. It was noted that when the plaintext file was deleted during encryption that there was no *lnk* file created. It was also noted that when plaintext file was not deleted that *Privatefile* was inconsistent in creating a *lnk* file. Under closer scrutiny this was understood to be conflicting with the anti-virus (AV) scan that was occurring at the time of experiment. Also noted was that *Meo* software didn't create any *lnk* nor temporary files and this was irrespective of any AV scans.

Table 1. Event-types

Event -type	Plaintext file deleted			Plaintext file not deleted		
	Privatefil	PGP	ME	Privatefil	PGP	ME
Create file for plaintext						
Create file for ciphertext						
Create lnk file	X	X	X			X
Create temp file(s)	(2)	(1)	X			X
Read plaintext file contents						
Write to temp file (s)	(2)	(1)	X			X
Write to ciphertext file						

5 Modeling Event Sequence Signature of the Encryption Process

Having observed the consistent occurrence of specific event types in table 1 during the file-encryption process, with different frequencies and event-sequences - this led to the question to see if the encryption process could be generalised using some formal approach. To this end a suitable formalism was introduced to define the encryption process as an event sequence signature.

5.1 Intrusion Detection Systems – Event Sequence Signature

In the design of Intrusion Detection Systems (IDS) there is a technique used to detect an intrusion which is called *anomaly detection*. The detection assumes attacks to be well-known sequences of actions. These actions are represented in the form of special patterns, called attack signatures. Attack signatures can be either mono-event or multi-event, depending on the number of steps in the corresponding attack scenarios. Defining multi-step attack signatures in a declarative form has been presented and it shows how temporal properties of multi-event attack signatures can be modelled in [10]. The presented model is based on high-level declarative Interval Temporal Logic (ITL). Temporal logic extends propositional logic with a notion of time by introducing special temporal operators e.g. always (D), sometime (\diamond), at the next

moment (D). Adding them allows true statements to be defined. The model in [10] is a slightly modified subset of ITL and it is called it *SigITL* (Signature ITL). Rules can be defined directly where a temporal formula that states a partially-ordered set of events in a multi-event signature. Each event is regarded as an atomic proposition of *SigITL* which are assumed to be mutually exclusive. Then temporal properties like sequence, any order, partial-order, exclusive choice, non-occurrence and repetition are defined.

5.2 Modeling Event Sequence Signature for the Encryption Process

By applying the SigITL specified in [10], the events in table 1 are grouped and modelled according to the artefact or “touch-point” that is recorded in \$logfile. In order to formally define the modelling rules, let the following be mutually exclusive atomic propositions of SigITL: A = Read Plaintext file process, B = Create and Write to *tmp* file (repetition), C = Create *lnk* file process, D = Create and Write Ciphertext file, E = Delete Plaintext file (if selected by user; non-occurrence), Z_1 = User selection: decision to delete plaintext file and Z_2 = User selection: decision not to delete plaintext file. The temporal events like sequence, non-order, non-occurrence and mutually exclusive are defined below. If the events must occur in a fixed sequential order, then they are expressed as follows:

$$\diamond A; \diamond B; \diamond C; \diamond D; \diamond E \quad \text{or} \quad \diamond (A ; B ; C ; D ; E)$$

Equation 1. Expressing an ordered sequence of events

When the events occur in no fixed order then they can be expressed using the “ \wedge ” operator. So the events in table 1 can be summarised as :

$$\diamond A ; (\diamond B \wedge \diamond C \wedge \diamond D \wedge \diamond E)$$

Equation 2. Sequence of events with deletion of plaintext file

However it was observed that proposition B must follow A and E must occur last. Propositions C and D occurred in no fixed order other than after A and B but before E. The non-occurrence of an event between two others can also be expressed using the “ $D \neg$ ” operator. The occurrence of at least n repetitions of a particular event type can also be expressed. In this case the proposition B is expressed as B^n where $n = 0, 1, 2$ since it was observed that B can occur zero times, once or twice, now the following sequence signature model is arrived at:

$$\diamond A ; (\diamond B^n \wedge D \neg E) \wedge (\diamond C D \neg E) \wedge (\diamond D \wedge D \neg E); \diamond E$$

Equation 3. Events with non-occurrence and with repetition

Alternatively when there is no deletion of the plaintext file the events can be modelled as:

$$\diamond A ; \diamond B^n \wedge \diamond C \wedge \diamond D$$

Equation 4. Events with no deletion of plaintext file

But the exclusive choice between two or more alternative events is represented by the operator \oplus . Since there is one decision to be made between Z_1 or Z_2 then there is an exclusive choice between Equation 3 and Equation 4. This is modelled: $\diamond Z_1; (\diamond A; (\diamond B^n \wedge D \neg E) \wedge (\diamond CD \neg E) \wedge (\diamond D \wedge D \neg E); \diamond E) \oplus (\diamond Z_2; (\diamond A; \diamond B^n \wedge \diamond C \wedge \diamond D))$. Equivalently, this true statement is now expressed as:

$$\diamond(Z_1; (A; (B^n \wedge D \neg E) \wedge (CD \neg E) \wedge (D \wedge D \neg E); E) \oplus (Z_2; (A; B^n \wedge C \wedge D)))$$

Equation 5. Model of event sequence signature

Equation 5 represents the generalised event sequence signature that occurs during file-encryption. This leads to the ability of recognising the occurrence of file-encryption and the subsequent analysis and investigation of encryption by using \$logfile.

5.3 Constraint Satisfaction (CS) and Backtracking

Now that the event sequence signature can be modelled and generalised for the file encryption process it will provide a basis to automate the methodology. The main components of the methodology will consist of identifying the atomic propositions (A, B, C, D, E, F, Z_1 & Z_2) listed above.

To classify the type of model that Equation 5 represents is not that complex as it clearly represents a constraint satisfaction problem (CSP). In general CS is the process of finding a solution to a set of constraints that impose conditions that the variables must satisfy [11]. The general CSP consists in finding a list of values $x = (x[1], x[2], \dots, x[n])$, that satisfies some arbitrary constraint i.e. a boolean function. In this research $x = (A, B, C, D, E, Z_1, Z_2)$. Backtracking is an important tool for solving CSPs. Backtracking recursively builds candidates to the solutions [12], and abandons each candidate as soon as it determines that it cannot be completed to a solution. Backtracking forms the basis of the automated solution to the methodology, this is implemented in *FindTheFile*, see section 8.

6 Methodology

6.1 Identify the Encrypted File to Be Investigated

As described [9] when a file or a folder is created then a series of log records are written out to the \$logfile. The hexadecimal series 0B/0B→08/00→0B/0B→07/07→1B/01 was observed to occur a number of times when the ciphertext file and other files are created during encryption. After an image of the HDD is taken and the \$logfile is exported for analysis the file name of the encrypted file under investigation is determined.

6.2 Determine BirthVolumeID of Ciphertext File and VolumeID

The *BirthVolume ID* of the encrypted file is determined and then matched with the *VolumeID* of the volumeID of the volume used. It can be concluded that the encrypted file was created on the same volume of forensically acquired volume under investigation. So updates or modifications to the ciphertext file would be in the \$logfile.

6.3 Determine \$FILE_NAME of Ciphertext File

The final occurrence of the ciphertext file name is searched for in the \$logfile as a unicode string. This provides a starting point from which to step backwards in the \$logfile, backtracking will be used here. Using the hexadecimal series referred to in 6.1 the \$FILE_NAME attributes are searched for in the \$logfile. These names are the Win32 name and the DOS name, see [9] for more detail. By analysing the last occurrence of \$FILE_NAME attribute in the \$logfile, the timestamps can be extracted. There are the three MAC times and the MFT modification time displayed here. Please see next step in 6.4 below, from this it can be seen when the ciphertext file was created.

6.4 Examine the Timestamps

NTFS timestamps contain the last modified, last accessed created and the MFT modified times of a file. These form part of the NTFS \$FILE_NAME attribute of a file. These hexadecimal values are decoded to give date and time in UTC. The creation date of the encrypted file is given to be at the time when there is IRP_FILE_CLOSE was executed on the ciphertext file. This time closely approximates the last access time of the plaintext file.

6.5 Determine Where the Add/Delete Index Entry

For the newly created files the 0x0e/0x0f log record is included in \$logfile as this indicates when a file is added/deleted from the index entry. This value is used to determine if plaintext file is deleted or not. This index entry includes a \$FILE_NAME attribute.

6.6 Determine Other Files Created during Process

Using the hexadecimal patterns outlined in 6.1 and 6.5 the temporary files created during encryption are identified. During the *Privatefile* encryption a temporary file is used where the data is *readin* and buffered from the plaintext file. Then the data is written from this temporary file to a second temporary file, where it is encrypted. The encrypted material is written to the ciphertext file from there. If the plaintext file is not deleted then a *.lnk* file is created, this links back to plaintext file.

6.7 Use the “FILE0” Entry in \$logfile to Step Backwards

Each MFT entry starts with the ascii signature string *FILE0* (or 0x46494C4530). By backtracking using the “FILE0” string and by examining the log details of the newly created and updated files (temp files, lnk, ciphertext and plaintext) the touch-points or the interfaces are revealed. This indicates the chronology and the sequence that would take place i.e. when the plaintext file was last updated with an MFT update on access time and also the newly created files’ MFT update entries.

6.8 Determine the Original Plaintext File Name

The unicode string value of the plaintext file name can easily be extracted from the \$FILE_NAME attribute in the \$logfile. By backtracking in \$logfile and passing each file or touch-point will lead to determining the original plaintext file name. There are Win32 and DOS \$Filename attributes. When deletion occurs the process follows a different series of log record entries i.e. 0F/0E→03/02→16/15→0B/0B→08/00→0B/0B→07/07→1B/01. For deletion the composite pattern 0F/0E→03/02→16/15 precedes the 0B/0B→08/00→0B/0B→07/07→1B/01 composite pattern. Indicating that deleting (0x0F) and adding (0x0E) index entry allocation, de-allocating (0x03) and the initializing (0x02) of file record segments. The first part of the composite is for “Redo” and the second part is for “Undo” operation.

6.9 Examine the contents of the Plaintext file

The contents of the \$DATA attribute of the plaintext file can be located and the hexadecimal values extracted. The contents will remain even if the file in question was deleted during the encryption process. This is because the \$bitmap attributes and the data is just marked as de-allocated in NTFS. Depending on the size of the plaintext file, then the data can be stored residently on \$mft or non-residently. The hexadecimal pattern 07/00→07/00 indicates what the resident data is otherwise the addresses of the clusters or run of clusters where non-resident data is specified in [7].

6.10 Determine BirthVolumeID of Plaintext File

This is to validate that the plaintext file identified was encrypted on the same volume. This would be carried out after identifying and locating the original plaintext file.

7 Case Study

The overall objective of this case study is to validate the methodology to see if it can establish an evidential link between the encrypted file and the original plaintext file and also view the plaintext contents. The name of the ciphertext file under investigation is *Secret.txt.pfs*. If the file name is obfuscated, renamed or its attributes changed then these activities can be tracked and traced in the \$logfile – so evidence

of this would be detected. This activity was observed to have the 0x05/0x06 composite pair in the \$logfile.

7.1 Determine BirthVolumeID of Ciphertext File and VolumeID

Determine the BirthVolumeID by using *fsutil*. A *fsutil* query is executed against the file and the resulting BirthVolumeID is outputted. The identical volumeID is confirmed to occur in \$logfile for the ciphertext file. The BirthVolumeID in Fig 9 and the volumeID in Fig 10 are the same so it can be concluded that the encryption took place on this volume.

```

C:\>fsutil objectid query "c:\Case Study\Secret.txt.pfs"
Object ID : b18f0b5ad448e111a0f708007bb2f121
BirthVolume ID : 286e0313b4a0f749aa25b6423d6e6326
BirthObjectId ID : b18f0b5ad448e111a0f708007bb2f121
Domain ID : 00000000000000000000000000000000
    
```

Fig. 9. BirthVolumeID of Ciphertext File extracted using fsutil

```

00F70040 | 00 00 00 08 6E 03 13 B4 A0 F7 49 AA 25 B6 42 3D | (n ' +I%TB=
00F70050 | 6E 63 26 25 7C D3 69 94 3E E1 11 A0 E4 64 31 50 | nc&|0i|>á àd1P
00F70060 | 8B 79 DF 28 6E 03 13 B4 A0 F7 49 AA 25 B6 42 3D | lYB(n ' +I%TB=
00F70070 | 6E 63 26 25 7C D3 69 94 3E E1 11 A0 E4 64 31 50 | nc&|0i|>á àd1P
    
```

Fig. 10. VolumeID of Ciphertext file in \$logfile

7.2 Determine \$FILE_NAME of Ciphertext File

Find the last occurrence of the ciphertext file name in \$logfile and backtracking search is initiated from this point, please see Fig 11. This is part of the \$FILE_NAME attribute.

7.3 Examine the Timestamps

As can be seen from Fig 11 the four timestamps are create, last modified, mft modified and last accessed– in this order. The 0x 9EADACA345DCCC01 value represents the time Thu, 26 January 2012 16:14:54 UTC, this represents the last modified, mft modified and last accessed times. The last access time of plaintext file will closely approximate this time.

```

0044E0B0 | 84 69 C5 A2 45 DC CC 01 9E AD AC A3 45 DC CC 01 | iA@EU! |--EU!
0044E0C0 | 9E AD AC A3 45 DC CC 01 9E AD AC A3 45 DC CC 01 | |--EU! |--EU!
0044E0D0 | 70 01 00 00 00 00 00 00 6E 01 00 00 00 00 00 00 | p n
0044E0E0 | 20 00 00 00 00 00 00 00 0E 01 03 00 65 00 63 00 | S e c
0044E0F0 | 72 00 65 00 74 00 2E 00 74 00 78 00 74 00 2E 00 | r e t . t x t .
0044E100 | 70 0C 66 00 73 00 08 00 61 0C 01 00 00 00 B9 00 | p f s a '
    
```

Fig. 11. Ciphertext file name and timestamps

7.4 Other Files Created during the Encryption Process

Using the same pattern 0B/0B→08/00→0B/0B→07/07→1B/01 (the series 0B/0B→07/07→1B/01 is used to close transactions) and by tracking in the \$logfile it was seen that the other files are created i.e. .lnk and two temporary files. The .lnk file is created if the plaintext file is not deleted during encryption.

7.5 Plaintext File - (Irrespective If It is deleted)

The plaintext file name is determined to be *Secret.txt*. After the encryption process has accessed and opened the plaintext file for reading, the last access time recorded in the \$logfile is 0x1E55289745DCCC01, when decoded is *Thu, 26 January 2012 16:14:33 UTC*, Fig 12.

```

00F6BD80 | 10 4A 85 9D 00 00 00 00 | 1E 55 28 97 45 DC CC 01 | JII U(IEU)
00F6BD90 | 00 00 08 80 00 00 00 00 | 2B 05 00 00 20 00 00 00 | | +
00F6BDA0 | 14 00 3C 00 53 00 65 00 | 63 00 72 00 65 00 74 00 | < Secret
00F6BDB0 | 2E 00 74 00 78 00 74 00 | B7 D7 1E 79 03 00 00 00 | .txt .x y
    
```

Fig. 12. Plaintext file-last access date

7.6 Plaintext Contents

Even if the plaintext file is deleted or not-deleted; the plaintext contents remains in the \$logfile, as can be seen in Fig 13. This is subject to the clusters and sectors not being overwritten by other data that might be added later. Note: The plaintext contents is also available in the \$MFT – this only occurs when the plaintext file is not deleted during the encryption process.

```

00BE1090 | 4F 22 43 00 00 00 00 00 | 53 65 63 72 65 74 20 4D | 0°C Secret M
00BE10A0 | 65 73 73 61 67 65 3A 20 | 49 63 61 72 75 73 20 66 | message: Icarus f
00BE10B0 | 6C 65 77 20 74 6F 6F 20 | 63 6C 6F 73 65 20 74 6F | lew too close to
00BE10C0 | 20 74 68 65 20 73 75 6E | 21 5E 08 00 24 00 49 00 | the sun! $ I
    
```

Fig. 13. Plaintext content in \$logfile

7.7 Determine BirthVolumeID of Plaintext file

This is a validation step and *fsutil* query is executed against the identified plaintext file and the resulting BirthVolumeID is outputted. The outputted BirthVolumeID is identical to what is extracted from the \$logfile, please see Fig 14 and Fig 15.

```

C:\>fsutil objectid query "c:\Case Study\Secret.txt"
Object ID : b28f0b5ad448e111a0f708007bb2f121
BirthVolume ID : 286e0313b4a0f749aa25b6423d6e6326
BirthObjectId ID : b28f0b5ad448e111a0f708007bb2f121
Domain ID : 00000000000000000000000000000000
    
```

Fig. 14. BirthVolumeID of Plaintext File using fsutil

00CEFA40	43 00 61 00 73 00 65 00 20 00 53 00 74 00 75 00	C a s e S t u
00CEFA50	64 00 79 00 5C 00 53 00 65 00 63 00 72 00 65 00	d y \ S e c r e
00CEFA60	74 00 2E 00 74 00 78 00 74 00 0D 00 43 00 3A 00	t . t x t
00CEFAA0	34 00 00 00 00 00 28 6E 03 13 B4 A0 F7 49 AA 25	(n ' -I%&
00CEFAB0	B6 42 3D 6E 63 26 26 7C D3 69 94 3E E1 11 A0 E4	¶B=nc&& óí >á à
00CEFAC0	64 31 50 8B 79 DF 28 6E 03 13 B4 A0 F7 49 AA 25	d1P yB(n ' -I%&
00CEFAD0	B6 42 3D 6E 63 26 26 7C D3 69 94 3E E1 11 A0 E4	¶B=nc&& óí >á à

Fig. 15. BirthVolumeId from \$logfile

7.8 Result of Investigation

The name of the plaintext file that was encrypted was revealed to be *Secret.txt*. The ciphertext file name is *Secret.txt.pfs* (as was known) and the secret message which was encrypted is “*Secret Message: Icarus flew too close to the sun!*”. The result of this case study is that it validates the methodology. It validates that the objectives of being able establish an evidential link between the encrypted file and the plaintext file while also revealing the plaintext contents of the encrypted file are met.

8 Automation of Methodology: FindTheFile Parser

Since the event sequence signature of the encryption process can be modeled and the occurrence of the event sequence signature can be classified as a constraint satisfaction problem- this provided the framework to the approach taken to automate the methodology and using the backtracking algorithm for searching event signatures. The use of backtracking is justified as it is the formally recognised solution to a CSP. As a result the parser was built using the JAVA high-level language. The JAVA language was selected as the language of implementation because of its platform cross-compatibility and its extensive library of APIs. The parser was called *FindTheFile* and the central class for parsing is the *StringTokeniser*. This class was instantiated for each activity that is carried out in the encryption process e.g. Create, Write, Delete file. These activities have corresponding hexadecimal entries in the \$logfile that facilitates identification of these activities. These hexadecimal entries are used to initialise each *StringTokeniser* class with the appropriate tokeniser e.g. for newly created files the hexadecimal composite of *0x0e/0x0f* log record would be used as a tokeniser to indicate the action of adding a newly created file. The *Runtime* class from the *java.io.** library is also used to call the external tool called *fsutil*.

8.1 Implementing the Backtracking and Recursion in Java

In order to apply backtracking to the data of a particular instance of a problem that is to be solved the following procedural parameters: root, reject, accept, first, next, and output are implemented. *FindTheFile* takes the instance data X as a parameter and would do the following: root(X): return the partial candidate at the root of the search tree, reject(X,c): return true only if the partial candidate c is not worth completing, accept(X,c): return true if c is a solution of X and false otherwise, first(X,c): generate the first extension of candidate c, next(X,s): generate the next alternative extension of a candidate, after the extension s and finally output(X,c): use the solution c of X, as

appropriate to the application. These steps are the pseudo code for implementing the backtracking solution (with recursion). This solution is the searching for the occurrence of an event sequence signature which is modeled by equation 5. The use of the *iterator* JAVA class was used at the core of this processing. This subsequently facilitated the automation of the methodology.

8.2 Recognising Temporary and Link Files

In order for the parser to be able to separate various created files (temporary and link) from the target file (plaintext file), the parser uses straightforward rationale. The rationale is that a temporary file- no matter what software package creates it, it will always be deleted. So the parser identifies a temporary file by the sequence of events that occur in the \$logfile i.e. the file is created, processed (written to or read from) and then deleted. There are hexadecimal operator codes to indicate these actions in the \$logfile. This eliminates any problems that might arise with *FindTheFile* not recognising file-naming conventions of a specific encryption software package might have. In the case of recognising the link file – the rationale is simply; a link file will always have a .lnk extension and this never changes no matter what encryption software is used.

8.3 FindTheFile

This parser was successfully implemented and was run against the case study. The \$logfile was loaded and the ciphertext file under investigation was entered and then the parser was started, please see Fig 16. The output panel was generated with the results of the investigation. The results of the parser demonstrate that the search located all files that are associated with the ciphertext file under investigation in the \$logfile, while backtracking (with recursion) was successfully implemented. It also indicates the original plaintext file name and shows with a check-box if it was deleted during the encryption or not. A text field is also outputted with the plaintext contents of the original file, regardless if the file was deleted or not. The timestamps (in UTC) display creation date of the encrypted file and the last accessed time of the plaintext file that was encrypted, see Fig 17.

FindTheFile

Load logfile:

c:\\$logfile

Enter ciphertext file name:

c:\Case Study\Secret.txt.pfs

Press start to parse \$logfile :

Fig. 16. FindTheFile: Initial Screen

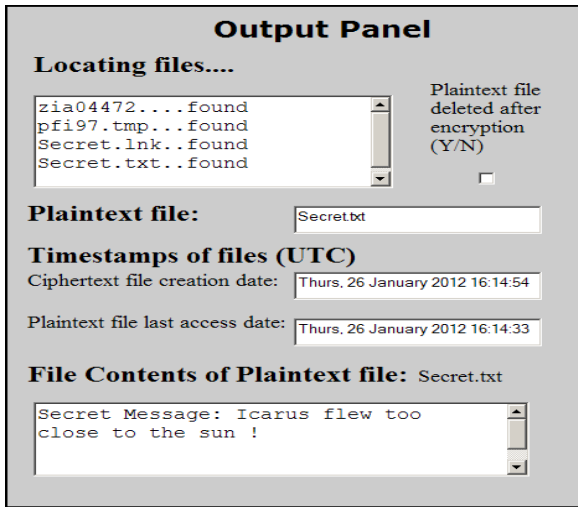


Fig. 17. FindTheFile: Output Panel

9 Performance Evaluation

9.1 Introduction

As stated previously the research objective was to develop and evaluate a methodology for the investigation of encrypted material using the \$logfile where: a) there is an evidential link established between encrypted file and the plaintext file and, b) the plaintext contents of the original file is revealed. Receiver Operator Curve (ROC) analysis is carried out to evaluate how well the methodology performed.

9.2 ROC Analysis

ROC analysis incorporates binary classification which is the classifying of a given set of objects into two groups on the basis of whether they have some property or not. The medical community applies this to testing techniques and a typical scenario is the medical testing carried out to determine if a patient has a certain disease or not. ROC analysis is a useful technique for visualizing, organizing and selecting/evaluating classifiers based on their performance. When the area under the ROC curve (AUC) is computed it will indicate the measure of performance as a scalar of the chosen classifier. The classifier of interest is the finding the right plaintext file and its contents in \$logfile. It is discussed [13] that when measuring the performance of medical and quality control tests, the concepts *sensitivity* (true positive rate - TPR), *specificity* (true negative rate - TNR) and *1-specificity* (false positive rate - FPR) are used; these concepts are readily usable for the evaluation of any binary classifier. The number of true positives, false negatives, true negatives, and false positives always add up to 100% of the set. It is explained that in statistical hypothesis testing of an experiment, there will be a null hypothesis and an alternative hypothesis [13]. Based on the outcome of the experiment, it will be decided whether to reject the null

hypothesis or not. If the result of the experiment is statistically significant, then the null hypothesis is rejected in favour of the alternative hypothesis.

9.3 Experiment

The following experiment was carried out –where there are six encryption scenarios i.e. encryption with three different packages and for each package there are two options –deleting and not deleting the plaintext file. As a result there are six event sequence signatures to monitor and analyse, for each instance two text files were encrypted. Then this means there are twelve instances of Equation 5. These are numbered in column “No.,” please see Table 2. The objective of the experiment is to determine a measure of performance of the methodology in terms of true-positives and false-positives. A classification variable (dichotomous) that would indicate results of instantiating the methodology was selected and this is called ‘Successful’ where Yes and No are the outcome. Then the binary representation of this is in column “Binary” where there are two classes 1= success and 0= failure.

9.4 Results and Observations

Using the “1”s listed in “Binary” column Table 2 as the list of true positives (TPs) which are also listed in column B of Table 3, then a list of false positives (FPs) can be created - take the TPs and replace “0” with “1” and vice-versa [13]. The FPs are listed in column C of Table 3. The TP rate is then calculated as being the proportion of files above this point that can be correctly investigated. This is calculated by summing the number of TPs above this point in the table and then dividing by the total number of TPs. These values are listed in column D of Table 3. Similar calculations are carried out for the FP rate in column E. The true negative (TN) rate is simply calculated by subtracting the FPR from 1 because $FPR=1-Specificity$. These calculations constitute the ROC data and are listed in Table 3.

Table 2. Evaluation for six event sequence signatures

No. (Instance of Eqn.5)	Delete plaintext File Yes = Y No = N	Encryption Package	Successful ✓ =Yes X=No	Binary ✓ =1 X= 0
1.1	Y	Privatefle	✓	1
1.2	N	Privatefle	✓	1
2.1	Y	Privatefle	✓	1
2.2	N	Privatefle	✓	1
3.1	Y	PGP	✓	1
3.2	N	PGP	✓	1
4.1	Y	PGP	✓	1
4.2	N	PGP	✓	1
5.1	Y	MEO	✓	1
5.2	N	MEO	✓	1
6.1	Y	MEO	X	0
6.2	N	MEO	✓	1

Table 3. Data to plot ROC graph

No. (Instance of Eqn.5)	A TP	B FP	C TP rate (Sensitivity)	D FP rate (1- specificity)	E TN rate (Specificity)	F
1.1	1	0	0	0	1	1
1.2	1	0	0.091	0	1	1
2.1	1	0	0.182	0	1	1
2.2	1	0	0.273	0	1	1
3.1	1	0	0.364	0	1	1
3.2	1	0	0.455	0	1	1
4.1	1	0	0.545	0	1	1
4.2	1	0	0.636	0	1	1
5.1	1	0	0.727	0	1	1
5.2	1	0	0.818	0	1	1
6.1	0	1	0.909	0	1	1
6.2	1	0	0.909	1	0	0

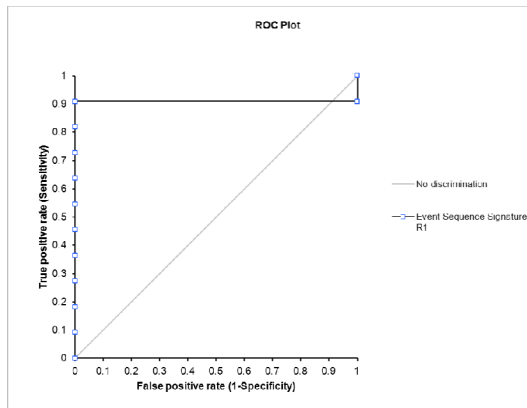


Fig. 18. ROC Plot

Using the data in the Table 3, the TPR as Y-axis and the FPR (recall that $FPR = 1 - Specificity$) as X-axis were graphed to give the ROC plot in Fig 18. Then the AUC was computed by calculating the area for each row (where A is the TPR column and C is the FPR column) using the trapezoid rule [13]. Subsequently, a decision plot can be drawn up – not included here. The decision plot will allow the choice of the sequence event signature that minimizes the rate of false positives and aids in the selection of a specific value to use as a threshold that provides a desired trade-off between the true positive rate and the false positive rate.

It is pointed out that when the AUC is 1 the accuracy of the classifier is concluded to be excellent, when AUC is between 0.80 and 0.90 the accuracy of the test is

regarded to be good, while 0.70 to 0.80 indicates a fair accuracy level, 0.60 to 0.70 is regarded to be poor and anything else warrants test failure [13]. The AUC in this research has been calculated to be 0.91. Therefore it can be inferred that the methodology yielded an excellent result. The results verify the selection of the instance of Equation 5 variable (Success/Failure) as a classifier. In relation to Hypothesis Testing, if the Null hypothesis is H_0 (indicating randomness) and the alternative hypothesis is H_1 (indicating non-randomness), then where H_0 : area ≤ 0.5 . H_1 : area > 0.5 . So in relation to this research, the results of the experiment prove to be more powerful than a random rule as the AUC is 0.91. As a result H_0 is rejected and H_1 is accepted i.e. results from the methodology classifier are not random and are statistically significant. As a note - any points that lie under the line of no discrimination in Fig 18, would represent where no discrimination can be made between TPR or FPR this would be regarded as test failure.

10 Research Contribution

The research presents a unique and original method of investigating encrypted material and revealing the plaintext. This is achieved by characterising encryption as a series of file I/O operations rather than a mathematical or a theoretical problem. Then by following the various points along the I/O process flow evidence artefacts can be identified in the \$logfile that lead to a successful investigation of encrypted material. A novel approach to the investigation of encrypted material is represented in the use of event sequence signature modelling which aided the classification of the presented problem as a constraint satisfaction problem. Then this provided the basis of implementing a successful backtracking search solution. The methodology was successfully automated by implementing a parser that parsed the \$logfile for events and was able to output the results of the investigation. A side-channel attack is defined as any attack based on side-channel information e.g. implementation or physical details of a cryptosystem. The side-channel attack is not based on brute force or theoretical weaknesses of the cryptosystem that can be exposed through cryptanalysis techniques [14]. Therefore it can be inferred that this research effectively results in a side-channel attack on encryption.

11 Future Work

This methodology will be extended to investigate multiple file formats; image formats like JPEG in particular. This will result in an investigative technique to analyse steganographically generated image files (stego-files). Once this is in place emphasis will be placed on admissibility of all evidence produced by methodology- thus ensuring *Daubert* compliance. As an approach to achieve this it is necessary to

get the methodology tested in actual field conditions rather than just in laboratory conditions. Therefore, it is intended to recommend and promote the use of this methodology within a LEA environment. Use of the methodology in this way would identify benefits and drawbacks and they would form the basis of future work.

12 Conclusion

The main outcome of this work is a formal methodology. This methodology has been validated through the development of an automated system and also through its practical application on a case study. The performance of the methodology has been evaluated using a binary classification system of true-positives and negatives and has resulted in an excellent score. The objective of the methodology has been to investigate encrypted material while revealing the original plaintext file and its contents- this has been carried out successfully in a case study. The modelling of the initial problem and backtracking solution served as a framework to facilitate the generalisation of the process and subsequent automation – so the presented methodology was compatible and interoperable with all tested types of encryption. The methodology relies fundamentally on the evidential value that can be extracted from the \$logfile. The main challenge in this research was that the data in the \$logfile is transient as the contents gets rolled over on a cyclical basis – as fresh data gets written in to the log file, older data is flushed out. Unfortunately there is insufficient knowledge on the precise nature of the log-rotation cycle of \$logfile in the public domain.

The following tools were used in this research: AccessData FTK Imager 2.5.1, WinHex 16.2 3, Process Monitor, Dcode, MS FSUTIL, Privatefile, Meo and PGP Desktop.

References

1. Carter, H.: Paedophiles jailed for hatching plot on internet (2007)
2. Joseh, S.: Hamas Terror Chat Rooms (December 11, 2007)
3. Siegfried, J., et al.: Examining the Encryption Threat, Computer Forensic Research and Development Center. International Journal of Digital Evidence (2004)
4. Bunting, S.: The Official EnCase Certified Examiner Guide. Wiley (2008)
5. McGrath, N., Gladyshev, P., Carthy, J.: Cryptopometry as a Methodology for Investigating Encrypted Material. International Journal of Digital Crime and Forensics 2(1) (January-March 2010); special edition of selected papers from e-Forensics (2009)
6. Russinovich, M.E., Solomon, D.A.: Windows Internals Covering Windows Server 2008 and Windows Vista. Microsoft Press, One Microsoft Way (2009)
7. Carrier, B.: File System Forensic Analysis. Addison Wesley, Boston (2005)
8. Parsonage, H.: The Meaning of Linkfiles in Forensic Examinations (2010)
9. Cho, G.-S., Rogers, M.K.: Finding Forensic Information on Creating a Folder in \$LogFile of NTFS. In: Gladyshev, P., Rogers, M.K. (eds.) ICDF2C 2011. LNICST, vol. 88, pp. 211–225. Springer, Heidelberg (2012)

10. Nowicka, E., Zawada, M.: Modeling Temporal Properties of Multi-event Attack Signatures in Interval Temporal Logic. Wrocław University of Technology (2006)
11. Rossi, F., Van Beek, P., Walsh, T.: Constraint Satisfaction: An Emerging Paradigm. In: Handbook of Constraint Programming. Foundations of Artificial Intelligence. Elsevier, Amsterdam (2006)
12. Gurari, E.: Backtracking algorithms “CIS 680: DATA STRUCTURES: Chapter 19: Backtracking Algorithms” (1999), <http://www.cse.ohio-state.edu/gurari/course/cis680/cis680Ch19.html#QQ1-51-128>
13. Altman, D.G., Bland, J.M.: Diagnostic Tests – Sensitivity and Specificity. *BMJ* 308(6943), 1552 (1994) PMID 8019315
14. Chen, S., Wang, R., Wang, X., Zhang, K.: Side-Channel Leaks in Web Applications: A Reality Today, A Challenge Tomorrow. In: IEEE Symposium on Security & Privacy (May 2010), <http://research.microsoft.com/pubs/119060/WebAppSideChannel-final.pdf>

Finding Data in DNA: Computer Forensic Investigations of Living Organisms

Marc B. Beck, Eric C. Rouchka, and Roman V. Yampolskiy

Cybersecurity Lab, Department of Computer Engineering and Computer Science,
Speed School of Engineering, University of Louisville, Louisville, KY 40292

Abstract. Recent advances in genetic engineering have allowed the insertion of artificial DNA strands into the living cells of organisms. Several methods have been developed to insert information into a DNA sequence for the purpose of data storage, watermarking, or communication of secret messages. The ability to detect, extract, and decode messages from DNA is important for forensic data collection and for data security. We have developed a software toolkit that detects the presence of a hidden message within a DNA sequence, and deciphers that message. In order to decode a message we are modifying several existing cryptanalysis tools that have been developed for solving simple substitution ciphers and compare their performance.

Keywords: Bioinformatics, Cryptography, DNA computing, Natural languages, Computer forensics, Steganalysis.

1 Introduction

Deoxyribose Nucleic Acid (DNA) is the carrier of hereditary information for every living organism. DNA is a double helix with two anti-parallel strands containing four different nucleotides, which are distinguished by one of the four bases adenine (A), cytosine (C), guanine (G), and thiamine (T). The two strands form base pairs of interacting complementary bases (A-T and C-G) held together by hydrogen bonds. DNA has the potential to store vast amounts of data using combinations of those four nucleotides within genomes that can range to several billion bases in length [1].

Contained within genomic sequences are regions that code for genes that produce proteins which are collections of amino acids. In the process of translation, an mRNA sequence that has been transcribed, or copied, from the gene coding region is used as a template to transform from the four base code of DNA to the 20 base code of amino acids. The process by which this transformation occurs, known as the genetic code, was first uncovered by Marshall Nirenberg [2]. In gene coding regions, a codon refers to a sequence of three nucleotides that determines which amino acid will be incorporated next during protein synthesis. With four nucleotides, this method allows $4^3=64$ possible combinations. Each codon encodes for one of 20 amino acids, with exception of the three STOP codons TAA, TAG, and TGA [3], thus allowing for degeneracy where multiple codon sequences code for the same amino acid. For the

purposes of DNA steganography, characters of messages may be encoded by variable lengths of DNA sequences that may or may not be three bases in length. While not codons in the strict biological sense, we will refer to these encoding patterns as codons for the purpose of this manuscript.

1.1 DNA Computing

DNA computing is an emerging new research field that uses DNA molecules instead of traditional silicon based microchips. The first researcher to demonstrate the computing capability of DNA was Leonard Adelman, who in 1994 developed a method of using DNA for solving an instance of the directed Hamiltonian path problem [4]. In 1997, Oghihara and Ray demonstrated that DNA computers can simulate Boolean AND and OR gates [5]. The advantage of DNA computers is that they are smaller and faster than traditional silicon computers, and they can be easily used for parallel processing. DNA has also been used as a tool for cryptography and cryptanalysis, using molecular techniques for its manipulation [3]. Bogard et al. describe how multiple sequence alignment can be used for error reduction in DNA computing [6].

1.2 DNA as Storage Medium

DNA has recently been investigated as an ultra-compact, long term data storage medium (Table 1) and a stegomedium for hiding messages. Instead of expressing a message as a series of ones and zeros, it is represented in DNA as a series of As, Cs, Gs, and Ts. A number of algorithms have been developed to encode a message in DNA characters and either disguise these messages as novel DNA sequences or encapsulate them within existing ones. It has been proven that it is possible to insert artificial DNA components that contain encoded information into the genomes of living organisms [3,7-15].

Craig Venter, who led the private effort to sequence the human genome, managed to create the first cell with a synthetic genome in 2010. The J. Craig Venter Institute (JCVI) took a computer file containing the DNA sequence of the bacterium *Mycoplasma mycoides*, modified it, produced physical DNA from this sequence, and inserted this DNA into a cell, which then reproduced under control of the new DNA to create a new bacterium [15]. This led to the creation of a company, Synthetic Genomics, which focuses on the creation of synthetic genomes for applications including vaccine design, bioenergy, and biofuels.

Using DNA as storage medium has many advantages, such as long life, redundancy, and high density. According to Bancroft et al. [16] about 200 novels or other data each equivalent in size to “A Tale of Two Cities” could be stored in a DNA microchip with the area of a postage stamp.

Table 1. Life expectancy and storage capacity of various data storage media compared to DNA

Type	Life Expectancy	Capacity
DNA	Millions of years	10^8 TB per 1 gram [1]
Hard disk	~10 years	Up to 4 TB (2011)
CD	~10 years	800 MB
DVD	<10 years	Up to 17GB
USB flash drive	~10 years, depending on usage	Up to 256GB (2011)
Tape	~30 years	Up to 35 TB

Yachie et al. [8] demonstrated the possibility to use DNA of living organisms as a data storage medium by inserting the message “ $E=mc^2$ 1905!” into the genome of *B. subtilis*. Over 99% of the encoded data was later recovered using sequence alignment methods.

Living organisms are a great storage medium when it comes to preserving data over timespans ranging in millions of years. When an organism reproduces, it automatically creates a backup copy of the data contained in its DNA. In addition, selective pressure and DNA error correction reduce the risk of the data being destroyed by random mutations. It has been suggested to use cockroaches, which are known for their resilience and high reproduction rate, as living time capsules for storing every issue of The New York Times Magazine for a certain year in their DNA which could theoretically be retrieved 1000 years later [17].

1.3 Error Correcting Approaches

Even though mutations are rare, occurring at a rate between 10^{-11} and 10^{-7} per base per replication in bacteria and higher eukaryotes [18], it is necessary to consider some form of error detection and error correction since a mutation can destroy the encrypted message in the DNA sequence. According to Yachie et al. [8], inserting the data redundantly into multiple loci of the genome is sufficient to allow the retrieval of stable and compact data without the need for template DNA, parity checks, or error correcting algorithms.

The comma code and the alternating code provide a form of error detection capability by encoding the message in a distinguishable pattern [19]. Arita [9] developed a comma-free code that has error correction capabilities. The message is translated into binary as an intermediary step. A parity bit is used in the binary code to keep the respective number of ones and zeroes odd.

The DNA-Crypt software developed by Heider and Barnekow [10] also translates messages into binary before encoding it in DNA code. It uses a very thorough approach to error detection by employing two error correction codes: the 8/4 Hamming-code and the WDH-code. The 8/4 Hamming-code is more compact, but it can correct fewer errors than the WDH code. DNA-Crypt has an integrated fuzzy controller using singleton fuzzyfication. The fuzzy controller decides which of the

two error detecting codes should be used, or none at all. This decision is based on the individual mutation rate of the DNA sequence that contains the secret message, the length of the sequence, and its stability over time. An answer is determined from those three factors by a set of rules based on heuristics [10].

2 Hiding Data in DNA

2.1 Overview

Steganography is the science of hiding information by transmitting secret messages through unsuspecting cover carriers in a way that makes the presence of any embedded messages undetectable. The term has its origins in the Greek language and means "covered writing". While the goal of cryptography is to make a message unreadable, steganography aims at avoiding suspicion to the existence of a hidden message [20]. Due to its properties as a data storage medium, DNA can be used for steganography (stegomedium).

One of the most important problems in espionage is how to get the obtained information out of the target country without the information being detected by the enemy. With the appropriate knowledge and technology, a spy could have the information inserted into the DNA of an organism, and send it out of the country as an unsuspecting biological sample. It is possible to insert not only text, but also images and many other forms of digitizable data into a DNA sequence. For that reason, it is important to develop forensic tools that can detect hidden information in DNA.

Table 2. Genetic code for protein translation (codons that code for the same amino acid regardless the third position are highlighted) [2, 3]

		Second Position of codon				
		T	C	A	G	
F i r s	T	TTT [F]	TCT [S]	TAT Tyr [Y]	TGT [C]	T
		TTC [F]	TCC [S]	TAC Tyr [Y]	TGC [C]	C
		TTA [L]	TCA [S]	TAA [end]	TGA [end]	A
		TTG [L]	TCG [S]	TAG [end]	TGG [W]	G
t P o	C	CTT [L]	CCT [P]	CAT His [H]	CGT [R]	T
		CTC [L]	CCC [P]	CAC His [H]	CGC [R]	C
		CTA [L]	CCA [P]	CAA Gln [Q]	CGA [R]	A
		CTG [L]	CCG [P]	CAG Gln [Q]	CGG [R]	G
s i t i	A	ATT [I]	ACT [T]	AAT Asn [N]	AGT [S]	T
		ATC [I]	ACC [T]	AAC Asn [N]	AGC [S]	C
		ATA [I]	ACA [T]	AAA Lys [K]	AGA [R]	A
		ATG [M]	ACG [T]	AAG Lys [K]	AGG [R]	G
o n	G	GTT [V]	GCT [A]	GAT Asp [D]	GGT [G]	T
		GTC [V]	GCC [A]	GAC Asp [D]	GGC [G]	C
		GTA [V]	GCA [A]	GAA Glu [E]	GGA [G]	A
		GTG [V]	GCG [A]	GAG Glu [E]	GGG [G]	G

An obvious choice of a location for inserting a message into a genome would be a noncoding genomic region. However, those regions might have other critical, unknown functions [7] and thus, inserting data there might possibly kill the organism. Therefore Arita et al. [7] suggested that it may be a more reliable solution to encode the message in the protein coding regions of genes. There are 20 amino acids and one stop symbol using a total of 64 possible codons [7]. Two or more codons often code for the same amino acid. Many of these redundant, or synonymous, codons typically differ in their third position, also known as the wobble base [3]. In Table 2, codons encoding the same amino acid regardless of the base in the third position are highlighted. These are the codons that can be used to embed messages.

Table 3. Research on data hiding in DNA

Researcher	Year	Coding	Message	Location	Organism
Clelland et al.[11]	1999	Clelland	June 6 invasion: Normandy	artificial	human
Brenner et al.[12]	1999	Comma code	Not reported	Bsp120I	<i>E.coli</i>
Wong et al. [20]	2003	Clelland variant	Not reported	Not reported	<i>Deinococcus radiodurans</i>
Arita and Ohashi [7]	2004	Arita	AO2KEIO1-F	ftsZ gene	<i>B. subtilis RIK8</i>
Tanaka et al. [21]	2005	Similar to Clelland	MESSAGE	Artificial sequence	Artificial DNA strand
Yachie et al. [8]	2007	Keyboard scan	E=mc ² 1905!	metB and proB	<i>B.subtilis BEST2136</i>
Heider and Barnekow[14]	2008	DNA-Crypt	TB	Vam7 sequence	<i>Saccharomyces cerevisiae CG783</i>
Jiao and Gouette [13]	2009	ASCII 8 bit binary	CODING	tatAD gene	<i>B. subtilis</i>
J. Craig Venter Institute[15]	2010	Clelland variant	Multiple messages	Not reported	Artificial bacterium

2.2 Coding Schemes

A code is an algorithm which uniquely represents symbols from some source alphabet, by symbols or strings of symbols in a target alphabet. In our case, the source alphabet is the English alphabet plus digits and punctuation characters, and the target alphabet consists of the four nucleotides. A coding scheme is a set of rules that determines which symbol of the source alphabet is represented by which symbol in the target alphabet.

Types of Coding Schemes. The coding schemes for inserting messages into DNA that have been developed can be grouped into three categories: schemes using direct translation, schemes that use intermediate steps for error detection, and schemes that have been optimized for detectability or efficiency.

The first category uses a straightforward approach by substituting a sequence of nucleotides of length n for each alphanumeric symbol [10, 12]. Since the codon in this case is of length n , up to $4n$ distinct characters can be encoded. Given the codon length of n , there are $4n!$ possible coding schemes. The coding schemes developed by Clelland [11] and Wong [20] fall into this category.

The second category of coding schemes consists of more complex schemes that use several intermediate steps, such as translating a message into binary before using a coding table to translate it into nucleotides. Intermediate steps like this are often used for error detection, since there are many proven error detection algorithms for binary messages.

The third category of coding schemes consists of schemes that were designed to meet certain criteria, such as providing error detection capability, being economical, or being easy to detect. The comma code, the alternating code, and a coding scheme based on the Huffman code [18] fall into this category.

Clelland's Coding Scheme and Wong's Coding Scheme. The coding scheme developed by Clelland et al. [11] is very similar to the one developed by Wong et al. [20]. They are both extensions of the three base codon encoding used by the genetic code. Since there are $4^3=64$ possible distinct characters that can be encoded, this scheme allows for all 26 characters of the English alphabet, the digits 0-9, and special characters. Both coding schemes do not use all possible codons.

DNA-Crypt. The DNA-Crypt coding scheme developed by Heider and Barnekow [10] translates a message into a five bit sequence, where one bit serves as parity bit to keep the respective number of ones and zeros odd. The other four bits are translated into nucleotides, with two bits per nucleotide. As mentioned earlier, it employs two error correction codes, the 8/4 Hamming-code and the WDH-code.

ASCII Based Coding Scheme. Another coding scheme implements the algorithm described by Jiao and Gouette [3] which inserts a message into the noncoding region of an existing DNA sequence. This method consists of several steps:

- 1) Convert each character in the message into its ASCII representation.
- 2) Convert the ASCII code from decimal into binary.
- 3) Converting binary to DNA by replacing 00 with A, 01 with C, 10 with G, and 11 with T.
- 4) Insert message into a carrier DNA sequence.

Steps 1-3 are referred to as the ASCII coding scheme throughout the remainder of this paper. The fourth step can be applied to other coding schemes if the message is to be inserted into a coding DNA region. This insertion is performed by replacing the last bits of redundant codons in the carrier sequence with characters from the message sequence. The ASCII coding scheme makes it possible to encode uppercase letters,

lowercase letters, numbers, and special characters. Each character is represented by a sequence of four bases.

There are $4! = 24$ different ways to choose which two-bit binary sequence is translated into which nucleotide. The DNA-Crypt coding scheme uses 00=T; 01=G; 10=C; 11=A [10], whereas the ASCII coding scheme uses 00= A; 01= C; 10= G; 11= T [3].

Yachie Coding Scheme. Yachie et al. [8] developed a coding scheme similar to ASCII encoding. Instead of ASCII it uses the keyboard scan code for each character. The keyboard scan code is the data that the keyboard sends to the computer to indicate which key has been pressed. This code, which is hexadecimal, is converted into binary, and then translated into DNA using a coding table.

Arita Coding Scheme. Arita and Ohashi [7] translated each letter of the English alphabet as well as an empty space and the characters ‘‘, ‘.’, ‘&’ into a 6-bit binary sequence. One of the bits serves as parity bit by keeping both the number of 0s as well as the number of 1s odd for error detection. When a message encoded with this coding scheme is inserted into a coding region of a DNA sequence, a 0 indicates to leave the 3rd base of a codon unchanged, while a 1 indicates that it needs to be changed. In order to extract the encoded message, one needs to compare the sequence that contains the message with the original, unchanged sequence to determine if a base was changed or not [10].

Coding Scheme Based on the Huffman Code. Another coding scheme is based on the Huffman code developed by David A. Huffman [22] and the frequency of letters in the English language from ‘‘The Code Book’’ by Simon Singh [23]. The Huffman

Table 4. Letter frequency in English language and DNA coding scheme using Huffman code [19]

Letter	Freq (%)	Codon	Letter	Freq (%)	Codon
e	12.7	T	w	2.4	AAT
t	9.1	AG	m	2.4	ACA
a	8.2	AT	f	2.2	ACG
o	7.5	GA	y	2.0	ACC
i	7.0	GG	g	2.0	ACT
n	6.7	GC	p	1.9	CCA
s	6.3	GT	b	1.5	CCG
h	6.1	CA	v	1.0	CCT
r	6.0	CG	k	0.8	CCCA
d	4.3	CT	j	0.2	CCCG
l	4.0	AAA	x	0.2	CCCC
c	2.8	AAG	q	0.1	CCCTA
u	2.8	AAC	z	0.1	CCCTG

code is an entropy encoding algorithm used for lossless data compression. The coding scheme based on this code currently only encodes letters, but not numbers or special characters. The average codon length is 2.2 bases. There are 4! possible ways to generate a Huffman code for encoding the 26 letters of the English alphabet, but it is also possible to create a Huffman code based scheme that includes numbers and special characters.

Comma Code. The comma code uses four base codons consisting of combinations of A, C, G and T, where G serves as a separator between the different characters. The term comma code may be misleading. It does not mean that G is the encoding for the comma character, but that it separates the encodings for each character. Smith et al. [19] suggest using five base codons with a separator every sixth base, but the original paper by Brenner et al. [12] is more descriptive and recommends the use of four bases per codon and a vocabulary made up of eight four-base “words” for biochemical reasons. The gaps between the Gs are filled with TTAC, AATC, TACT, ATCA, ACAT, TCTA, CTTT, or CAAA. Since this method would only allow the encoding of eight characters, combinations of two such words separated by a G are used for each character. This approach results in a total of 64 possible characters consisting of ten nucleotides each. The comma code encodes lowercase letters, numbers from 0-9, and special characters. The mapping of codons to characters was arbitrarily constructed. A sequence in comma code can easily be identified as containing a message, due to the occurrence of G every five bases, including the beginning and the end of the sequence. The comma code is the least efficient coding algorithm.

Alternating Code. The alternating code uses 64 codons with six bases per codon, alternating between purines (A or G) at odd positions and pyrimidines (C or T) at even positions. This coding scheme creates a pattern that does not occur naturally and can easily be recognized. For the same reason, the bases could be arranged for example in a pattern that has three purines followed by three pyrimidines or vice versa. The alternating code encodes the same characters as the comma code. The decision which codon codes for which character was made arbitrarily [19].

Summary of Coding Schemes. The coding schemes differ in codon length, detectability, number of characters that can be encoded, and the number of steps involved in encoding a message. The Huffman code is the most economical in terms of codon length, while the comma code is the least economical. The Clelland coding scheme and the Wong coding scheme are the easiest to implement. The comma code, alternating code and DNA-Crypt are the easiest to detect, and DNA-Crypt offers the best error correction.

Inserting a message into a coding region only replaces bases, but does not add new ones. Therefore the size of the genome is only affected if the message is inserted into a non-coding region. The length of the encoded message is the length of the unencoded message multiplied by the codon length of the coding scheme. For example, the message “UNIVERSITY OF LOUISVILLE” is 24 characters long, including spaces. It would be

CTGCCTCAGCTTATGCGTCTACAGCTCGAGCGAATCCCCGACTGCAGCTACT
 TCAGCCCCCATG, which is 72 characters in Wong's coding scheme and
 GTACTGACATGAATCGACATGAATCGTTACGTACTGTCTAGTTACGACATGTACTGAA
 TCGTACTGTACTGAATCGTTACGTACTGATCAGATCAGTTACGAATCGCTTTGTTACG
 TCTAGAATCGATCAGAATCGCTTTGTACTGACATGAATCGTTACGTACTGTACTGTAC
 TGTCTAGAATCGTTACGAATCGATCAGAATCGATCAGTTACGACATG, which is 240
 characters in Comma Code.

For inserting pictures, audio, and video files into a DNA sequence it would be best to translate the binary representation of the file into DNA code, with each base encoding two bits, for example A=00, C=01, G=10, and T=11.

2.3 Encryption and Watermarking of DNA Messages

To make detection even more difficult, it is possible to encrypt a message using modern encryption algorithms such as Data Encryption Standard (DES), RSA, and Number Theory Research Unit (NTRU) before encoding it into DNA.

One application for inserting messages into DNA is watermarking. This method can help establish brand names for engineered bacteria strains in order to resolve legal disputes regarding gene related patents [7]. Watermarking infectious agents can be useful for tracking them back to their source after an accidental release [24].

Researchers at the JCVI inserted four watermarks using a Category 1 coding scheme similar to Clelland's and Wong's into their artificial genome. The first watermark consists of a copyright like statement; the coding table for Ventner's coding scheme, and a hidden HTML page. The second, third, and fourth watermarks consist of a list of the authors and three quotations.

The coding scheme created by Arita and Ohashi [7] and the DNA-Crypt algorithm developed by Heider and Barnekow [10] were both designed for watermarking short trademarks or signatures into genomic DNA.

2.4 Messages Finding Data in DNA

Steganalysis is the process of discovering hidden messages [25]. There are two main categories of steganalytic methods: blind steganalysis and specific steganalysis. Blind steganalysis can be used to detect a variety of different steganographic algorithms, including previously unknown ones. The goal of specific steganalysis is to detect a specific known steganographic algorithm by exploring how this particular algorithm works and how it changes the statistics of the cover media [26].

The research on steganalysis is important for several reasons: First, detecting the presence of secret messages can help intercept communication between members of terrorist organizations or other illegal groups. Second, improvements in steganalysis also help to develop better methods for information hiding. Third, better statistical methods for multimedia contents can emerge as a byproduct of steganalysis research. These can then be applied in other related research fields, such as digital forensics [26], or bioinformatics.

Most existing steganalysis approaches focus on images as a stegomedium, especially JPEG images as well as audio and video files. Text documents are not used as often since they can only hold a smaller amount of information than a graphic document with same amount of carrier data. However, text files are still used because they are easily edited, stored, and transferred [27].

2.5 Experiments Performed *in Silico*

Wang and Zhang [28] have developed a software called WordSpy to detect certain biological features within a genome. This software regards these biological features of a genome as a message hidden in a cover-text of genomic sequences. A Hidden Markov Model is used to decipher the message and to extract over-represented motifs. WordSpy combines word counting and statistical modeling to detect frequently occurring sub-sequences [28].

Since many different coding schemes for inserting messages into DNA have been developed, we decided to develop a software toolkit that would enable us to insert and extract messages from DNA sequences, allow us to compare different coding schemes, and serve as basis for research into developing methods to find and extract messages encoded with unknown coding schemes.

2.6 Solving Substitution Ciphers

The way messages are encoded in DNA is typically through the use of substitution ciphers, for example, the letter 'a' is substituted by the sequence 'AAA', the letter 'b' by 'AAC', and so on. Several methods have been developed for breaking substitution ciphers. One of our goals is to adapt an algorithm for breaking substitution ciphers to decode a message written in DNA symbols. Almost all approaches use n-grams of letters.

Spillman et al. [29] developed a Genetic Algorithm and although they report good results for their Genetic Algorithm, Delman [30] found Genetic Algorithms to be unreliable for solving substitution ciphers and was unable to reproduce the results.

Another software called Quipster has been developed by Hasinoff [31]. The software decodes a median of 94% of the cipher letters correctly.

A Particle Swarm Optimization (PSO) algorithm has been developed by Uddin and Youssef [29]. Their results show that PSO provides a very powerful tool for the cryptanalysis of simple substitution ciphers using a ciphertext only attack. Uddin and Youssef [32] also investigated the use of Ant Colony Optimization (ACO) for automated cryptanalysis of classical simple substitution ciphers and found them to be very effective on various sets of encoding keys.

Lucks [33] developed an algorithm which employs an exhaustive search in a dictionary for words that satisfy constraints on word length, letter position and letter multiplicity. His method is not restricted to English and can be used for any language.

It is especially difficult to decode short ciphers, because they have different distribution statistics than larger texts. Hart [34] developed a method that addresses these problems by using whole words instead of n-grams and by employing a maximum likelihood estimator.

Jakobsen [35] developed a fast algorithm that is based on a process where an initial key guess is refined through a number of iterations. Each step of this algorithm evaluates the plaintext corresponding to the current key and the result is used as a measure of how close the algorithm is to discovering the correct key. The author claims that only knowledge of the bigram distribution in the ciphertext and the expected bigram distribution in the plaintext is necessary in order to decipher the message. The algorithm currently only uses bigrams, but the author suggests using trigrams or whole words for future research.

Forsyth and Safavi-Naini [36] approached the solving of substitution ciphers as a combinatorial optimization problem and developed an algorithm that uses simulated annealing. This algorithm is very complicated and difficult to implement, but it is very successful at decrypting ciphertexts, especially ones with over 5000 letters.

Peleg and Rosenfeld [37] address it as a probabilistic labeling problem and assigned probabilities of representing plaintext letters to every code letter. This approach was done by using joint letter probabilities. These probabilities were updated in parallel for all code letters, and using this scheme iteratively, they were able to break the cipher.

3 Description of DNA-Steg

3.1 Encoding and Inserting Messages

The DNA steganography and steganalysis toolkit we developed currently consists of two programs. One for encoding a message in a DNA sequence (steganography), and the other for detecting and extracting a hidden message from a DNA sequence (steganalysis).

Our software offers a choice of several different coding schemes. It reads in the coding table for the selected coding scheme from a file and then prompts the user to either type the message to be encoded on the keyboard or to read it in from a file. Since the ASCII coding scheme is the only one that distinguishes between uppercase and lowercase characters, the program converts all characters in the message into uppercase characters for all coding schemes other than the ASCII coding scheme. The steganography program then encodes the message using the appropriate coding table.

The toolkit implements the following coding schemes:

- Huffman code based scheme [19]
- Alternating code [19]
- Comma code [12, 19]
- Wong's coding scheme [20]
- Clelland's coding scheme [11]
- DNA-Crypt [10]
- ASCII coding scheme [3]

Coding tables for comma code and alternating code were created arbitrarily, since the original researchers did not provide any.

The message can either be directly written to a file by itself if it is to be stored in a noncoding region, or be inserted into the coding region of an existing DNA sequence file. For inserting a message in a coding region, the algorithm described by Jiao and Gouette [3] is used. DNA sequences can be chosen from a folder where they are stored in FASTA format [24], which is widely used in bioinformatics. The program displays the maximum number of characters a message can have, depending on the coding scheme and the sequence it is to be inserted into.

3.2 Approaches to Detecting Messages in DNA

Finding a message that has been inserted into the coding region of a DNA sequence is relatively simple if the original sequence is known. We have developed a program which compares a modified DNA sequence with its original. Since the message is assumed to have been inserted into wobble bases, the first step is to identify wobble base codons in both sequences and to compare them to each other. The first codon where the wobble base is different from the one in the original is identified as the beginning of the message. The last codon where it differs is marked as the end of the message.

The limitation in this approach is that there are codons where the wobble base does not change. This is not a problem if it happens in the middle of the message. The program therefore assumes it contains one long message instead of several smaller ones. Problems can arise if this happens at the beginning or end of the message, but if the message can be decoded and it is seen that pieces are missing, the program can be expanded to go back and fix it.

In order to test the program, the message “THIS IS A TEST” was inserted into *ftsZ* using the Wong coding scheme. The program then compared the modified sequence with the original one. It correctly identifies the beginning codon and the end codon of the message and extracts the modified wobble bases.

Finding a message in a noncoding DNA region is much more difficult. But there are ways to determine if a DNA sequence is artificial by statistical analysis. For example, if a certain base is significantly overrepresented, underrepresented, or not present at all, it can be assumed that the sequence is artificial and should be further analyzed to determine if it may contain a message.

Messages that have been encoded using a variation of the alternating code or the comma code are more likely to be detected than messages that were encoded with a different coding scheme. The reason for that is that they have a repeating pattern, which can be detected by a human or a computer program. If every n^{th} base is the same, this hints at the possibility that comma code or a variation thereof has been used to create this sequence.

Another coding scheme that is easy to identify is the DNA-Crypt coding scheme because the low occurrence of As in a message encoded with this scheme.

However, as a countermeasure against attempts to detect messages by counting the occurrence of nucleotides, a coding scheme such as the one developed by Modego [38] can be used. Modego’s coding scheme uses two codons to encode each letter. Which codon is used is determined by the GC content of the carrier sequence. For

example, if a message was to be inserted into a sequence with a high GC content, the letter L would be encoded as CTG, but in a sequence with low GC content it would be TTA [38]. The obvious tradeoff is the number of characters that can be encoded is cut in half.

In order to detect the alternating code, the program stores all odd position characters in one list and all even position characters in another and then compares both of them. If none of the even characters appears in the list with the odd ones and vice versa, the program has detected a message in alternating code with the pattern XYXYXY, where X is either an A or a G and Y a C or a T, or vice versa. The program can easily be extended to detect alternating codes with pattern XYYXXYYY or XXXYYY.

Currently the steganalysis part reads messages from a DNA sequence by executing the algorithm that was used for encoding the message in reverse order. Since it does not know which coding scheme was used, it uses a brute force approach to test all supported coding schemes and displays the resulting message on the screen. The user can choose if the message was hidden in a noncoding region or in an existing DNA sequence, which can be selected from a folder. If a sequence is chosen, the program will display the number and the percentage of occurrences of each nucleotide in the altered sequence as well as in the original sequence. Usually there is not much difference in the statistics of both sequences, since the inserted messages are fairly small.

One possible way to detect unknown encoding schemes might be to use the WordSpy algorithm developed by Wang and Zhang [28].

4 Further Research

We are currently working on a way to modify an existing approach for solving simple substitution ciphers where each letter in the English alphabet is substituted for a different English letters to solving simple substitution ciphers in which each letter of the English alphabet, numbers from 0-9, and several special characters such as spaces, commas, and periods are each substituted by a combination of three DNA bases. While the original program searches over the space of 26! possible keys, our program will have a search space of 64! possible keys.

The goal is to modify several existing approaches and then use the Wisdom of Artificial Crowds (WoAC) [39, 40] post-processing algorithm instead of brute force guessing in order to find out with which coding scheme the message has been encoded in. As a proof of concept, a program will be developed that will be able to decode any message that has been encoded with a category 1 coding scheme of codon length 3. This approach can be adapted to be used for other coding schemes as well.

Currently the steganalysis tool can only detect and extract messages that have been encoded with the previously described coding schemes. For example, it only has two coding tables for coding schemes of category 1, namely the ones developed by Clelland [11] and Wong [20]. But since there are four nucleotides, a category 1 coding scheme with a codon length of three, which can encode 64 characters, can be generated in 64! possible ways. And that is only if the same 64 characters are being

used. For example, one coding scheme can start with A=AAA, B=AAC, C=AAG... while another one could be A=AGT, B=CCG, C=CTG... Brute force guessing which variant has been used to encode the message would take an enormous amount of time and would therefore not be feasible.

Knowing the characteristics of several previously developed coding schemes and their different variations makes it possible to develop a more optimized coding scheme. For example, coding scheme that is based on the Huffman code could be expanded to include numbers and special characters.

We also need to develop methods to detect and decode a message if the encoded message is also encrypted.

5 Conclusion

DNA steganography is a new field and therefore it offers many opportunities to improve upon existing approaches for steganography as well as steganalysis.

References

1. Anam, B., Sakib, K., Hossain, A., Dahal, K.: Review on the Advancements of DNA Cryptography. In: International Conference on Software, Knowledge, Information Management and Application, Paro, Bhutan, August 25-27 (2010)
2. Nirenberg, M.: Historical review: Deciphering the genetic code – a personal account. *Trends in Biochemical Sciences* 29(1), 46–54 (2004)
3. Jiao, S.-H., Goutte, R.: Code For Encryption Hiding Data Into Genomic DNA. In: International Conference on Software Process (2008)
4. Adleman, L.M.: Molecular Computation of Solutions To Combinatorial Problems. *Science, New Series* 266(5187), 1021–1024 (1994)
5. Ogihara, M., Ray, A.: Simulating Boolean Circuits on a DNA Computer. *RECOMB* (1997)
6. Bogard, C.M., Rouchka, E.C., Arazi, B.: DNA media storage. *Progress in Natural Science* 18, 603–609 (2007)
7. Arita, M., Ohashi, Y.: Secret Signatures Inside Genomic DNA. *Biotechnology Progress* 20(5), 1605–1607 (2004)
8. Yachie, N., Sekiyama, K., Sugahara, J., Ohashi, Y., Tomita, M.: Alignment-Based Approach for Durable Data Storage into Living Organisms. *Biotechnology Progress* 23(2), 4 (2007); (Epub January 25, 2007)
9. Arita, M.: Comma-free design for DNA words. *Communications of the ACM* 47(5), 99 (2004)
10. Heider, D., Barnekow, A.: DNA-based watermarks using the DNA-Crypt algorithm. *BMC Bioinformatics* 8, 176 (2007) (Epub May 31, 2007)
11. Clelland, C.T., Risca, V., Bancroft, C.: Hiding messages in DNA microdots. *Nature*, 533–534 (1999)
12. Brenner, S., Williams, S.R., Vermaas, E.H., Storck, T., Moon, K., McCollum, C., et al.: In vitro cloning of complex mixtures of DNA on microbeads: Physical separation of differentially expressed cDNAs. *Proceedings of the National Academy of Sciences of the United States of America* 97(4), 1665–1670 (2000)

13. Jiao, S.-H., Goutte, R.: Hiding data in DNA of living organisms. *Natural Science* 1(3), 181–184 (2009)
14. Heider, D., Barnekow, A.: DNA watermarks: A proof of concept. *BMC Molecular Biology* 9, 40 (2008); (Epub April 23, 2008)
15. Gibson, D.G., Glass, J.I., Lartigue, C., Noskov, V.N., Chuang, R.Y., Algire, M.A., et al.: Creation of a bacterial cell controlled by a chemically synthesized genome. *Science* 329(5987), 52–56 (2010); (Epub May 22, 2010)
16. Bancroft, C., Bowler, T., Bloom, B., Clelland, C.T.: Long-Term Storage of Information in DNA. *Science, New Series* 293(5536), 1763–1765 (2001)
17. A Y3K bug.pdf. *nature biotechnology* 18 (2000)
18. Drake, J.W., Charlesworth, B., Charlesworth, D., Crow, J.F.: Rates of Spontaneous mutation. *Genetics* 148(4), 20 (1998)
19. Smith, G.C., Fiddes, C.C., Hawkins, J.P., Cox, J.P.L.: Some possible codes for encrypting data in DNA. *Biotechnology Letters* 25(14), 1125–1130 (2003)
20. Wong, P.C., Wong, K.-K., Foote, H.: Organic Data Memory Using the DNA Approach. *Communications of the ACM* 46(1), 95–98 (2003)
21. Tanaka, K., Okamoto, A., Saito, I.: Public-key system using DNA as a one-way function for key distribution. *Bio Systems* 81(1), 25–29 (2005); (Epub May 27, 2005)
22. Huffman, D.A.: A Method for the Construction of Minimum-Redundancy Codes. In: *Proceedings of the IRE*, pp. 1098–1102 (1952)
23. Singh, S.: *The Code Book: The Evolution of Secrecy from Mary, Queen of Scots to Quantum Cryptography*. Doubleday, New York (1999)
24. Jupiter, D.C., Ficht, T.A., Qin, Q.-M., de Figueiredo, P.: DNA Watermarking of Infectious Agents Progress and Prospects. *Public Library of Science Pathogens* 6(6), 1–3 (2010)
25. Johnson, N.F., Jajodia, S.: Steganalysis: The Investigation of Hidden Information. In: *IEEE Information Technology Conference*, Syracuse, NY (1998)
26. Li, B., Huang, J., Shi, Y.Q.: Steganalysis of YASS. *IEEE Transactions on Information Forensics and Security* 4(3), 369–382 (2009)
27. Xin-guang, S., Hui, L., Zhong-liang, Z.: A Steganalysis Method Based on the Distribution of Characters.pdf. In: *8th International Conference on Signal Processing*, Beijing, China (2006)
28. Wang, G., Zhang, W.: A steganalysis-based approach to comprehensive identification and characterization of functional regulatory elements. *Genome Biology* 7(6), R49 (2006); (Epub June 22, 2006)
29. Spillman, R., Janssen, M., Nelson, B., Kepner, M.: Use of a Genetic Algorithm in the Cryptanalysis of Simple Substitution Ciphers. *Cryptologia* 17(1), 31–44 (1993)
30. Delman, B.: *Genetic Algorithms in Cryptography*. Rochester Institute of Technology, Rochester (2004)
31. Hasinoff, S.: *Solving Substitution Ciphers*. A Technical Report, University of Toronto (2003)
32. Uddin, M.F., Youssef, A.M.: An Artificial Life Technique for the Cryptanalysis of Simple Substitution Ciphers. In: *CCECE+CCGEI*, May 7-10, pp. 1582–1585. IEEE, Ottawa (2006)
33. Lucks, M.: A Constraint Satisfaction Algorithm for the Automated Decryption of Simple Substitution Ciphers. In: Goldwasser, S. (ed.) *CRYPTO 1988*. LNCS, vol. 403, pp. 132–144. Springer, Heidelberg (1990)
34. Hart, G.W.: To decode short Cryptograms. *Communications of the ACM* 37(9), 102–108 (1994)

35. Jakobsen, T.: A fast method for cryptanalysis of substitution ciphers. *Cryptologia* 19(3), 265–274 (1995)
36. Forsyth, W.S., Safavi-Nani, R.: Automated Cryptanalysis of substitution ciphers. *Cryptologia* 17(4), 407–424 (1993)
37. Peleg, S., Rosenfeld, A.: Breaking Substitution Ciphers Using a Relaxation Algorithm. *Communications of the ACM* 22(11), 598–605 (1979)
38. Modegi, T.: Watermark Embedding Techniques for DNA Sequences Using Codon Usage Bias Features. In: 16th International Conference on Genome Informatics, Yokohama, Japan (2005)
39. Yampolskiy, R.V., El-Barkouky, A.: Wisdom of artificial crowds algorithm for solving NP-hard problems. *International Journal of Bio-Inspired Computation* 3(6) (2011)
40. Yampolskiy, R.V., Ashby, L.H.: Genetic Algorithm and Wisdom of Artificial Crowds Algorithm Applied to Light Up. In: The 16th International Conference on Computer Games, Louisville, KY, pp. 27–32 (2011)

On the Completeness of Reconstructed Data for Database Forensics

Oluwasola Mary Adedayo and Martin S. Olivier

ICSA, Department of Computer Science,
University of Pretoria, South Africa
{mfasan,molivier}@cs.up.ac.za

Abstract. Databases are often used to store critical and sensitive information in various organizations and this has led to an increase in the rate at which databases are exploited in computer crimes. Even though various investigations involving databases have been explored, very little amount of research has been done on database forensics. This paper briefly describes a database reconstruction algorithm presented in an earlier work and shows the limitation that can be encountered when the algorithm has to deal with partially reconstructed relations or the deletion of tuples in a relation. Since reconstructed data can often be used as the evidence to support or refute claims about the data in a database, the inability to reconstruct necessary data may imply the absence of evidence. However, according to an axiom from forensic science, this does not mean an evidence of absence. As such, this paper presents two different techniques that can be used in reconstructing more tuples in a relation and provide corroborating evidence to claims about the data on a database. A typical example is used to describe the limitation of the database reconstruction algorithm and how the limitation can be overcome by using the techniques described in the paper.

Keywords: Digital forensics, Database forensics, Database reconstruction algorithm, Digital evidence, Forensic science.

1 Introduction

The use of databases in today's commercial systems cannot be over-emphasized as databases have become a core component of many computing systems and are often used to store critical and sensitive information to an organization or her clients. Unfortunately, the increased usage of databases in storing volumes of information together with the increased relevance of the data on many databases in solving various crimes have led to an increase in the number of attacks directed towards databases and interests investigating databases for artifacts that may assist in solving various different crimes.

Database forensics is an emerging branch of digital forensics [16,1] that deals with the identification, preservation, analysis and presentation of evidence from databases [7]. Even though digital forensics has grown over the last decade from a

relatively obscure trade-craft to an important part of many investigations [8], the same cannot be said of database forensics, despite the importance of databases. Although a large amount of research has been done on digital forensics, database theory and database security, very little has been done on database forensics [15] even though investigations involving databases have been explored in theory and in practice. Similar to other branches of digital forensics, database forensics helps in determining the root cause of an attack and finds out what was done. An important aspect of database forensics deals with the ability to revert data manipulation operations and determine values contained in a database at an earlier time.

Although various data restoration techniques such as rollback and incremental backups have been explored over the years, these techniques are sometimes inadequate for database forensics. For example, a rollback operation can only be used provided that the transaction has not been committed and the use of incremental backups is dependent on the availability of viable backups from which data can be restored. Database forensics requires the ability to revert data manipulation operations even when a transaction has been long committed or when there are no viable backups.

In our earlier work [6], we presented an algorithm that can be used for reconstructing the information in a database at an earlier time of interest. The algorithm makes use of the inverse functions of the relational algebra [4] and incorporates the notion of value blocks (a group of queries whose evaluation does not change the information in a particular relation). The inverse of a query is found by taking the database schema and the log of modifying queries performed on the database into consideration. In another work [5], we prove that the relation generated from the algorithm is always correct. That is, even though the reconstructed relation may be incomplete if compared with the original relation, it is at least a subset of the original relation.

The generation of incomplete relations or inability to reconstruct values of interest in a relation when using the database reconstruction algorithm [6] stems from the fact that the inverse generated from some of the inverse operators of the relational algebra may be missing one or more tuples or values in a column of the original relations. It also implies that the evidence needed from a database during an investigation may not be found. However, this does not imply that such evidence does not exist. The objective of this paper is to discuss the limitation of the database reconstruction algorithm and describe some of the techniques that can be applied in conjunction with the algorithm in order to generate more complete relations or tuples of a relations as well as provide corroborating evidence regarding claims about the information on a database at an earlier time. The paper describes a typical application of the reconstruction algorithm that reflects its limitation. It also discusses two different techniques of reconstructing more information from a database using the reconstruction algorithm.

2 Background and Notation

This section gives a brief background on database forensics and introduces the relational model of database management systems (DBMS) and its operators. It also describes the notation used in the rest of the paper.

2.1 Database Forensics

As earlier mentioned, database forensics often requires the determination of the information in a database at an earlier time. Although the information in a database at any instance can be determined by querying the database, much more effort is required in order to determine the information contained at an earlier time since various modifications might have occurred. Some of the little work that has been done in database forensics include the series of papers by Litchfield [9,10,11,12,13,14] all of which focus on Oracle forensics. Wright [18] published a book that also explains Oracle forensics and investigates the possibility of using Oracle LogMiner as a forensics tool [17]. Another book by Fowler [7] focuses on SQL server database forensics and discusses the effect of rootkits on data collection and analysis during the forensics investigation of an SQL server database. None of these works describes the process of reconstructing the information in a database at an earlier time.

The ability to reconstruct the information on a database at earlier time is an important aspect of database forensics. An illustration of this fact, which requires forensics investigation is a situation where a sales representative claims to have sold a large quantity of a certain good at the selling price on the database at a particular date even though the price presents a huge loss to the organization. Verifying the representative's claim requires that the selling price of the good at that particular date can be determined even though several modifications/updates might have been performed on the database which might have affected the price of the good since the date of interest.

In our earlier work, we present an algorithm [6] that can be used to determine the information contained in a database at an earlier time. Although our focus in this paper is to discuss the completeness of the result generated from the algorithm and how this can be improved, it is important to introduce the notion of inverse relational algebra and value blocks which are a major component of the algorithm. The relational algebra [4] is employed since it represents a fundamental aspect of databases and gives a formal description of how the information stored in a database relate with each other.

2.2 Inverse Relational Algebra

The relational model for DBMS was developed by Codd [4] and works on the relational theory of mathematics. The model is composed of one type of compound data known as a relation. Given a set of domains, $D = D_1, D_2, \dots, D_n$ over which attributes $A = A_1, A_2, \dots, A_n$ are defined respectively, a relation R (also called an *R-table* or $R(A)$) is a subset of the Cartesian product of the

Table 1. Inverse Operators of the Relational Algebra

Operators	Query	Inverse Operators
Rename (ρ)	$R \leftarrow \rho_{A_i=B_j}(R)$	$\rho^{-1}(R) = \rho_{B_j=A_i}(R)$
Cartesian product (\times)	$T \leftarrow R(A) \times S(B)$	$\times^{-1}(T) = (R, S)$ where $R = \pi_A(T)$ and $S = \pi_B(T)$
Union (\cup)	$T \leftarrow R \cup S$	$\cup^{-1}(T) = (R^*, S)$ where $R^* = T - S$ provided that S is known and vice versa
Intersection (\cap)	$T \leftarrow R \cap S$	$\cap^{-1}(T) = (R^*, S^*)$ where $R^* = S^* = T$
Difference ($-$)	$T \leftarrow R - S$	$-^{-1}(T) = R^* = T$. If R is known, $S^* = R - T$.
Division ($/$)	$T \leftarrow R/S$	$/^{-1}(T) = (R^*, S^*)$ where $R^* = RM$ and RM is the remainder of the division
Join (\bowtie)	$T \leftarrow R(A) \bowtie_{p(A,B)} S(B)$ $T \leftarrow R \bowtie_{p(A,B)} S$	$\bowtie^{-1}(T) = (R^*, S^*)$ where $R^* = \pi_A(T)$ and $S^* = \pi_B(T)$
Projection (π)	$T \leftarrow \pi_{A_1, A_2, A_3}(R)$ $T \leftarrow R[A_1, A_2, A_3]$	$\pi^{-1}(T) = S^* = T$
Selection (σ)	$T \leftarrow \sigma_{p(A)}(R)$ $T \leftarrow R[p(A)]$	$\sigma_{p(A)}^{-1}(T) = S^* = T$

domains. A relation can be conceived as a table where the columns are the attributes, the rows are referred to as tuples and the domains define the data types of the attributes.

The relational algebra consists of basic operators used to manipulate relations and a relational assignment operator \leftarrow . The basic operators transform either one or two relations into a new relation. Such transformations are known as relation-valued expressions (*rve*). A query is defined in the form $T \leftarrow rve$, where T is the name of the relation obtained when the *rve* is evaluated. The basic operators as defined by Codd [4] and the corresponding notation often used are shown in the second column of table 1, where R, S , and T are relations and A, B and C are attributes of these relations. The notation $p(\text{attributes})$ is a logical predicate on one or more attributes representing a condition that must be satisfied by a row before the specified operation can be performed on it.

The inverse operators of the relational algebra work on the assumption that the database schema is known. The aim of the inverse operators is to find the value of one or more attributes of a relation at a specific time t by finding the inverse of the most recent query performed on the current relation R_t sequentially until the desired time t_i is reached. The output generated by an inverse operator may either be partial or complete when compared to the original relation. That is, the inverse of a query Q can be defined as Q^{-1} such that,

$$Q^{-1}(Q(R_t)) = R_t^*$$

where R_t^* is a subset of R_t . That is, some tuples or values that should be in some columns of R_t^* may be missing¹. In cases where $R_t^* = R_t$, we refer to the inverse found as a complete inverse. Otherwise, we refer to the inverse found as a partial inverse. A partial inverse can be a partial tuples inverse and/or a partial columns inverse depending on whether some of the tuples or values in some columns of the original relation are missing, respectively. However, regardless of the classification of the inverse operators, there are often instances where a complete inverse can be found with some of the operators grouped as partial inverse. A summary of the inverse operators of the relational algebra and how inverses are computed is given in table 1. More details about relational algebra, the inverse operators and instances where a complete inverse can be generated from the inverse operators classified as partial inverses can be found in earlier works [6,5].

2.3 Relational Algebra Log and Value Blocks

A relational algebra log (RA log) is a log of queries expressed as relational algebra operations instead of the traditional SQL notation. The use of the RA log allows us to easily determine when a relation has been modified. Using the relational algebra notation, a relation is changed only when a new assignment operation is made into the relation. This knowledge allows us to group the RA log into a set of overlapping *value blocks*. Another advantage of using the RA log instead of the usual SQL log file is that relational algebra allows queries to be represented as a sequence of unary and binary operations involving relational algebra operators. Thus, the log file is more readable. For example, a typical select statement in a SQL log file can take several forms; however, the use of the RA log eliminates ambiguities that may arise in defining an inverse for select statements since any select statement can be expressed with relational algebra operators.

A value block is defined as a set of queries within which a particular relation remains unchanged. Value blocks are named based on the relation that remains the same in the block and subscripts are used to signify which value block occurs first. A value block starts with an assignment operation into the relation and ends just before another assignment operation into the relation is encountered. For example, the value block of a relation R is denoted as V_{R_i} where $i = 1, 2, 3, \dots$. The relation R remains the same throughout the execution of block V_{R_1} until it is updated by the execution of the first query of block V_{R_2} . Thus, the value block of a relation can be contained in or overlap that of another relation, so that V_{R_1} and V_{S_2} can have a number of queries in common. However, two value blocks of the same relation, V_{R_1} and V_{R_2} cannot overlap or be a subset of the other [6]. The time stamps usually associated with the traditional query log is preserved in the RA log in order to group the value blocks into appropriate sequences. An example of a RA log generated from a traditional log file and divided into value blocks is shown in figure 1. Subsequent examples in the paper will refer to this RA log.

¹ The mapping generated by queries are not usually a bijection. However, this does not mean that some inverses cannot be found.

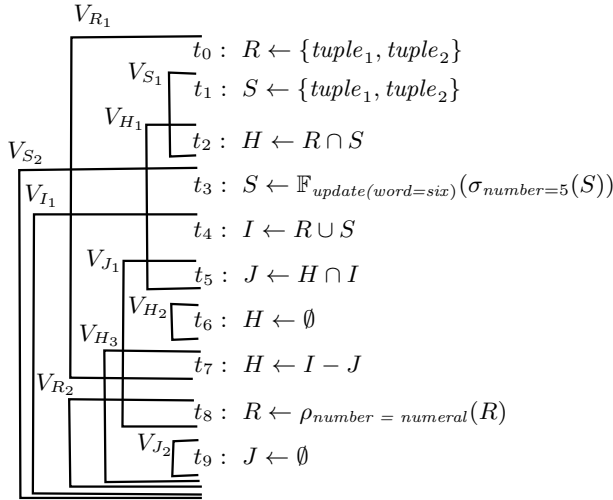


Fig. 1. A Relational Algebra Log Grouped into Value Blocks

2.4 Database Reconstruction Algorithm

The database reconstruction algorithm employs the notion of RA logs and value blocks as well as the inverse operators of the relational algebra. In this section, we give a brief description of the reconstruction algorithm necessary to understand the content of this paper, a more detailed explanation of the algorithm can be found in [6].

```

01: INVERSE(Relation D, RA Query  $V_{D_i}[1]$ ) {
02: OUTPUT: Inverse of the assignment into D from query q
03: Let q = the query at  $V_{D_i}[1]$ ;
04: switch(q) {
05:   case ( $D \leftarrow \emptyset$ ):
06:      $T = \emptyset$ ; return T;
07:   case ( $D \leftarrow \text{op } D$ ):
08:      $T = \text{op}^{-1}(D)$ ; return T;
09:   case ( $D \leftarrow A \text{ op } D$ ):
10:   case ( $D \leftarrow D \text{ op } A$ ): //Assume A is in  $V_{A_i}$ 
11:     if (op =  $\cap$ ):  $T = D$ ; return T;
12:     if ((op =  $\cup$ ) and ( $\exists V_{A_{i+1}}$ ):  $T = \emptyset$ ; return T;
13:     else:
14:        $A \leftarrow \text{SOLVE}(A, V_{A_i}, \text{log}, S)$ ;
15:        $T = \text{op}^{-1}(D)|A$ ; return T;
16:   }
17: }
```

Fig. 2. The INVERSE(Relation D, RA Query $V_{D_i}[1]$) function

Although the algorithm is aimed at reconstructing specific values in a relation on a database at some earlier time, it can also be applied to generate tables in a database. The algorithm assumes that the query log exists, and contains the complete set of modifying queries that have been performed on the database from at least the particular time of interest (or earlier) to the present time.

The reconstruction algorithm consists mainly of two functions: the **inverse** and the **solve** functions. The **solve** function makes use of the **inverse** function (shown in figure 2) which takes as input the name of the relation to be reconstructed (D) together with a query, specifically the first line of a value block of D (denoted as $V_{D_i}[1]$) and finds the inverse of the query in order to determine

```

SOLVE(Relation  $D$ , Value Block  $V_{D_i}$ , RA Log  $log$ , Set  $S$ )
OUTPUT: Reconstructed relation  $D$  in value block  $V_{D_i}$  ( $RD$ )
01: Let  $Q$  = Set of queries involving relation  $D$  in value block  $V_{D_i}$ ;
02: Let  $R$  = Set to reconstructed  $D$  from different approaches;
03: If  $(D, V_{D_i}, RD) \in S$ : return  $RD$ 
04: else:
05:  $S = S \cup (D, V_{D_i}, RD)$ ; //  $RD$  is initialized as an empty relation
06: for each element  $e$  in  $Q$ :
07:   switch( $e$ ) {
08:     case  $(D \leftarrow op D)$ :
09:       if  $(\notin V_{D_{i+1}})$ : return  $D$ ;
10:       else:
11:          $D \leftarrow SOLVE(D, V_{D_{i+1}}, log, S)$ ;  $T \leftarrow INVERSE(D, V_{D_{i+1}}[1])$ ;
12:         Insert  $T$  into  $R$ 
13:         OR
14:          $D \leftarrow SOLVE(D, V_{D_{i-1}}, log, S)$ ;  $T \leftarrow op D$ ;
15:         Insert  $T$  into  $R$ 
16:     case  $(D \leftarrow op A)$ : // Assume  $A$  is in  $V_{A_i}$ 
17:       if  $(\notin V_{D_{i+1}})$ : return  $D$ ;
18:       else:
19:         if  $(\notin V_{A_{i+1}})$ :
20:            $D \leftarrow op A$ ; return  $D$ ;
21:         else:
22:            $A \leftarrow SOLVE(A, V_{A_{i+1}}, log, S)$ ;  $A \leftarrow INVERSE(A, V_{A_{i+1}}[1])$ ;
23:            $D \leftarrow op A$ ; return  $D$ ;
24:     case  $(D \leftarrow A op D)$ :
25:     case  $(D \leftarrow D op A)$ : // Assume  $A$  is in  $V_{A_i}$ 
26:       if  $(\notin V_{D_{i+1}})$ : return  $D$ ;
27:       else:
28:          $D \leftarrow SOLVE(D, V_{D_{i+1}}, log, S)$ ;  $T \leftarrow INVERSE(D, V_{D_{i+1}}[1])$ ;
29:         Insert  $T$  into  $R$ ;
30:         if  $(\notin V_{A_{i+1}})$ :
31:            $D \leftarrow SOLVE(D, V_{D_{i-1}}, log, S)$ ;
32:            $T \leftarrow A op D$  or  $(D op A)$ ; //depending on case
33:           Insert  $T$  into  $R$ ;
34:         else:
35:            $D \leftarrow SOLVE(D, V_{D_{i-1}}, log, S)$ ;
36:            $A \leftarrow SOLVE(A, V_{A_i}, log, S)$ ;
37:            $T \leftarrow A op D$  or  $(D op A)$  //depending on case
38:           Insert  $T$  into  $R$ ;
39:         OR
40:          $D \leftarrow SOLVE(D, V_{D_{i-1}}, log, S)$ ;
41:          $A \leftarrow SOLVE(A, V_{A_{i+1}}, log, S)$ ;  $A \leftarrow INVERSE(A, V_{A_{i+1}}[1])$ ;
42:          $T \leftarrow A op D$  or  $(D op A)$ ; //depending on case
43:         Insert  $T$  into  $R$ ;

```

Fig. 3. The SOLVE function

```

44: case (G ← op D): //Assume G is in VGi
45:   if (∄ VDi+1): return D;
46:   else:
47:     if (∄ VGi+1):
48:       T ← op-1(G); Insert T into R;
49:     else:
50:       D ← SOLVE(D, VDi+1, log, S); T ← INVERSE(D, VDi+1[1]);
51:       Insert T into R;
52:     OR
53:       G ← SOLVE(G, VGi+1, log, S); G ← INVERSE(G, VGi+1[1]);
54:       T ← op-1(G); Insert T into R;
55: case (G ← D op A):
56: case (G ← A op D): //Assume G and A are in VGi and VAi respectively
57:   if (∄ VDi+1): return D;
58:   else:
59:     if (∄ VGi+1):
60:       if (op = ∩):
61:         Insert G into R;
62:       if (op ≠ ∪):
63:         T ← op-1(G)[1]; //D is at index 1 in the output of op-1(G)
64:         Insert T into R;
65:       if (∄ VAi+1):
66:         T ← op-1(G)|A; Insert T into R;
67:       else:
68:         A ← SOLVE(A, VAi+1, log, S); A ← INVERSE(A, VAi+1[1]);
69:         T ← op-1(G)|A; Insert T into R;
70:     else:
71:       if (∄ VAi+1):
72:         G ← SOLVE(G, VGi+1, log, S); G ← INVERSE(G, VGi+1[1]);
73:         T ← op-1(G)|A; Insert T into R;
74:       else:
75:         G ← SOLVE(G, VGi+1, log, S); G ← INVERSE(G, VGi+1[1]);
76:         if (op = ∩): Insert G into R;
77:         else:
78:           A ← SOLVE(A, VAi+1, log, S); A ← INVERSE(A, VAi+1[1]);
79:           T ← op-1(G)|A; Insert T into R;
80:       }
81: RD ← union of all the relations in R; //Reconstructed D
82: return RD;

```

Fig. 3. (Continued.)

D in its previous value block ($V_{D_{i-1}}$). The `solve` function takes as input the name of the relation to be reconstructed D , its value block in which it is to be reconstructed V_{D_i} , a relational algebra log log , and a set S which is used to store tuples of relation and value block (and the corresponding result) which have been considered during the reconstruction. The reconstructed relation D in the specified value block is returned from the algorithm. A listing of the `solve` function is shown in figure 3.

3 Limitation of the Reconstruction Algorithm

In section 2.2, we mentioned that the output generated from the inverse operators of the relational algebra may either be complete or partial when compared with the original relation. Even though every reconstructed relation is always correct, that is, it is at least a subset of the original relation [5], partial inverses

sometimes affects the amount of information that can be reconstructed using the database reconstruction algorithm. Since the algorithm depends on the inverse operators of the relational algebra, the generation of partial inverses from some of these operators sometimes result in the generation of significantly incomplete (or empty) relation when using the algorithm.

This section gives a brief description of how the values in a relation at an earlier time can be reconstructed using the database reconstruction algorithm and reveals the limitation of the algorithm when dealing with inverse operators that generate partial inverses. In figure 1, we show a typical example of a RA log generated from a traditional SQL log file. For simplicity and further explanations in subsequent sections of the paper, we assume that the content of each relation after executing the queries in the RA log at each timestamp are as computed in figure 4.

Our aim is to reconstruct the tuples in relation H at time t_3 since several modifications of the relation has occurred. The relations R and S both have attributes **word** and **number** and contains two tuples each, which are assigned at time t_0 and t_1 , respectively.

The relation H at time t_3 in figure 1 is the same as H at any time between t_2 and t_5 inclusively, since the queries executed between these times are in the same value block of H , that is, V_{H_1} . Using the reconstruction algorithm, there are three different ways in which the relation H at t_3 can be reconstructed:

1. By reversing the query performed on the first line of value block V_{H_2} at time t_6 . Unfortunately, this cannot be achieved since the relation H was dropped (or all its contents were deleted) at this point.
2. Another alternative is to find the inverse of the intersection operation performed at time t_5 in order to obtain a partial reconstruction of H . However, since the relation J was also subsequently deleted at time t_9 , this inverse cannot be found since the inverse of the intersection operation is given as $\cap^{-1}(J) = (H^*, I^*)$ where $H^* = I^* = J$.
3. The last possible way of reconstructing H at t_3 is to re-execute the query at time t_2 . This requires that the relations R and S at time t_2 are known (or reconstructed first). Since relations R and S are in value blocks V_{R_1} and V_{S_1} , respectively at time t_2 and they both have subsequent value blocks, the relations must first be reconstructed in their respective value blocks at t_2 before the query at t_2 can be re-executed. The relation R at time t_2 (or in V_{R_1}) can be found by finding the inverse of the rename operation performed at time t_8 , which is given as $\rho^{-1}(R) = \rho_{\text{numeral} = \text{number}}(R)$. Since an inverse rename operation always generates a complete relation, the relation R at t_2 is successfully reconstructed from the inverse rename operation. The relation S at time t_2 (or in V_{S_1}) can be found by getting the inverse of the update operation performed on S at t_3 . The inverse of the update can be represented as:

$$\mathbb{F}_{\text{update}(\text{word}=\text{six})}^{-1}(\sigma_{\text{number}=5}^{-1}(S)) = \mathbb{F}_{\text{update}(\text{word}=\text{null})}(\sigma_{\text{number}=5}(S)).$$

This generates the partial relation S^* shown in table 2. Since relations R and S at t_3 are now known, the query at t_3 ($H \leftarrow R \cap S$) can be re-executed

$t_0 : R \leftarrow \{tuple_1, tuple_2\}$	<table border="1"> <thead> <tr> <th></th> <th>Word</th> <th>Number</th> </tr> </thead> <tbody> <tr> <td rowspan="2">R</td> <td>five</td> <td>5</td> </tr> <tr> <td>six</td> <td>6</td> </tr> </tbody> </table>		Word	Number	R	five	5	six	6				
	Word	Number											
R	five	5											
	six	6											
$t_1 : S \leftarrow \{tuple_1, tuple_2\}$	<table border="1"> <thead> <tr> <th></th> <th>Word</th> <th>Number</th> </tr> </thead> <tbody> <tr> <td rowspan="2">S</td> <td>four</td> <td>4</td> </tr> <tr> <td>five</td> <td>5</td> </tr> </tbody> </table>		Word	Number	S	four	4	five	5				
	Word	Number											
S	four	4											
	five	5											
$t_2 : H \leftarrow R \cap S$	<table border="1"> <thead> <tr> <th></th> <th>Word</th> <th>Number</th> </tr> </thead> <tbody> <tr> <td rowspan="1">H</td> <td>five</td> <td>5</td> </tr> </tbody> </table>		Word	Number	H	five	5						
	Word	Number											
H	five	5											
$t_3 : S \leftarrow \mathbb{F}_{update(word=seven)}(\sigma_{number=5}(S))$	<table border="1"> <thead> <tr> <th></th> <th>Word</th> <th>Number</th> </tr> </thead> <tbody> <tr> <td rowspan="2">S</td> <td>four</td> <td>4</td> </tr> <tr> <td>seven</td> <td>5</td> </tr> </tbody> </table>		Word	Number	S	four	4	seven	5				
	Word	Number											
S	four	4											
	seven	5											
$t_4 : I \leftarrow R \cup S$	<table border="1"> <thead> <tr> <th></th> <th>Word</th> <th>Number</th> </tr> </thead> <tbody> <tr> <td rowspan="4">I</td> <td>four</td> <td>4</td> </tr> <tr> <td>five</td> <td>5</td> </tr> <tr> <td>six</td> <td>6</td> </tr> <tr> <td>seven</td> <td>5</td> </tr> </tbody> </table>		Word	Number	I	four	4	five	5	six	6	seven	5
	Word	Number											
I	four	4											
	five	5											
	six	6											
	seven	5											
$t_5 : J \leftarrow H \cap I$	<table border="1"> <thead> <tr> <th></th> <th>Word</th> <th>Number</th> </tr> </thead> <tbody> <tr> <td rowspan="1">J</td> <td>five</td> <td>5</td> </tr> </tbody> </table>		Word	Number	J	five	5						
	Word	Number											
J	five	5											
$t_6 : H \leftarrow \emptyset$	<table border="1"> <thead> <tr> <th></th> <th>Word</th> <th>Number</th> </tr> </thead> <tbody> <tr> <td rowspan="1">H</td> <td></td> <td></td> </tr> </tbody> </table>		Word	Number	H								
	Word	Number											
H													
$t_7 : H \leftarrow I - J$	<table border="1"> <thead> <tr> <th></th> <th>Word</th> <th>Number</th> </tr> </thead> <tbody> <tr> <td rowspan="3">H</td> <td>four</td> <td>4</td> </tr> <tr> <td>six</td> <td>6</td> </tr> <tr> <td>seven</td> <td>5</td> </tr> </tbody> </table>		Word	Number	H	four	4	six	6	seven	5		
	Word	Number											
H	four	4											
	six	6											
	seven	5											
$t_8 : R \leftarrow \rho_{number = numeral}(R)$	<table border="1"> <thead> <tr> <th></th> <th>Word</th> <th>Numeral</th> </tr> </thead> <tbody> <tr> <td rowspan="3">R</td> <td>four</td> <td>4</td> </tr> <tr> <td>five</td> <td>5</td> </tr> <tr> <td>six</td> <td>6</td> </tr> </tbody> </table>		Word	Numeral	R	four	4	five	5	six	6		
	Word	Numeral											
R	four	4											
	five	5											
	six	6											
$t_9 : J \leftarrow \emptyset$	<table border="1"> <thead> <tr> <th></th> <th>Word</th> <th>Number</th> </tr> </thead> <tbody> <tr> <td rowspan="1">J</td> <td></td> <td></td> </tr> </tbody> </table>		Word	Number	J								
	Word	Number											
J													

Fig. 4. Original Relations Obtained from Queries Executed

Table 2. Reconstructed relation S^*

	Word	Number
S^*	four	4
	<i>null</i>	5

in order to reconstruct the tuples in H at time t_3 . However, because the reconstructed relation S^* is a partial inverse, the tuple in H at t_3 cannot be reconstructed from the re-execution of this query and an empty relation H^* with the same attributes as the original relation H is generated (table 3).

Table 3. An Empty Reconstructed relation H^*

H^*	Word	Number

This example reflects a major limitation of the database reconstruction algorithm that can be encountered when dealing with partial inverses. In the rest of this paper, we discuss some of the techniques that can be applied in conjunction with the reconstruction algorithm in order to generate more complete reconstructed relations and/or find corroborating evidence regarding the data on a database during database forensics.

4 Absence of Evidence

The database reconstruction algorithm [6] can be used to find the information in the database at an earlier time. In this same way as data collected in different branches of digital forensics can be the required evidence or assist in carrying out an investigation, reconstructed relations may often be used as the evidence², provide support for other evidence during an investigation, or to provide more information about an investigation. Unfortunately, the fact that a reconstructed relation may be incomplete implies that some evidence may not be found. In situations where the evidence to refute or support a claim cannot be found in a reconstructed relation, it is important to remember an axiom from Forensics Science that says that, “absence of evidence is not evidence of absence” [2]. For example, if no evidence (or reconstructed data) could be found to support the sales representative’s claim about the price of good sold on a particular date, it does not mean that the representative is lying. If no evidence could be found on a computer to determine whether or not it accessed a particular web page, it does not mean that the computer was used to access the site. It is important to base all assertions on solid supporting evidence and not on an absence of evidence [2]. Thus, it is necessary for an investigator to find corroborating evidence that clearly demonstrates the falsity or truth of a claim about the information on a database at an earlier time.

In this paper, we present two techniques of finding corroborating evidence about claims on the data in a database. The first technique works based on Locard’s exchange principle that contact between two items will always result

² Evidence may or may not be admissible in a court of law.

in an exchange [3]. That is, there will always be some trace evidence with every interaction even though it may not be easily detected. According to Casey [2], this principle applies in both the physical and digital realms and can provide links between them. For example, in a case involving email harassment, the act of sending messages over a web-based email service can leave traces such as files and links on the sender's hard disk and/or web browser as well as some date-time related information. Other information may also possibly be obtained from the email service provider [2]. Although this principle may not be true for all systems in general, it is true for systems that keep record of their actions or activities. In database reconstruction, the items involved in an interaction are the relations on a database while the interaction is the operation performed on such relations. This technique works on the fact that if there is a claim that some data was in a relation at an earlier time, then there should be some trace evidence that can be gathered from the interaction of the relation with other relations on the database.

The second technique for finding corroborating evidence involves the reconstruction of more complete relations through the iteration of the database reconstruction algorithm presented in [6] and inferences from reconstructed relations. The technique works on the fact that the data created when an investigator reenacts the events in a crime should resemble the original evidence collected as close as possible. That is, given a reconstructed relation, if an investigator re-executes the queries performed on the database (using the log record), the recreated database instance should be the same as the current instance of the database. If this is not the case, then it implies that some information is missing in the reconstructed relation since we have already proved that any data in a reconstructed relation is indeed correct and contained in the original relation [5].

In the following sections we describe how these techniques can be applied in finding corroborating evidence regarding claims about the information in a database at an earlier time and how the database reconstruction algorithm can be used to get more complete reconstructed relations. The techniques are currently not automated as this paper is focused on describing the logical steps to be followed during reconstruction.

5 Reconstruction from Interaction

According to Locard's exchange principle [3], the interaction or contact between two items will always result in an exchange. The technique of reconstructing data from interaction works on this principle and is synonymous to the collection of trace evidence at a crime scene.

From the reconstruction example in section 3, it is obvious that the tuples in relation H at t_3 could not be reconstructed because of two reasons:

1. the database reconstruction algorithm depends on the inverse of the update performed on relation S , which results in the generation of a partial relation S^* with missing values in one of its columns.

2. The inverse of the first query of the subsequent value block of H after time t_3 , that is, the query at t_6 in value block V_{H_2} cannot be found since H was either dropped or all of its tuples were deleted at this point.

In general, a particular situation in which the database reconstruction algorithm may be unable to reconstruct required data during an investigation is when the relation to be reconstructed is deleted in the subsequent value block of the relation; one or more relations which the relation being reconstructed interacted with have been deleted; or where the re-execution of the actual query that led to the relation being reconstructed cannot be done due to the inability to determine or reconstruct a complete version of other relations involved in the query.

An alternative way of reconstructing data in these cases is to explore the interaction of the relation to be reconstructed with other relations (using the RA log) and making inferences based on the operations performed during the interaction. A summary of inferences that can be made when considering different operations in an interaction are given below:

1. **Cartesian product:** if $H \leftarrow I(A) \times J(B)$, where A and B are attributes of the relations, then:
 - (a) $x \in \pi_A(H) \Leftrightarrow x \in I$
 - (b) $x \in \pi_B(H) \Leftrightarrow x \in J$.
2. **Union:** if $H \leftarrow I \cup J$, then:
 - (a) $x \in H \Leftrightarrow x \in I$ or $x \in J$, and this means that,
 - (b) $x \in H$ and $x \notin I \Rightarrow x \in J$ and
 - (c) $x \in H$ and $x \notin J \Rightarrow x \in I$.
3. **Intersection:** if $H \leftarrow I \cap J$, then:
 - (a) $x \in H \Leftrightarrow x \in I$ and $x \in J$.
4. **Difference:** if $H \leftarrow I - J$, then:
 - (a) $x \in H \Leftrightarrow x \in I$ and $x \notin J$
 - (b) $x \in J \Rightarrow x \notin H$.
5. **Division:** if $H \leftarrow I/J$, then
 - (a) $x \in H \times J \Rightarrow x \in I$. That is $H \times J \subseteq I$.
6. **Projection:** if $H \leftarrow \pi_A(J)$, then:
 - (a) $H \subseteq J$, that is, $y \in H \Rightarrow y \in J$ where y are values in similar columns of H and J .
7. **Selection:** if $H \leftarrow \sigma_A(J)$, then:
 - (a) $H \subseteq J$, that is, $x \in H \Rightarrow x \in J$.
8. **Rename:** if $H \leftarrow \rho_{A=B}(J)$, where A and B are attributes, then:
 - (a) $J = \rho_{B=A}(H)$ and $x \in H \Leftrightarrow x \in J$.

Considering the reconstruction example in section 3, this technique can be applied to reconstruct the tuple in H instead of the empty relation H^* generated from the reconstruction algorithm. It is important to note that the technique of reconstruction from interaction is not independent and requires the usage of the inverse operators of the relational algebra or the use of the reconstruction algorithm in regenerating other relations that might be involved in an interaction. This technique can be used in reconstructing the tuples in a relation by taking the following steps. The reconstruction of the tuples in relation H at t_3 (problem from section 3) is used to provide an example of the process at each step.

1. Identify all the interactions involving the relation to be reconstructed from the RA log. There should be at least one interaction before and after the deletion of the relation. For example, the interactions of H in figure 4 include the query at t_5 (that is, $J \leftarrow H \cap I$) and at t_7 (that is, $H \leftarrow I - J$).
2. Determine the tuples in the other relations involved in the interaction(s) that occurred after the deletion of the relation of interest either through the reconstruction algorithm or the inverse operators of the relational algebra. For example, we need to find the inverse of the query $H \leftarrow I - J$ in order to determine the tuples in J since relation I is known. Thus, we have³:

$$-^{-1}(H) = J^* = I - H$$

which is as shown in table 4.

Table 4. Relation J^* from the inverse difference operation

J^*	Word	Number
	five	5

3. The last step involves making inferences from the other relations that have been reconstructed in step 2, and which were also involved in an interactions with the relation being reconstructed before its deletion. For example, the relation J was involved in an interaction with H at t_5 (that is the query, $J \leftarrow H \cap I$) and since this involves an intersection operation, the inferences described earlier implies that every tuple in J must also be in H . That is,

Table 5. H through Reconstruction from Interaction

H	Word	Number
	five	5

we have the relation H which is given as table 5 instead of the earlier empty relation in table 3.

6 Reconstruction through Iteration

Another technique that can be used in reconstructing the information in a database is through the iteration of the database reconstruction algorithm and the queries in the RA log, and making inferences from tuples generated and queries performed during the process.

³ Although the resulting J^* is complete when compared with the original J at t_7 , this is not always the case with inverse difference operator.

	Word	Number
R_r	five	5
	six	6

	Word	Number
S_r	four	4
	<i>null</i>	5

Fig. 5. Reconstructed relations R_r and S_r

The technique works on the notion that if the queries in a log are re-executed using some reconstructed relations, then the final instance of the database generated after the re-executions should be the same as the current instance of the database. Since it was proven in an earlier work [5] that the output generated from the database reconstruction algorithm is correct, any difference between the current instance of the database and the instance generated from the re-executions implies that there are some missing data in one or more relations involved in the queries that were re-executed. The differences identified between the two database instances can be used to make inferences and reconstruct the missing data in the relations involved.

Considering the reconstruction example in section 3, this technique can be applied to reconstruct the tuple in H instead of the empty relation H^* generated from the reconstruction algorithm. The steps involved in this technique are listed below. The reconstruction of the tuples in relation H at t_3 (problem from section 3) is used to provide an example of the process at each step.

1. Attempt the reconstruction using the database reconstruction algorithm and identify other relations that needed to be reconstructed. For example, our attempt to reconstruct relation H at t_3 in figure 4 required the reconstruction of relations R and S . For simplicity, we will use a subscript r to denote relations that were reconstructed or generated from reconstructed relation. Thus, the reconstruction of relations R and S generated the relations $R_r = R$ and $S_r = S^*$ (as explained in section 3) given in figure 5.
2. Re-execute the queries in the log using the reconstructed relations and make possible inferences whenever a reconstructed relations differs from the current instance on the database. For examples, in the reconstruction of H , we can re-execute the queries in figure 4 using the relations R_r and S_r . The re-execution process is shown in figure 6.

At time t_4 of the re-execution, the relation I_r generated differs from the relation I in the current instance of the database. A comparison of the two relations (figure 7) shows that I contains a tuple that is not in I_r and I_r contains a tuple that is not in I . It is possible to assume that the *null* value in I_r is indeed the value “seven” since there is only one column with a missing value and the second column in both I and I_r matches. Alternatively, we can make inferences from the tuple in I which is not in I_r . That is, since I is a union of R and S , then the tuple $\langle \text{seven}, 5 \rangle$ should be in either R_r or S_r . However, since we are sure that the relation R_r is complete, it implies that the tuple is in S_r . That is, S_r is given as table 6. Since the relation S_r generated from the inferences are exactly the same as the current

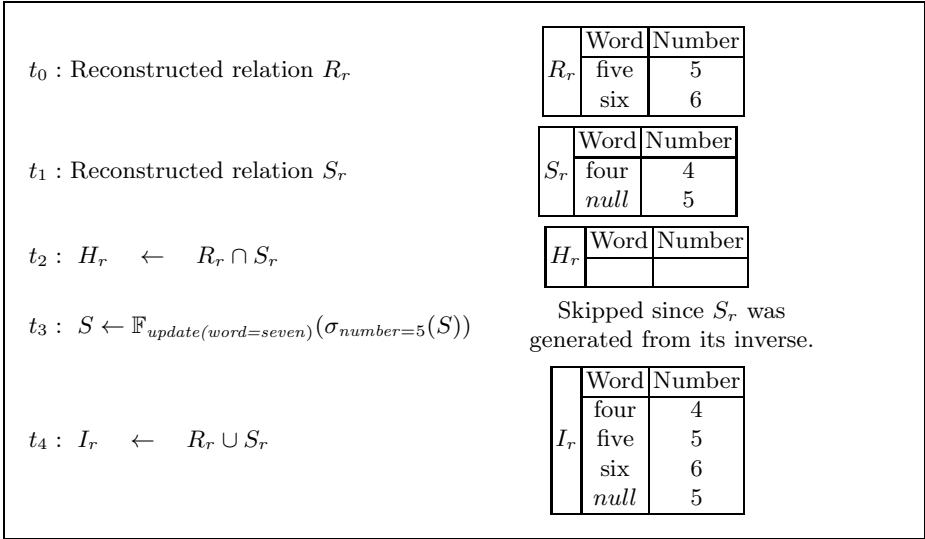


Fig. 6. Re-execution of the query log using the reconstructed relations

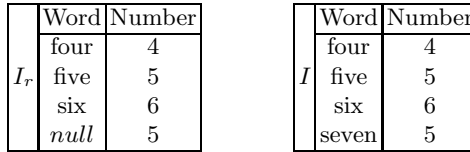


Fig. 7. Reconstructed relation I_r and current relation I

Table 6. Table S_r generated from re-execution and inferences

Word	Number
four	4
seven	5

instance of the relation S , no further inferences can be made at this point. The concluding part of the re-execution process is shown in figure 8. The relation H_r generated from the re-execution process at time t_7 should be the same as the current instance of H on the database. But, this is not the case (as shown in figure 9). Again, the differences between the two relations can be used to make inferences about the data in the database. Relation H_r contains the tuple $\langle \text{five}, 5 \rangle$ which is not present in the current instance of H , this implies that some data was missing in the reconstructed relations used to compute H_r . Since the tuple, $\langle \text{five}, 5 \rangle$ is not expected to be in H_r , then the only possibility is that it should have been in the relation J_r since

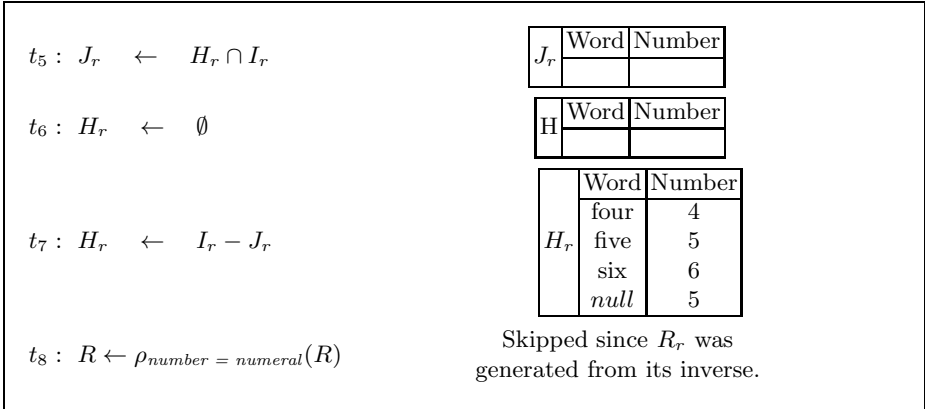


Fig. 8. Re-execution of the query log using the reconstructed relations

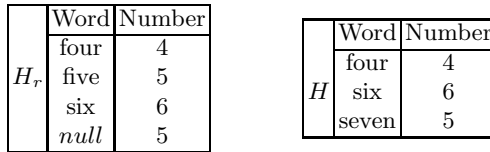


Fig. 9. Reconstructed relation H_r and current relation H

all the tuples in J_r were removed from I_r to generate H_r (from the difference operation at t_7). If the tuple is in J_r at t_7 , it implies that it was also in J_r at t_5 since both times are in the same value block of J . This further implies the tuple $\langle \text{five}, 5 \rangle$ was in both H_r and I_r at time t_5 . Also, since t_5 and t_2 are in the same value block of H , it implies that the tuple $\langle \text{five}, 5 \rangle$ was in H at t_2 and subsequently at t_3 . Thus, the relation H at t_3 can be reconstructed as shown in table 7.

Table 7. H from Reconstruction through Iteration

H	Word	Number
	five	5

As with the technique of reconstruction from interaction and as shown in the example above, the technique of reconstruction data through iteration also rely on the use of the database reconstruction algorithm, inverse relational algebra and value blocks. Also, the techniques are currently not automated as this paper is focused on describing the logical steps to be followed during reconstruction. Both techniques can be used in reconstructing data when dealing situations

involving incomplete reconstruction of some other relations or the deletion of required relation at some point in the log file. The decision about which of the techniques to use will depend on the content of the log file and/or an intuitive decision of which technique is likely to enable the reconstruction of more data.

7 Conclusion and Future Work

This paper discusses an algorithm for reconstructing the information in a database at an earlier time and presents the limitation of the algorithm using a typical example. The limitation of the algorithm arises mainly because of the possibility of generating incomplete inverses when using the inverse operators of the relational algebra. Since the reconstructed relation or tuples of a relation may often be used as evidence in an investigation; to refute or support claims about the content of a database at an earlier time; or to simply get for information about an investigation, the reconstruction of incomplete data may imply that some evidence are missing.

The paper describes two different techniques that can be used in conjunction with the database reconstruction algorithm and the inverse operators of the relational algebra to generate more complete relations or provide corroborating evidence for claims about the data on a database at an earlier time. The first technique works based on Locard's exchange principle while the other rely on the iteration of the reconstruction algorithm and re-execution of the queries in the log file. Both techniques are described using a typical example.

Future work will entail investigating if these techniques can be used to reconstruct all the tuples in a relation always and if not, describe the conditions under which complete relations can be reconstructed. In addition, we will determine whether the information recovered from a database using the reconstruction algorithm and these techniques is "maximal" in that one determines that the log contains no further information that may be used to reconstruct values.

Acknowledgement. This research was supported by the Organization for Women in Science for the Developing World (OWSD).

References

1. Carrier, B.: Defining digital forensic examination and analysis tools using abstraction layers. *International Journal of Digital Evidence* 1, 2003 (2002)
2. Casey, E.: *Digital Evidence and Computer Crime - Forensic Science, Computers and the Internet*, 3rd edn. Academic Press (2011)
3. Chisum, W.J., Turvey, B.: Evidence dynamics: Locard's exchange principle & crime reconstruction. *Journal of Behavioural Profiling* 1(1) (January 2000)
4. Codd, E.F.: *The Relational Model for Database Management, Version 2*. Addison-Wesley (1990)
5. Fasan, O.M., Olivier, M.S.: Correctness proof for database reconstruction algorithm. *Digital Investigations* (2012)

6. Fasan, O.M., Olivier, M.S.: Reconstruction in database forensics. In: Peterson, G., Sheno, S. (eds.) *Advances in Digital Forensics VIII*. IFIP AICT, vol. 383, pp. 273–287. Springer, Heidelberg (2012)
7. Fowler, K.: *SQL Server Forensic Analysis*. Addison Wesley Professional (2008)
8. Garfinkel, S.L.: Digital forensics research: The next 10 years. *Digital Investigation* 7, S64 – S73 (2010); *The Proceedings of the Tenth Annual DFRWS Conference*
9. Litchfield, D.: Oracle forensics part 1: Dissecting the redo logs. NGSSoftware Insight Security Research (NISR) Publication (March 2007)
10. Litchfield, D.: Oracle forensics part 2: Locating dropped objects. NGSSoftware Insight Security Research (NISR) Publication (March 2007)
11. Litchfield, D.: Oracle forensics part 3: Isolating evidence of attacks against the authentication mechanism. NGSSoftware Insight Security Research (NISR) Publication (March 2007)
12. Litchfield, D.: Oracle forensics part 4: Live response. NGSSoftware Insight Security Research (NISR) Publication (April 2007)
13. Litchfield, D.: Oracle forensics part 5: Finding evidence of data theft in the absence of auditing. NGSSoftware Insight Security Research (NISR) Publication (August 2007)
14. Litchfield, D.: Oracle forensics part 6: Examining undo segments, flashback and the oracle recycle bin. NGSSoftware Insight Security Research (NISR) Publication (August 2007)
15. Olivier, M.S.: On metadata context in database forensics. *Digital Investigation* 5(3-4), 115–123 (2009)
16. Palmer, G.: A road map for digital forensic research. Technical report. In: *First Digital Forensic Research Workshop (DFRWS)*, Utica, New York (August 2001)
17. Wright, P.M.: Oracle database forensics using logminer. *Next Generation Security Software* (January 2005)
18. Wright, P.M., Burleson, D.K.: *Oracle Forensics: Oracle Security Best Practices*. Rampant Techpress (2010)

BlackBerry PlayBook Backup Forensic Analysis

Mohamed Al Marzougy¹, Ibrahim Baggili², and Andrew Marrington¹

¹ Advanced Cyber Forensics Research Laboratory, College of Technological Innovation,
Zayed University, Abu Dhabi, U.A.E.

Mohamed.Almarzougy@gmail.com, Andrew.Marrington@zu.ac.ae

² Tagliatela College of Engineering, Department of Electrical and Computer Engineering
and Computer Science, University of New Haven, CT
Ibaggili@newhaven.edu

Abstract. Due to the numerous complicating factors in the field of small scale digital device forensics, physical acquisition of the storage of such devices is often not possible (at least not without destroying the device). As an alternative, forensic examiners often gather digital evidence from small scale digital devices through logical acquisition. This paper focuses on analyzing the backup file generated for the BlackBerry PlayBook device, using the BlackBerry Desktop Management software to perform the logical acquisition. Our work involved analyzing the generated “.bbb” file looking for traces and artifacts of user activity on the device. Our results identified key files that can assist in creating a profile of the device’s usage. Information about BlackBerry smart phone devices connected to the tablet was also recovered.

Keywords: BlackBerry, Forensics, PlayBook, Backup.

1 Introduction

The BlackBerry PlayBook is Research in Motion’s (RIM) entrant into the heated tablet race which includes the iPad and various Android tablets. One of the main differences between the PlayBook device and other tablets is the ability to tether (via Bluetooth) to a BlackBerry smart phone for network access while away from WiFi networks at home or in the office, as compared to using an on-board 3G modem for that purpose. This tethering is provided by the BlackBerry Bridge feature that extends the functionality of the paired BlackBerry smart phone to the PlayBook’s larger screen, enabling the viewing of emails, messages and files stored on the phone.

Although the iPad and the various Android tablets run a tablet-version of an operating system designed for a smart phone, the BlackBerry PlayBook runs a custom operating system. This means that research into the forensic acquisition of BlackBerry smart phones may not be applicable to the PlayBook device. To date, there has been no research performed on the forensic acquisition and analysis of the PlayBook’s backup structure. Although the PlayBook has a comparatively small market-share [1], the PlayBook was the first tablet to gain FIPS 140-2 certification and cleared to be used by the U.S. Government [2]. Therefore, it is a worthwhile exercise to study the forensic acquisition, analysis and examination of the device via its backup structure.

This approach has recently been applied successfully to the iPad [3] and we therefore thought to investigate its applicability to the BlackBerry PlayBook.

The remainder of this paper is organized as follows: in section 2 we briefly discuss the literature about the forensic examination of various types of tablet computers. In section 3, we describe the methodology for our experiment and we discuss our findings in section 4. In section 5 we draw conclusions from our work and we finish by discussing future research work into this area of small scale digital device forensics.

2 Background

Mobile phones and tablets are of particular interest to forensic investigations for the simple reason that due to their mobility they are likely to be in regular contact with suspects and/or victims throughout the course of the events under investigation. With enormous diversity in operating system software, hardware specifications, and vendors, small-scale digital devices like smart phones and tablets are an area of serious concern in digital forensic research [4].

Small-scale digital device forensics is a rapidly evolving subfield of digital forensics. The initial popularity of the iPhone and subsequently the iPad led to research into the retrieval and analysis of digital evidence from these devices [5][6][7]. There has been some research into Android devices [8], although it has been almost exclusively focused on phones and much remains to be done before a generalized methodology for Android forensics is possible [9]. There has also been some research on BlackBerry smart phone devices [10], but at the time of writing there is little published research about the BlackBerry PlayBook tablet, which is the focus of this paper.

2.1 iPhone and iPad

The iPhone, iPod Touch, and iPad all run the iOS operating system, and may be conceived of as broadly similar devices from a forensics perspective. All iOS devices interface with a personal computer or accessory peripherals through a proprietary port on the bottom of the device which connects to the computer's Universal Serial Bus (USB) port via a special cable. None of the iOS devices feature removable storage and consequently, any digital forensic examination of the device must take place via this cable.

Physical acquisition for iOS devices is limited to commercial products and law enforcement personnel. Andrew Hoog and Katie Strzempka [11] reviewed most tools that support iOS device forensics using the criteria: installation, acquisition, reporting and accuracy, where they came up with a ranking system they used to rank 13 digital forensics products and methodologies. The Zdziarski method scored the highest (4.1) where the rest averaged 3.3. Zdziarski's iPhone forensics method is one of the few which does not require the target device to be jailbroken - all an examiner has to do is put the device into recovery mode and load Zdziarski's tool into the device's RAM. The technique is conceptually similar to using a boot CD - essentially the device boots to an "alternate" system partition that has all the necessary software to run a

“dd” command and create a forensic image of the user partition, bypassing any password protection. The National Institute of Standards and Technology validated the Zdziarski method as forensically sound [12].

Gómez-Miralles and Arnedo-Moreno employ jailbreaking in their approach, which uses the Apple Camera Connection Kit for the iPad to connect the device to an external hard drive [13]. After the iPad is jailbroken, OpenSSH and core utilities (coreutils) are installed on it, and the investigator connects to the device from a computer on the same WiFi network as the iPad using ssh. The “dd” command is issued to the target device specifying that the output is to be stored on the external hard drive connected via the camera connection kit.

2.2 Android Devices

Similar to the iPhone, Android keeps all the system files and some of the user information protected on the kernel level. Consequently, many forensic scientists suggest that the device should be “rooted” (a similar process to jailbreaking) to facilitate examination [9]. The Android file system is “Yet Another Flash File System 2” (YAFFS2). YAFFS, developed in 2002, was the first file system designed for NAND (Not-AND) flash memory devices. YAFFS2 was designed in 2004 in response to the availability of larger sized NAND flash devices; older chips support a 512 byte page size whereas newer NAND memory has 2096 byte pages. YAFFS2 is backward compatible with YAFFS [8].

The first and most obvious step is to perform a traditional forensics analysis of the microSD card from the Android device. This step will obviously only result in the acquisition of whatever data has been stored to the SD card but not the data which is stored in the device’s non-removable memory. Android device SD cards use the FAT32 file system and are easily imaged and examined using traditional forensics tools (including write-blocking hardware).

In order to acquire access to the Android device’s internal memory as opposed to simply the SD card, USB debugging must be enabled on the device. This mode can be enabled by the user through the appropriate configuration menu on the Android device. If the Android device’s keylock is active, then the investigator requires the user’s passcode to gain access to the configuration menu. According to Lessard and Kessler [8], unless USB debugging has been enabled, it is not possible to root the Android device. Golubev [14] explains that in the absence of the passcode, root access is necessary to bypass the Android device’s keylock. This creates a “chicken and the egg” scenario where if the keylock passcode is unknown, the investigator must disable the keylock remotely via root access, but if the investigator cannot disable the keylock, he/she will be unable to root the Android device.

The exact process of rooting an Android device varies depending on the hardware manufacturer follows the same general process. This process requires inserting an SD card (preferably fresh and not the one used by the device as it may store evidence on it) and enabling USB debugging mode, then, through the use of Android Development Tools (ADT, part of the Android SDK) and the Android Development Bridge (ADB), a small program is copied to the SD card. This program is usually

copied to /data/local/tmp, a folder where most installation files reside. The program is then run in order to root the device.

Other research has focused upon the analysis of the live memory of Android devices. Researchers have developed a tool that performs a dump of each running process' memory [15]. Although excellent for the analysis of a single process (such as a single running application), many other potentially interesting parts of the Android device's memory are not analyzed including in-kernel structures, networking information, etc. Another issue is that this approach requires memory to be extracted separately for each process of interest, which requires a number of interactions with the live system, increasing the chance that evidence will be contaminated. Sylve et al developed a kernel module that can be loaded to a rooted Android device to dump the memory of the device to the device's SD card with very high accuracy [16].

2.3 BlackBerry Devices

BlackBerry devices have long had the reputation for security, both with respect to the data stored on the device and to the security of emails and messages sent to and from the device. Previous work studying BlackBerry smart phone devices found that data was only forensically recoverable on devices where the users had not employed the device's encryption features [10]. However, the BlackBerry PlayBook uses a different operating system entirely from the BlackBerry OS used on the generations of BlackBerry smart phones up to this point. The BlackBerry Tablet OS is based on QNX Neutrino, an OS that is employed to run on many other portable devices. This operating system is Unix-based and features a microkernel.

BlackBerry devices were among the first smart devices to hit the market and as a result they became popular among government officials and corporate customers alike. Most BlackBerry devices come with the option to completely encrypt its memory. Further, the device makes it possible to encrypt the device's Secure Digital (SD) card as well. It is also possible to wipe a BlackBerry device remotely in the event that the device has been lost or stolen. BlackBerry devices, both the BlackBerry smart phones and the BlackBerry PlayBook, can also be backed up to a desktop computer using the BlackBerry Desktop Manager software. These backups may contain much information of forensic value to an investigator, just as they do for the iPhone [5] and iPad [3].

3 Methodology

Our method can be summarized as using a BlackBerry PlayBook device under manual observation, involving recording of all actions and their outcomes, before backing the device up with BlackBerry Desktop Manager and then analyzing the backup files produced to determine those of most potential interest to an investigator and their structure.

3.1 Test Equipment

Hardware:

- 64 GB BlackBerry PlayBook running OS 2.0.7971
- BlackBerry Bold 9900 running OS 7.1 Bundle 921 (7.1.0.267, Platform 5.1.0.230)
- IBM ThinkVantage with 2.6 Ghz Quad Core Intel processor, 4 GB RAM running Windows XP Professional, Service Pack 3.

Software and tools:

- BlackBerry Desktop Software 6.1.0.35
- Facebook for BB PlayBook 2.2.1.7
- WinRAR 3.30
- Hex Workshop 6.6
- SQLite Browser 2.0b1
- AccessData FTK 3.2
- Snagit

3.2 Test Procedure

The BlackBerry PlayBook device was initiated and connected to a wireless network as part of the initiation process. The device was connected to the lab's wireless network and the timezone was selected. After that the device required a BlackBerry ID, which was created using the following details:

- BlackBerry ID: bbbpmail@gmail.com
- First name: ZUPlayBook
- Last name: Student
- BlackBerry ID username: bbbpmail@gmail.com
- Password: zuBlackBerry
- Recovery Question: Where?
- Recovery Answer: Here
- Screen name: ZU

After the successful BlackBerry ID registration, the device was forced to update to OS 2.0.7971 and went through the first launch tutorials and demo. After that, the device was connected to the BlackBerry 9900 smart phone through the BlackBerry Bridge connection (over Bluetooth). Accessing the BlackBerry Bridge applications required the smart phone's password. The PlayBook then accessed emails from the smart phone through the bridge to the first author's email address, and we sent and received test emails to and from the account bbbpmail@gmail.com. The next bridge app we used was the BlackBerry Messenger (BBM), specifically checking received messages and then sending and receiving some BBM messages to members of the smart phone's contact list. We then disconnected the PlayBook from BlackBerry Bridge.

The next step was using a new feature in OS 2.0: direct email setup. Using this feature, the PlayBook device is used to directly receive and send emails over WiFi without the need for a tethered smart phone device connected via BlackBerry Bridge.

Subsequent to that, we performed some browsing activities on the PlayBook device, using the default browser, and then we started to run the YouTube and FaceBook applications. We also used the camera to take two photos and one video. Finally, a hotspot was created using the BlackBerry smart phone, and the PlayBook device was connected to that hotspot.

After that the device was connected to the PC to capture a backup. From the Desktop Software the backup option was set to “Full (all device data and settings)”.

After the backup was taken, more operations were made for comparison. One of the image files was deleted, a website was deleted from the browsing history and more images were copied to the device using the Desktop Manager Software. Files named *dizer.jpg* and *low.jpg* and *chub.jpg* were copied using the file explorer of the Desktop Manager Software. Then from the device the file *chub.jpg* was deleted. The device was then backed up again.

WinRAR was used to extract the files from the .bbb files, which are ZIP files with the “bbb” extension. After extracting everything into 2 folders, “before delete” and “after delete”, the folders were added to AccessData FTK as live evidence.

4 Analysis and Findings

After extraction, both files had the same structure. The backed up files were divided into 3 main tar files: *App.tar*, *Setting.tar* and *Media.tar*. Along with these tarballs was an xml file describing the content of the files called *Manifest.xml*. It showed the device PIN and OS version as well as file size for the above mentioned tarballs as shown in Figure 1.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <BlackBerry_Backup>
  <Client platform="windows" osversion="Microsoft Windows NT 6.1.7601 Service Pack 1" dtmversion="6.1.0.35" />
  <Version>2.0</Version>
  <Encryption type="RIM_AES_CBC" version="1.0" Salt="" />
- <SourceDevice pin="50109FDE" hwid="6001A06">
  <Platform type="QNX" version="2.0.0.7971" />
</SourceDevice>
- <QnxOSDevice>
  - <Archives>
    <Archive id="app" name="Application Data" count="71" bytesize="21430784" />
    <Archive id="media" name="Media" count="20" bytesize="8634368" />
    <Archive id="settings" name="Settings" count="820" bytesize="1114624" />
  </Archives>
</QnxOSDevice>
</BlackBerry_Backup>
```

Fig. 1. Content of Manifest.xml

4.1 Media.tar

Examining tar files using WinRAR, we first started out with the *Media.tar* file which had two folders in it, *Media* and *dtm*. The “Media” folder has the same structure of folders when you connect your device to the PC as shown in Figure 2.

We found all the images as well as the video taken by the camera in the folder *Camera* in the first .bbb file. Additionally, all the uploaded images were saved in the *\photos\Pictures\BlackBerry* folder. There were no traces of the deleted image taken by the camera, but moving into the *dtm* folder of the “after delete” .bbb file we found the file *c2f39ce10000004.bbms* which listed the file name of all images uploaded into the device including the one we deleted.

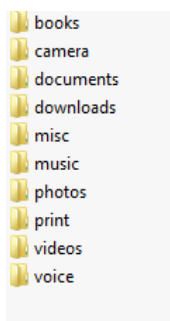


Fig. 2. Media folder content

4.1 Settings.tar

Settings.tar is an archived folder containing several files. Notably, *Settings.tar* contained another file called *dynamic.lm* in the directory *\accounts\1000\sys\input\fluency\user*. This file contained the emails sent from the device.

The directory *\apps\services* listed all the services in the device such as: accelerometer, audio, clock, geolocation, input, light_sensor and more. Table 1 summarises the files found with evidence in them:

Table 1. Evidence Files in services folder

File name	Path	Description
Status	<i>\apps\services\accelerometer</i>	The file shows the orientation of the device at the time of backup, and whether it was facing up or down.
Status	<i>\apps\services\audio</i>	The file shows the audio status and whether a2dp bluetooth audio is enabled or not.

Table 1. (Continued.)

File name	Path	Description
Status	\pps\services\clock	The file showed which time zone the device was using.
Status	\pps\services\geolocation \country	The file showed the country code for the country the device was in.
Status	\pps\services\network-time	Showed the time stamp of the clock update and the ntp server used.
orientation	\pps\services\sensor	Same information provided by the accelerometer status file
Settings	\pps\system	The file contained: <ul style="list-style-type: none"> • Time format • Language used • Time Zone

Notably we found two sub-folders in the folder `\settings\var` which appear worthy of further investigation; `certmgr` and `keymgr`. The first one seemed to contain all the certificates the device uses for communication and the other one contained a set of private keys.

Digging further in the folder we found the file `wpa_pps.conf` in the directory `\var\etc\netsecure` which stored all the info related to the wireless networks to which the device had been connected. Another interesting finding was that the device also copied all the networks to which the BlackBerry smart phone had ever connected, including all the SSIDs and passwords, in clear text. This included wireless networks to which the BlackBerry smart phone had connected before it had connected to the PlayBook device using BlackBerry Bridge.

4.2 Apps.tar

This file contained obscured folder names, similar to what Apple does with iOS application folders with obfuscated names. We speculate that the names may be generated through a hash function of some description. Within Apps.tar, we found a file named `core.all` in the directory `sys.navigator/appdata\data`. This file can be thought of as a map for the obfuscated folders within the tarball. Furthermore, in the same folder were other files that were a subset of `core.all`, like `userapps` (shown in Figure 3) which lists only third party apps installed, and `core.corporate` that lists all the OS built-in apps, while the file dock showed the apps “pinned” to the dock in the PlayBook’s GUI.

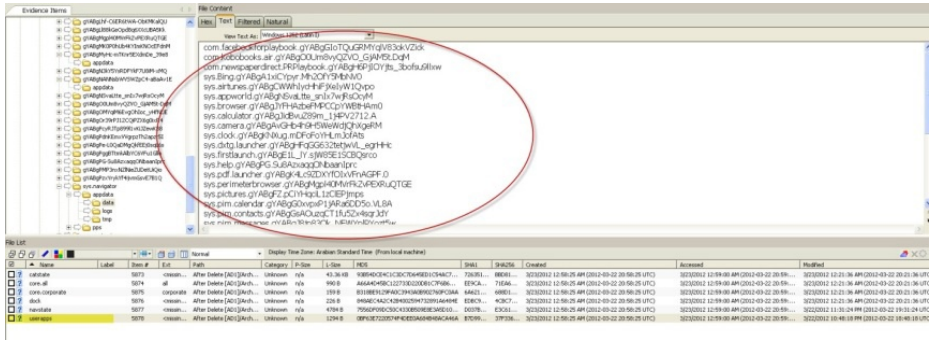


Fig. 3. Userapps lists all third party apps installed on the BlackBerry PlayBook device

Using the abovementioned files we focused on the folders of apps that may have potential evidence in them. We started by examining the browser’s folder as we expected it to be the richest in terms of recoverable data. The browser’s folder was named *gYABgJYFHAzbeFMPCpYWBtHAM0* and it contained the files shown in Table 2.

Table 2. Evidence from browser app

File name	Path	Description
settings.sol	\#SharedObjects \browser.swf	This file showed the settings used by the browser: history expiry, homepage, default search engine, encoding used, font size and user agent string can be found.
Cache(folder)	\appdata\data\cache	This folder contained cached web files which can be used to reconstruct browsing history and browsed pages.
WebpageIcons.db	\appdata\data\database	An SQLite database that contains information about visted sites’ fav icon and where to get them. Can be used to track browsing histroy.
Favicon(folder)	\appdata\data\favicon	Fav icon are cashed in this folder in png formats.

Table 2. (Continued.)

File name	Path	Description
Snapshot(folder)	\appdata\data\snapshot	This folder holds the snapshots of the visited websites as seen on the browser history section. It stores them in 2 sizes in landscape and portrait.
browser-v1.0.db	\appdata\data	A SQLite database that has 2 tables, Bookmark and history, which are self explanatory.
cookieCollection.db	\appdata\data	Another SQLite database that stores all the cookies that are stored on the device with information like: host, expiry and last accessed

Further key term searching led us to the YouTube folder *gYABgPcyRJTp8991IvKiJZewK88*, to the file *qnx.youtube.sol* located in *\appdata\data\#SharedObjects\Youtube.swf*. Here we also found additional information about the clip we watched, including the URL, the URL of the comments and some related videos. The folder *appdata\data\appdata* had a SQLite database, *cookies.sqlite*, that held the cookies used for the YouTube application and others.

4.7 Limitations

The analysis was conducted on only one PlayBook device, so we couldn't record how hardware changes might affect the results, if indeed they would affect them. The experiment was performed on the original PlayBook device, not the new 4G LTE PlayBook device, which is capable of connecting to the mobile network independent of the BlackBerry Bridge tethering feature. Likewise; our examination of artifacts left as a result of the BlackBerry Bridge tethering feature only involved one additional device, a BlackBerry Bold 9900. Other BlackBerry smart phone devices were not used in this experiment, although BlackBerry Bridge is supported on a wide range of BlackBerry smart phone models. Most significantly, the technique described in this paper depends on logical acquisition through the BlackBerry backup procedure, and therefore shares the limitations common to logical acquisitions of all digital devices. Our plans to address these limitations are discussed in section 5, below.

5 Conclusions and Future Work

The original (non-4G LTE) BlackBerry PlayBook device is a low-priced tablet which integrates with the BlackBerry smart phone device. Despite an overall smaller share of the tablet and smart phone markets than iOS and Android-based competitors, the BlackBerry devices (PlayBook and smart phones alike) remain popular and widely deployed in the corporate and government markets, and in the mainstream consumer market in many countries. This paper described a logical acquisition-based approach to investigating the BlackBerry PlayBook device. Our approach is based on the use of the BlackBerry PlayBook backup file created by the normal backup procedure through the BlackBerry Desktop Management software. We examined the backup data structure and identified files stored within which appeared to be of forensic interest. Table 3 lists the files within this backup structure which we identified as likely containing information of interest to a digital investigation.

Table 3. Summary of results

Tarball	File Path within Tarball	Description
Settings.tar	\accounts\1000\sys\input\fluency\user \dynamic.lm	Emails that are sent from the device.
Settings.tar	\pps\services\accelerometerb\Status	Orientation of the device at the time of the backup
Settings.tar	\pps\services\audio\Status	Bluetooth and A2DP usage
Settings.tar	\pps\services\clock\Status	Current Time Zone
Settings.tar	\pps\services\geolocation\country \Status	Country code for geo location at the time of the backup
Settings.tar	\pps\services\network-time\Status	NTP server used and time stamp for last update (Unix EPOCH time)
Settings.tar	\pps\services\sensor\orientation	Orientation of the device at the time of the backup
Settings.tar	\pps\system\Settings	Language, time format and time zone
Settings.tar	\settings\var 2\certmgr	x.509 certificates
Settings.tar	\settings\var 2\keymgr	Private keys
Settings.tar	\var\etc\netsecure\wpa_pps.conf	Information about wireless networks, including passwords

Table 3. (Continued.)

Tarball	File Path within Tarball	Description
Media.tar	\media\camera	All the images and videos taken by the camera, none of the deleted
Media.tar	\dtm\MediaSync \c2f39ce100000004.bbms	List of all images synced to the device, even the deleted ones.
Apps.tar	\sys.navigator\appdata\data\core.all	Map to all application folders
Apps.tar	\sys.navigator\appdata\data \core.corporate	Subset that shows only OS built in applications
Apps.tar	\sys.navigator\appdata\data\userapps	Subset that shows user installed applications
Apps.tar	\gYABgJYFHAzbeFMPCCpYWBtHAm0\	Browser application folder, can be different
Apps.tar	\SharedObjects\browser.swf \settings.sol	Browser settings
Apps.tar	\appdata\data\cache\	Browser Cache
Apps.tar	\appdata\data\database \WebpageIcons.db	Fav icon information
Apps.tar	\appdata\data\favicon\	Fav icon image files
Apps.tar	\appdata\data\snapshot	Websites snapshots
Apps.tar	\appdata\data\ browser-v1.0.db	Bookmarks and history tables
Apps.tar	\appdata\data\cookieCollection.db	Browser cookies
Apps.tar	\gYABgE1L_JY-sjW85E1SCBQsrco \firstlaunch.sol	Device name and BlackBerry ID used to initiate the device
Apps.tar	\gYABgPcyRJTp89911vKiJZewK88	YouTube Application folder
Apps.tar	\appdata\data\#SharedObjects \Youtube.swf	YouTube searches and videos watched.
Apps.tar	\appdata\data\appdata\cookies.sqlite	Cookies stored by YouTube Application.

In the future, we plan to run more extensive tests on a broader range of BlackBerry hardware and software. We also plan on creating stronger usage scenarios to create more complete user profiles. As we continue to investigate these devices, it will be possible to develop a software parser for the PlayBook backup structure which can be used to automate the discovery of the different items of interest we discovered in the

investigation described in this paper. This parser could be combined with a convenient user interface to display or export this information, to assist forensic investigators.

The new 4G LTE BlackBerry PlayBook is substantially similar to the original BlackBerry PlayBook except for the addition of a 4G LTE modem. Although this new model still supports BlackBerry Bridge, the 4G LTE modem allows connection to the mobile network directly through a micro-SIM. We anticipate that the back-up structure would be extremely similar to the structure described in this paper, but have yet to confirm this through our own testing. One point of interest is that the new 4G LTE modem may lead to a significant reduction in the use of the BlackBerry Bridge feature which, as we have shown, leaves interesting evidence about paired BlackBerry smart phones used with the subject PlayBook.

Another limitation of the work described in this paper, as noted above, is our dependence on the BlackBerry PlayBook backup file. The backup file may be thought of as a logically acquired image of the PlayBook device, and as with all logical acquisitions, there may be some additional evidence stored on the device itself which cannot be retrieved. For example, deleted files or data stored in primary memory only as opposed to secondary storage on the device (e.g. cryptographic keys, passphrases) may be of forensic interest, but will not be retrieved through a logical acquisition of the PlayBook device's secondary storage. We plan to address this deficiency by investigating techniques for physical acquisition of the PlayBook device. DingleBerry [17] is a PlayBook hacking tool which permits root access to the BlackBerry PlayBook device. DingleBerry may provide a mechanism for our future work in the physical acquisition of the BlackBerry PlayBook device.

References

1. Gartner Research. Gartner Says Worldwide Media Tablets Sales to Reach 119 Million Units in 2012 (2012), <http://www.gartner.com/it/page.jsp?id=1980115> (retrieved)
2. BlackBerry PlayBook cleared for government use, <http://www.cbc.ca/news/technology/story/2011/07/22/technology-BlackBerry-PlayBook-rim.html> (retrieved)
3. Ali, S., AlHosani, S., AlZarooni, F., Baggili, I.: iPad2 logical acquisition: Automated or manual examination? In: Proceedings of the 2012 ADFSL Conference on Digital Forensics, Security and Law, Richmond, VA (2012)
4. Garfinkel, S.L.: Digital forensics research: The next 10 years. In: Proceedings of the 2010 Digital Forensics Workshop published in Digital Investigation, vol. 7, pp. S64-S73 (2010), doi:10.1016/j.diin.2010.05.009
5. Bader, M., Baggili, I.: iPhone 3GS Forensics: Logical analysis using Apple iTunes Backup Utility. Small Scale Digital Device Forensics Journal 4(1) (2010)
6. Gómez-Miralles, L., Arnedo-Moreno, J.: Universal, Fast Method for iPad Forensics Imaging via USB Adapter. In: Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), Valencia, pp. 200–207 (2011)

7. Iqbal, B., Iqbal, A., Al Obaidli, H.: A Novel Method of iDevice (iPhone, iPad, iPod) Forensics without Jailbreaking. In: 2012 International Conference on Innovations in Information Technology (IIT), Al Ain (2012)
8. Lessard, J., Kessler, G.C.: Android Forensics: Simplifying Cell Phone Examinations. *Small Scale Digital Device Forensics Journal* 4(1) (2010)
9. Vidas, T., Zhang, C., Christin, N.: Toward a general collection methodology for Android devices. In: Proceedings of the 2011 Digital Forensics Workshop published in *Digital Investigation*, vol. 8, pp. S14-S24 (2011)
10. Valli, C., Jones, A.: A Study into the Forensic Recoverability of Data from 2nd Hand BlackBerry Devices: World-Class Security, Foiled by Humans. In: Proceedings of World Congress in Computer Science, Computer Engineering and Applied Computing, Las Vegas, pp. 604–607 (2008)
11. Hoog, A., Strzempka, K.: Independent Research and Reviews of iPhone Forensic Tools (2010), <https://viaforensics.com/resources/white-papers/iphone-forensics/> (retrieved)
12. National Institute of Standards and Technology, Test Results for Mobile Device Acquisition Tool: Zdziarski's Method (2010), <http://www.nij.gov/pubs-sum/232383.htm>
13. Gómez-Miralles, L., Arnedo-Moreno, J.: Versatile iPad forensic acquisition using the Apple Connection Kit. *Computers & Mathematics with Applications* 63(2), 544–553 (2012)
14. Golubev, N.: Android Forensics Study of Password and Pattern Lock Protection (October 28, 2011), <http://android-forensics.com/android-forensics-study-of-password-and-pattern-lock-protection/143> (retrieved)
15. Thing, V.L.L., Ng, K.-Y., Chang, E.-C.: Live memory forensics of mobile phones. In: 2010 Digital Forensics Research Workshop Published in *Digital Investigation*, vol. 7, pp. S74-S82 (2010)
16. Sylve, J., Case, A., Marziale, L., Richard, G.G.: Acquisition and analysis of volatile memory from Android devices. *Digital Investigation* 8(3-4), 175–184 (2012)
17. Wade, C.: (2012), <http://www.dingleberry.it/> (visited July 5, 2012)

ANTS ROAD: A New Tool for SQLite Data Recovery on Android Devices

Lamine M. Aouad, Tahar M. Kechadi, and Roberto Di Russo

Centre for Cybersecurity and Cybercrime Investigation
University College Dublin, Ireland
{lamine.aouad, tahar.kechadi}@ucd.ie

Abstract. Recovering deleted information is one of the most important probative elements in a forensic investigation that involves a mobile phone. In this paper, we present a new tool implementing an innovative method, based on a low-level analysis, to recover deleted data from SQLite databases on Android devices, taking as an initial example text messages. The paper then proposes a generic framework for deleted data recovery that can be used with a range of SQLite databases on a variety of Android systems and devices. Indeed, although our initial aim was to recover deleted SMSs, we realized along the way that, with the appropriate changes, the initial implemented method can be applicable to the extraction of deleted information from any SQLite database file.

1 Introduction

In the last decade or so, the world of mobile phones has gone through a tremendous change, transforming the devices from simple phones to pocket computers, mostly referred to as smart phones. Prior to this change, using a mobile phone meant by and large emitting and receiving calls or text messages (SMSs). Nowadays, with the spread of new hardware and software technologies, a mobile user can also surf the Web, chat, and do more or less everything he or she can do with a desktop or a notebook computer, and even more. The capability of these devices is still growing, as the number of their users. Indeed, they are today the highest-selling consumer electronic devices.

The smart phones proliferation has brought new issues in terms of forensics evidence acquisition. They handle a large amount of personal data, including text messages, communications logs, contacts, multimedia, geo-location information, etc. These could potentially help answering crucial questions in a criminal investigation. However, the huge variety of devices and the lack of standards imply that there is no unified method of accessing, extracting, or retrieving this data. Surely, a huge contribution to the smart phones market growth was brought by Android OS, by which an impressive amount of relatively low-price devices has been sold. According to data from Google, the activation rate is projected to reach a million per day by mid August of this year (2012), and if it continues we could see 1.5 million per day by the end of 2013. Android accounts for 68% share of the global smart phone market (2nd quarter of 2012).

These devices store most of the information in database files, which keep track of information deleted by the user, to a certain extent. However, this information cannot be accessed by traditional databases browsers and tools, and need alternative techniques.

This paper presents a new low-level analysis method for the recovery of deleted information from SQLite database files on Android devices. We particularly applied it to the recovery of deleted SMSs, then generalize it to other databases and devices. The next section presents related work and surveys some of the existing tools. Section 3 will then present the proposed method. Section 4 shows the initial evaluation supporting few databases from different Android releases and devices, along with a discussion and future work. Concluding remarks are then given in section 5.

2 Related Work

Android OS stores the information in a set of database files. These files are managed by SQLite, a database engine, also used by Mozilla Firefox, Thunderbird, Skype, Apple's iOS and Blackberry, among others. According to the SQLite documentation [2]: *“when you delete information from an SQLite database, the unused disk space is added to an internal free-list and is reused the next time you insert data. The disk space is not lost, but neither it is returned to the operating system”*.

About this deleted information, it adds: *“if you do not have a backup, recovery is very difficult. You might be able to find partial string data in a binary dump of the raw database file. Recovering numeric data might also be possible given special tools, though to our knowledge no such tools exist. (...) Recovery is also impossible if you have run vacuum since the data was deleted. If vacuum has not been run, then some of the deleted content might still be in the database file, in areas marked for reuse. But, again, there exist no procedures or tools that we know of to help you recover that data”*.

In a nutshell, the vacuum operation, mentioned earlier, rebuilds the database. It simply copies the contents of the database into a temporary database file and then overwrites the original with the contents of the temporary file. This procedure allows to reclaim the free marked space. It ensures that each table is stored contiguously and it may also reduce the number of partially filled pages. In auto-vacuum capable databases, the Database Management System executes the command automatically. The application designer or the database administrator has no control on the rebuilding operation. This feature is to keep in mind in the deleted records recovery. Indeed, since the user cannot know the last time that the vacuum command has been executed, he/she cannot know how many records can be recovered. Also, as a result of this operation, the recovery on the same database at different times will not necessarily return the same result set.

In [1], the authors have reported the recovery properties of few database systems, including SQLite, by comparing their behavior in terms of deletion, update, insert, and vacuum operations. An important behavior of SQLite to mention here is that data is deleted logically, and not removed. Previous values of a record are however completely overwritten in many cases during an update. Deleted records are freed and subsequent insertions may overwrite the data. However, freed pages are not returned to the file system until vacuum is performed. The recovery rate was also fairly low, at about 400 to 500 records throughout the tested workload, which was up to 30000. Although this is a completely different use case, including about hundred operations of insertion, update, deletion and so on, it shows the challenges of recovering information from this database

management system. In terms of software tools, Epilog [9] for Windows platforms, is the only dedicated tool to deleted data recovery from SQLite databases we could find. It includes three recovery algorithms that can be used on any SQLite database, regardless of the type of the data stored. Nevertheless, the tested version of Epilog, on few SMS databases, did not recover any of the deleted messages. This is more likely the result of the wide range of customization and differences in the database structure and fields across devices and OS versions.

On the other hand, there are many studies in the literature on the retention and recovery of deleted data from different underlying systems including file systems, memory, and even specific applications such as browsers and documents [11], [12], etc. There are also many forensic tools and methods performing physical and logical data acquisition, for Android and other devices, surveys on existing tools can be found in [3], [4], and few methods in [5], [6], [7], among many others. However, logical acquisition simply queries the databases, and therefore cannot extract information that are not accessible from an SQL browser. The existing literature is very limited, and deleted SMSs recovery from an Android device, and more generally any other deleted data source, can be considered as a relatively unexplored field. The existing forensic analysis support of database files, other than as physical memory dumps, is very limited. This work aims at covering this gap.

3 The Method

In order to extract deleted text messages from an Android device, we performed a low-level analysis on the related database file. Android stores all the information about SMSs and MMSs in the `mmsms.db` file. The data we are interested in resides in the `sms` table. For a forensically-sound analysis, we pulled out a copy of this database from the device.

3.1 SQLite Database Structure

Before explaining the method, let us present the structure of SQLite databases. It is composed of pages, most of which are organized in a B-tree structure. A page is a set of a fixed number of bytes, whose size is a power of 2, between 512 and 65536 inclusive. All the pages in the same database have the same size and they are numbered, starting from 1. Each page can have only a single use between the following.

- *Freelist pages*: pages that are not in active use anymore, and that are put in a linked list to be reused if additional pages are required.
- *B-tree pages*: a B-tree page can be an internal page or a leaf page. The content is stored only in the leaf pages, so it is on these pages that we focused our attention on. A B-tree page is either a table B-tree page or an index B-tree page. However, since the data is stored only in table B-tree pages, we focused only on these. The data stored in a B-tree table leaf page is organized in cells. Usually, but not always, each cell contains exactly one record.

- *Overflow pages*: sometimes the payload of a B-tree cell is too big to fill in a B-tree page. In these cases, the surplus is stored into an overflow page. The overflow pages form a chain, the first four bytes of every overflow page are a pointer to the next page, or have value 0 if the page is the end of the chain.
- *Pointer map pages*: pages whose aim is to make the auto-vacuum operation more efficient. They simply contain links between pages from child to parent. The first, and usually the only one, pointer map page is page 2. A pointer map page exists only if the database is auto-vacuum. It has been useful to our work to further shrink the set of candidate pages where to look for the deleted records.

3.2 Analysis Set Up

In every SQLite database, the first page contains the 100 bytes database header, that is divided into fields. The multibytes fields are stored in big-endian format. For our analysis, the most significant fields are listed in the following table. All offsets are intended from the beginning of the first page and all the sizes are expressed in bytes.

Offset	Size	Description
0	16	The header string "SQLite format 3\0".
16	2	The database page size in bytes.
52	4	If greater than 0, the database is auto-vacuum capable.
56	4	The database text encoding. A value 1 means UTF-8.

Before starting the analysis, it is necessary to check that the first 16 bytes match the string "SQLite format", followed by the null terminator character. If not, the database is not a valid SQLite database file, and this method cannot be applied. In our case, if the `mmssms.db` copy is not corrupted, it will pass the check.

At offset 16, important information is located, stored in 2-bytes big-endian format, it is the page size. This is useful for dividing the database into pages, the working units of the first part of our work, that is the selection of a candidate set of pages where to look for the deleted records. The Android SMSs database has a page size of 1024 bytes. The 4-bytes big-endian integer at offset 52 indicates if the database is auto-vacuum. This information is quite important here. Indeed, if this field has value 0, the database is not auto-vacuum capable and it is not possible to explore the pointer map stored in the second page. The `mmssms.db` file is auto-vacuum, which is the default.

At offset 56, there is a 4-bytes value that indicates the database text encoding. In our case, the relevant value is 1, which means UTF-8 encoding. Other values are 2 for UTF-16 little-endian and 3 for UTF-16 big-endian. For the other fields meaning, we refer the interested reader to the official SQLite documentation [2]. After the initial first page analysis, we went on with the analysis of the pointer map page that represents the starting point of the first selection of interesting pages, i.e. potential source of deleted text messages.

3.3 The Pointer Map Page Analysis

As already mentioned, in an auto-vacuum database, the second page represents a pointer map page, which aim is to facilitate moving the pages around in the database as part of performing the vacuum operation. It is a sort of lookup table that stores a 5 byte record for every page that follows the pointer map page. In these records the first byte indicates the page type, and the others 4 bytes, to read in big-endian format, are a reference to the parent page, indicated with `0x00 0x00 0x00 0x00` if it is null, or `0xVV 0xVV 0xVV 0xVV` (where `VV` stands for variable) otherwise.

- **0x01** `0x00 0x00 0x00 0x00`: a B-tree root page has no parent page.
- **0x02** `0x00 0x00 0x00 0x00`: a B-tree free page has no parent page.
- **0x03** `0xVV 0xVV 0xVV 0xVV`: the first page of an overflow chain. Its parent is the B-tree page containing the B-tree cell to which the overflow chain belongs.
- **0x04** `0xVV 0xVV 0xVV 0xVV`: a page that is part of an overflow chain, but that is not the first page. Its parent is the previous page in the overflow chain.
- **0x05** `0xVV 0xVV 0xVV 0xVV`: a page that is part of a table or index B-tree structure and is not a root page or an overflow page. Its parent is the page containing the parent tree node in the B-tree structure.

Interested readers can find a deeper analysis of the pointer map page in [2], or [8]. In our work, in order to perform an initial shrink of the candidate pages set, we kept only pages that are part of a table B-tree structure and pages in overflow chains. Then, we made a further selection in this set, keeping only the pages that are not child of the B-tree root page, because we empirically realized that they do not contain any useful data. By doing so, the searching set has been significantly reduced.

3.4 B-Tree Table Leaf Pages Analysis

Once we obtained the first candidate set, we started the pages analysis. First of all, we checked out the first byte of each page. If its value is `0x0D`, i.e. 13, it means that the page is a leaf node in the B-tree structure, so it contains data and it is a good candidate to contain deleted records.

The second and the third bytes of a page represent the relative offset of the first free space block inside this page. If this offset is zero, it means that in the page there is no free space, and then there cannot be any deleted record, since the space occupied by deleted records is considered to be free. We went on in the analysis selecting only the pages with at least one free space block, further shrinking the candidate set.

The next two bytes tell us the number of cells in the page (`nPages`), in our case the number of non-deleted SMS records. At relative offset 5, a 2-bytes field indicates the offset from the page starting of the first cell that contain a valid record, while the next byte is a null separator (`0x00`). Going on, there are (`nPages`) byte pairs, each one containing the relative offset of a valid content cell. Each 'pointed' cell containing a non-deleted SMS has a fixed structure, similar to the one shown in figure 1.

The payload length, the Row Id and all the payload header subfields are stored using a `VarInt` format. `VarInt` (*Variable Integer*) can take between 1 and 9 bytes, depending on the value stored. The Most Significant Bit (MSB) of each byte indicates if the next

Record Size	Row Id (Record Key)	Payload Header	Payload
-------------	------------------------	----------------	---------

(a) General record structure

Payload Header Size	NULL	thread_id	address	person	date	protocol	read	status	type	reply_path_present	subject	body	service_center	locked	error_code	seen
---------------------	------	-----------	---------	--------	------	----------	------	--------	------	--------------------	---------	------	----------------	--------	------------	------

(b) Payload header of a SMS record

Fig. 1. SMSs record structure

byte is also part of the field (1) or not (0), while the remaining 7 bits are used to store the value itself. For the `mms_sms.db` file it is enough to consider the case with at most two bytes VarInt, following this algorithm:

```

Let x be the value of the first byte
if x < 128 then
    result = x
else
    Let y be the value of the second byte
    result = (x - 128) * 128 + y
end if
    
```

Checking that the first byte is less than 128 is equivalent to checking that its MSB is 0. Indeed, if the x's most significant bit is 1, the result can also be computed by concatenating the latter 7 bits of the second byte to the latter 7 bits of the first one. Storing data in VarInt field allows to save space. Indeed, VarInts are big-endian that, using a static Huffman encoding, need less space for small positive values.

The data in the payload is stored in a serialized way: it is the Payload header that indicates how to identify each payload field. In fact, for each payload field there is a Serial Type Code, what we previously called "payload header subfields", that denotes its type of data, according to the following table. Note that we reported only the Serial Types that are relevant to the SMSs database.

Serial Type	Meaning
0	null field.
$N \leq 4$	big-endian 8*N bit two's complement integer.
5	big-endian 48 bit two's complement integer.
6	big-endian 64 bit two's complement integer.
$N \geq 13$ and odd	(N-13)/2 bytes string in the database encoding.

The payload header varies from one table to another. Each manufacturer can add custom fields to the shown structure. We can check how many fields are in the payload header simply by reading the payload header length. In a deleted SMS record there are some differences. Remembering that if in a page there is more than one free block, they are concatenated in a chain, the first two bytes form a pointer to the next free cell in

the chain; a value zero indicates that the current block is the last one. The third and the fourth bytes represent the size of the block in bytes, including the header. Both fields are big-endian integers.

After this *pre-header*, we find the payload header. In our analysis, we realized that the payload header lacks the record size and the record key. Indeed, three different cases are possible:

- It can start with the payload header size,
- With the NULL byte that precedes the Serial Type Code list, or
- Directly with the first Serial Type Code, in this case the `thread_id`.

Since the payload header is an important information about the record, but it is not always available, we computed it in advance simply by taking the minimum size between all the valid record (non-deleted SMSs) size. This is a valid method since all the SMS record sizes differ at most by one byte, depending on whether the body Serial Type Code needs one or two bytes. In this computation, we considered only the complete records that can be recognized by a payload header with at least the 16 basic Serial Type Codes.

There are also record fragments or records containing only the body, without any other field, but we did not consider them neither in the computation, nor in the analysis. Applying this method before starting to collect data, we further reduced the candidate set to the pages that contain only complete SMSs. Knowing the payload header size, the data collection has been performed using each Serial Type Code for reading the specified number of bytes and interpreting them according to the above table. Besides the chains of deleted cells, built by the first two bytes of each free block, we also managed the *internal chains*. We define an internal chain as a chain of deleted SMSs inside the same cell and it can be recognized by the fact that the record size indicated in the first two bytes is much bigger than the actual payload content. A simple counter is enough to understand the transition from a record to the next one in the same cell.

Deleted SMSs could also be found in the space between the page header and the first valid content cell. When this happens, the first bytes pair after the page header is greater than 0 and different from the last cell offset (that is, the last bytes pair in the page header) and it represents the offset, from the beginning of the page, of a deleted SMS. This SMS can be considered the head of a chain and its first field indicates the offset of the next element, usually the same offset indicated in the page header as the first free space block inside this page. Based on these patterns, experimentally induced from a large set of test data, we carried out a set of evaluations described in the next section.

4 Evaluation

We tested the method on two different mobile phones, mounting two different versions of Android OS: the LG Optimus One with Android 2.3.3 and the Samsung Galaxy Gio with Android 2.3.5.

On the Optimus One we recovered 22 deleted SMSs: 10 incoming, 10 outgoing, 1 draft and 1 of unknown type. On the Samsung Galaxy Gio, we recovered 6 deleted

SMSs: 3 draft, 2 outgoing and 1 incoming. Manually analyzing page by page the databases, we found 23 deleted SMSs on the Optimus One and 8 on the Galaxy Gio. The method recovered 22 out of 23 (95% of the deleted records) in the former, and 6 out of 8 (75%) in the latter case. In both cases, however, the unrecovered SMSs were partially filled or corrupted, i.e. partially overwritten by other SMSs. This means that the proposed method recovered all the complete deleted SMSs that were still present in the database files.

After one week of use, we repeated the tests. Since we did not delete any additional SMSs, the results on the Optimus One have not changed. On the Galaxy Gio, on the other hand, we deleted all the received and sent SMSs and the proposed method recovered only 2 SMSs, 1 draft and 1 outgoing. This indicates that an auto-vacuum operation has been performed on it. We also applied the vacuum command manually to the copies of the acquired databases. No deleted SMSs have been recovered. This confirms that the vacuum operation, both manual and automatic, remove all the deleted data that might have still been in the database. These results highlight how strong is the link between the deleted SMSs recovery, and more generally the deleted records recovery from an SQLite database, and the unpredictability of the vacuum operation execution.

4.1 Discussion

The method proposed in this paper implements an efficient way to recover complete deleted SMS records from a SQLite database on Android phones. Nevertheless, the applicability of this work remains subject to the vacuum operation. On the one hand, auto-vacuum is useful to our goal because it allows us to navigate the pointer map page and shrink considerably the candidate set of pages on which to carry on the deleted SMSs lookup. Indeed, if the database file was not auto-vacuum capable, its second page would not contain a valid pointer map and the lookup would have been performed on the whole set of the database pages, which can be quite big. This would have resulted with a considerable loss of efficiency. On the other hand, working with an auto-vacuum enabled database means that we do not have the control of the time of the last clean up. As a result, it is also possible that two recoveries made at different times on the same database return two different result sets.

4.2 Additional Use Cases

To understand what part of the deleted SMSs extraction can be reused, we analyzed two additional databases. Particularly, we tested other Android databases, namely `browser.db` and a proprietary database, WhatsApp's `msgstore.db`. In the browser database, we are interested in the `bookmarks` table that, despite its name, contains the whole Web history, recording for each visited link whether or not it is saved as a bookmark. While the general record structure is the same as mentioned for the SMSs, the payload header changes. We can see that in figure 2. As for the SMS record, we represented only the basic payload header, but each manufacturer can add custom fields to the shown structure. The information we are targeting is:

Payload Header Size	NULL	title	url	visits	date	created	description	bookmark	favicon	thumbnail	touch_icon	user_entered
---------------------	------	-------	-----	--------	------	---------	-------------	----------	---------	-----------	------------	--------------

Fig. 2. History browser item payload header

- title: item title shown on a browser tab,
- url: item url (what we actually see in the browser address bar),
- visits: how many times the url has been visited by the user,
- date: date/time of the last visit,
- bookmark: it indicates whether or not the url is saved as a bookmark.

The second database that we studied is `msgstore.db`. It is the database used to store messages sent and received by WhatsApp, a proprietary instant messaging App that uses the Web to send messages. In this database, we are interested in the `messages` table that contains all the sent and received messages. Again, the general record structure is the same as previously discussed, while the payload header, shown in figure 3, changes. In this case, however, the payload header is well-defined and it does not depend on the device manufacturer. For this database, we are targeting the following information.

- `key_remote_jid`: for incoming messages it is the sender id, for outgoing ones the recipient id,
- `status`: message status. It indicated if the message is incoming or outgoing, or if it has an unknown value,
- `data`: message body content,
- `timestamp`: the timestamp when the message has been created,
- `sendTimestamp`: the timestamp when the message has been sent,
- `receivedTimestamp`: the timestamp when the message has been actually received by the target user,
- `receivedServerTimestamp`: for outgoing messages, the timestamp when the message has been received by the WhatsApp server,
- `receivedDeviceTimestamp`: for outgoing messages, the timestamp when the message has been received by the recipient device.

Payload Header Size	NULL	key_remote_jid	key_from_me	key_id	status	needs_push	data	timestamp	media_url	media_mime_type	media_size	media_name	...
...	latitude	longitude	thumb_image	remote_resource	received_timestamp	send_timestamp	receipt_server_timestamp	receipt_device_timestamp					

Fig. 3. WhatsApp message payload header

As in the SMS records recovery, for both the Web history items and the WhatsApp messages, the payload length, the Row id and all the payload header subfields are stored using a VarInt format. The rules to compute these values are those we discussed in the previous section. However, this is not the only analogy with the deleted SMSs recovery. Indeed, we realized that most of the work made can be reused, from the pointer map

analysis to specific details such as the internal and the external chains management. The only aspect that changes is the field extraction itself. Since the payload header and, consequently the payload itself, is different from database to another, we need to redefine for every database how to extract each field. In practice, using the discussed method, we need to establish how to rebuild each record field from a byte group, whose size is indicated by the related Serial Code Type. Following this process, and implementing it in a modular way, we could obtain, with a relatively small effort, a generalized module to recover deleted information from a variety of databases.

4.3 Future Work

In order to recover more deleted information, one direction would be in trying to retrieve the previous versions of the database files and then apply the proposed method to each of them. However, since the auto-vacuum execution overwrites every time the previous database version with the same name, this should be thought of beforehand and probably integrated in the development process to facilitate potential forensic investigations. The analysis has been made on SQLite general assumptions, we have already showed how we investigated ways to reuse this work to recover deleted information from other Android databases. The following step is to consider a wider range of information stored by SQLite databases in a variety of systems and devices. Indeed, many other phones operating systems use SQLite to store data, including Apple's iOS and Blackberry. This method is not limited to Android systems and devices, and it should be sufficient to check the page size and to adapt the matching Serial Code Types/Payload Field to the specific table structure to obtain a specific-purpose software that runs for any auto-vacuum valid SQLite database. We will validate this generalization in our future work.

5 Conclusion

In this paper, we proposed and implemented a method for deleted information recovery from SQLite databases. The initial target was database files under Android OS. We discovered a set of patterns related to deleted information recovery on these databases and validate it with a set of use cases. This work has a potentially wide range of applications in the important area of mobile digital forensics. It also aims at setting blueprints for a wider range and deeper analysis involving additional databases and operating systems. Indeed, there is a gap in the literature in this area, and this work is addressing it by proposing a method and a tool implementing and documenting the acquisition and analysis of deleted information using a popular database engine.

References

1. Stahlberg, P., Miklau, G., Levine, B.: Threats to privacy in the forensic analysis of database systems. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, SIGMOD 2007 (2007)
2. The SQLite Official Documentation, <http://www.sqlite.org>

3. Hoog, A., Gaffaney, K.: iPhone forensics. Via Forensics White paper (2009)
4. Hoog, A.: Android Forensics - Investigation, Analysis and Mobile Security for Google Android. Elsevier (2011)
5. Aouad, L., Kechadi, T., Trentesaux, J., Le Khac, N.-A.: An Open Framework for Smartphone Evidence Acquisition. In: Peterson, G., Sheno, S. (eds.) *Advances in Digital Forensics VIII*. IFIP AICT, vol. 383, pp. 159–166. Springer, Heidelberg (2012)
6. Aouad, L., Kechadi, T.: Android Forensics: A Physical Approach. In: *The 2012 International Conference on Security and Management* (July 2012)
7. Quick, D., Alzaabi, M.: Forensic analysis of the Android file system YAFFS2. In: *Australian Digital Forensics Conference* (December 2011)
8. Rob, P., Coronel, C.: *Database Systems: Design, Implementation and Management*. Thomson Course Technology (2009)
9. The Epilog SQLite forensic tool, <http://www.ccl-forensics.com/Software/epilog-from-ccl-forensics.html>
10. Drinkwater, R.: Forensics from the sausage factory - An analysis of the record structure within SQLite databases. Technical report (May 2011)
11. Carrier, B.: *File System Forensic Analysis*. Addison Wesley (2005)
12. Rosenblum, M.: *Understanding data lifetime*. Stanford University (2006)

Evaluating and Comparing Tools for Mobile Device Forensics Using Quantitative Analysis

Shahzad Saleem, Oliver Popov, and Oheneba Kwame Appiah-Kubi

Department of Computer and Systems Sciences
Stockholm University, Forum 100, Isafjordsgatan 39
SE- 16440 Kista, Sweden
{shahzads, popov, okak}@dsv.su.se

Abstract. In this paper we have presented quantitative analysis technique to measure and compare the quality of mobile device forensics tools while evaluating them. For examiners, it will provide a formal mathematical base and an obvious way to select the best tool, especially for a particular type of digital evidence in a specific case. This type of comparative study was absent in both NIST's evaluation process and our previous work (Evaluation of Some Tools for Extracting e-Evidence from Mobile Devices). We have evaluated UFED Physical Pro 1.1.3.8 and XRY 5.0. To compare the tools we have calculated Margin of Error and Confidence Interval (CI) based on the proportion of successful extractions from our samples in different scenarios. It is followed by hypothesis testing to further strengthen the CI results and to formally compare the accuracy of the tools with a certain level of confidence.

Keywords: Digital Forensics, Mobile Device Forensics and tools, e-Evidence, Evaluation, Confidence Interval, Hypothesis Testing and Quantitative Analysis.

1 Introduction

The digital world as we know it today is becoming increasingly mobile, mostly based on the growing computational and communication capabilities of the small scale digital devices (SSDD) and the associated services. The rate of penetration of these devices is three times faster than the one of personal computers [1] and recent statistical studies by ITU, indicate that 86.7 individuals out of 100 are using a mobile device [2].

Indeed, mobile SSDD have literally become a sort of digital behavioral archives both on individual and collective levels. They are omnipresent recordings of all our activities, even the illicit ones. Hence, during investigations these digital archives can prove crucial in providing the evidence in furthering and/or resolving a potential legal case.

Although every investigation does not end up in a court, even then it is advisable to treat the entire investigative process in a forensically sound manner. Hence, one can produce evidence which is admissible in a court of law, if such a need arises. The term forensically sound and how digital evidence must be handled is stipulated by

many published documents (that contain principles, standards, rules and guidelines) such as IOCE's guidelines [3], RFC 3227 [4], Daubert's Principle [5], and Federal Rules of Evidence [6], [7].

Growth in the number of mobile device forensics (MoDeFo) tools is almost proportional to the volume and variety of mobile devices. These tools are rarely verified and validated by independent third parties. The evaluation results provided by the vendors are the only results available to the investigator for selecting the right tool in a particular situation.

National Institute of Standards and Technology (NIST), as an independent third party, realized the need to evaluate MoDeFo tools to facilitate the selection of a better tool for a particular scenario. Therefore, NIST has developed "Smartphone Tool Specifications [8]" and subsequently formulated "Smart Phone Tool Test Assertions and Test Plan" [9].

NIST has also evaluated some MoDeFo tools and published their results at CFTT-Mobile Devices Project's website [10]. Each tool has been evaluated individually and the results published for each tool separately [10]. Every test case is elaborated in a tabular format where one table represents the data regarding the single case. The outcomes of the evaluation process are presented as either pass or fail with some additional comments on anomalies. Neither a visualization of evaluation results nor a comparative study is conducted to help an investigator in selecting a better tool. The whole process of selection relies on use of heuristics rather than on provable formal procedures.

In the earlier published paper titled "Evaluation of Some Tools for Extracting e-Evidence from Mobile Devices" [11] the visualization of reliability assurance levels is provided for assisting the investigator to compare the tools together in order to select the better one. This paper tries to improve the selection of MoDeFo tool by using formal quantitative analysis methods.

While [11] addresses only the **reliability assurance** levels derived from NIST's specifications, the work presented in this paper deals with **accuracy** and **integrity protection** as discussed in Section 3.1. The tools we have evaluated are XRY 5.0 developed by Micro Systemation¹ and UFED Physical Pro 1.1.3.8 developed by Cellebrite². Mobile phones used to evaluate these tools were Nokia 5800 Xpress Music and Sony Ericsson Xperia X1.

Mathematical foundations by using quantitative analysis are provided to compare the tools for each type of digital evidence. In particular, we have calculated the confidence interval (CI) and the margin of error (MoE) for each tool based on the proportion of successful extractions. Both CI and MoE factors when studied together should help an investigator to select a better tool for a specific investigation.

By using inferential statistics we have further strengthened our findings and made the comparison process more obvious. Based on hypothesis testing we are able to formally compare the tools in order to determine which one performs better for a specific type of digital evidence. Graphical visualization of hypothesis testing results will simplify the comparison and selection process even further.

¹ <http://www.msab.com/>

² <http://www.cellebrite.com/>

We have organized our paper in five sections. First section is a brief introduction of the overall research and the relevance of the work. Brief discussion concerning digital and mobile device forensics is the subject of the second section. In this section, we have also outlined the forensic process model which has been followed. Third section is about the methodology and the performance measurements. It describes CI and MoE for evaluation of the tools. Finally hypothesis testing is employed to formally compare the tools together. The analysis and discussion of the results is presented in the fourth section, while the last one (fifth) is about conclusions and the direction of future work.

2 Digital and Mobile Device Forensics

Digital Forensics (DiFo) is a relatively new and rapidly evolving discipline of the traditional Forensics Science. Its roots can be traced back to 1984 [12][13]. One of the first definitions of the term came from First Digital Forensics Research Workshop (DFRWS) [14].

DiFo is related to digital evidence or data stored, transformed and transmitted using a computer, which can help to support or refute a theory about an offense or its critical elements [15]. Advancement and evolution in the field of digital systems has spurred the progress in DiFo as well, resulting in the development of four new branches namely:

1. Computer Forensics
2. Network Forensics
3. Database Forensics
4. Mobile Device Forensics.

In this paper, the focus is on the evaluation of MoDeFo tools. MoDeFo tools deal with the digital evidences found in mobile devices. Mobile devices, as indicated in Section 1, have become important archives of the daily human behavior thus making the topic of this research both important and interesting.

2.1 Mobile Device Forensics Process Model

Various organizations, working groups and standardization bodies such as DFRWS, SWGDE, CART, NIJ, TWGDE have tried to build a standardized vocabulary, remove inconsistencies and to formalize the terminologies and the overall process as well as sub-processes [16][17]. As a result some digital forensic process models have also been developed [12], [14], [18–29].

DiFo is applied on cases with varying circumstances, heterogeneous requirements and technologies so creating a single DiFo model that fits all is a challenge in itself. Moreover, we were working in a controlled laboratory environment; with a goal to find a better tool through evaluation and comparison of various available MoDeFo tools. So, we followed a condensed form of “Forensic Investigation Process Model For Windows Mobile Devices” [29], as depicted in Figure 1.

3 Tool Evaluation

Evaluation is a process used to ensure that a tool behaves satisfactorily and it meets the performance requirements. According to Matt Bishop “*Evaluation is a process in which the evidence for assurance is gathered and analyzed against criteria for functionality and assurance*”[31]. Formally, verification and validation are the two different approaches to evaluation.

Verification requires high expertise and knowledge of the source code that is not available in our case due to the commercial nature of the tools. Therefore, in our case, validation approach is selected. According to IEEE glossary, “validation is the process of evaluating a software system or component during, or at the end of, the development cycle in order to determine whether it satisfies specified requirements” [32]. During validation, we have tested whether the tool performs as intended. In addition we also worked to find out some statistical performance measures to provide a formal basis for matching MoDeFo tools together.

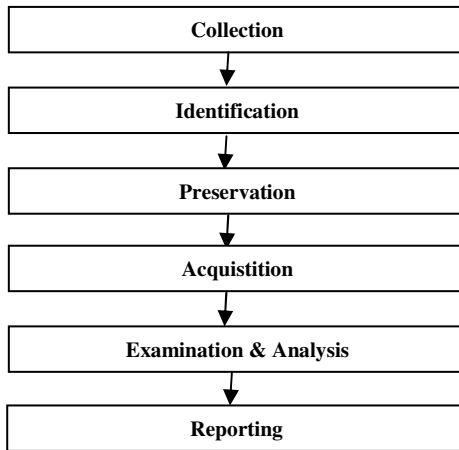


Fig. 1. The condensed form of the Windows Mobile Forensic Process Model [30]

3.1 Measuring Quality of MoDeFo Tools

The objective of the MoDeFo tools evaluation is to identify measures of their performance as criteria of quality. According to Carrier [27] DiFo tools operate by employing layers of abstraction. They transform raw bits and bytes into a presentable format which is human readable at the apex of the abstraction process. An abstraction layer transforms input data to output data by following a certain rule set, and of course, with some margin of error as depicted in Figure 2.

We validated the tools and calculated the individual and cumulative MoE induced by the underlying layers of abstraction for all types of digital evidence. The proportion of successful extractions of digital evidences was used as a base to calculate the performance indicators.

Carrier identified the following requirements, a MoDeFo tool must have: [27]

1. Usability: to address the complexity problem a tool must provide the data at a layer of abstraction that should help the investigator.
2. Comprehensive: the investigator must have access to all the data at the given layer of abstraction.
3. Accuracy: the MoE must be known to solve the “Error Problem” and to interpret the results accurately.
4. Deterministic: tool must produce the same output data when given the same input data and the rule set.
5. Verifiable: the ability to ensure accuracy of the tool by verifying its results either manually or by some independent third party tool.
6. Read Only: the ability to only read and not modify the original contents.
7. Sanity Checks: to detect any modification in the digital evidence.

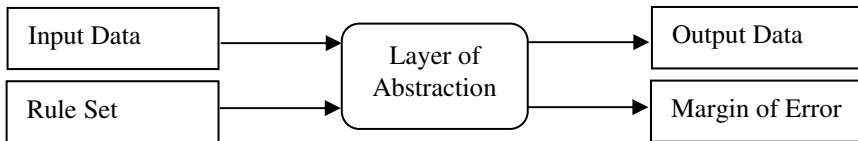


Fig. 2. Abstraction Layer Inputs and Outputs [27]

However, in our case these requirements have been condensed to **Reliability**, **Accuracy** and **Integrity Preservation**. Reliability includes the notions of usability, comprehension, determinism and verifiability. We have already measured and published [11] reliability assurance levels by following NIST smart phone tools specifications [8], smart phone test assertions and test plan [9].

The thrust of this work is to measure the accuracy and the integrity preservation capabilities of the MoDeFo tools by following B. Carrier’s requirements [27]. To do so, MoE and the CI were calculated for the proportion of successful extractions by the MoDeFo tools. Hypothesis testing was then used to not only strengthen the results of CI and MoE but also to formalize and automate the comparison process. Additional tests were done to determine the ability of MoDeFo tools in preserving the integrity of digital evidence. All these results will help an investigator to choose a better tool for a specific job.

B. Carrier [27] treats integrity preservation as a recommended feature. However, in case of MoDeFo it is very hard to detach the media from the mobile system. Consequently, the extraction is performed on a live mobile system. So, the extracted copy of the potential digital evidence becomes a snap shot of a particular system in a specific time. Some portions of the original data are eventually modified during the normal operations of the mobile device. Thus, there is no “original data” to compare with the extracted copy for the verification of its integrity. Therefore, preserving the integrity of digital evidence is a must have feature for the MoDeFo tools.

In traditional forensics a trained serologist can comment on the correctness of DNA by using the explanations from molecular biology, genetics and probability

theory [14]. Nevertheless, finding similar analogy is difficult in DiFo, because digital evidence is a transformation, representation and interpretation of reality.

Moreover, digital evidence is very fragile in nature and one can possibly modify it without being detected [33]. So, to avoid any ambiguities in such circumstances the tools should not only extract the data in a forensically sound manner (as explained in Section 1) but they must also preserve its integrity to make its admissibility more plausible.

3.2 Evaluation Methodology

NIST has developed an evaluation methodology in the field of DiFo. The project is called Computer Forensics Tool Testing Project (CFTT) [34]. One of the CFTT's branch is associated with testing of MoDeFo tools [10]. NIST has developed a set of Smartphone Tool Specifications [8] and Smartphone Test Assertions and Test Plan [9] to evaluate MoDeFo tools. We have followed them [8], [9] to measure the Reliability Assurance Level and published in our paper as well [11]. We also classified and published different types of digital evidences associated with mobile devices [11].

In this paper, the same classification as presented in [11] is used to extend our previous research work. All the data, processed while calculating CI, MoE and inferential statistics, comes from our previous work as well [11]. For the sake of reproducibility, we have again explained the procedure used to populate the potential digital evidences in the sample mobile devices.

To further extend the work described in [11], we have used "Quantitative Research Methodology" to evaluate the tools for MoDeFo in terms of their accuracy for retrieving the digital evidences. As discussed by B. Carrier [27] and presented in Section 3.1, we calculated point estimate of the proportion of successful extractions by the MoDeFo tools from our samples. Then we used those proportions to calculate MoE and CI. In our research, CI is an interval within which the success proportion will lie with 95% confidence level. We have used 95% confidence level because it is the number usually used in the scientific research [35].

In the second step, "hypothesis testing" is used to formally compare the MoDeFo tools by using one tailed tests. It assisted us in choosing the tool which performs better in terms of accuracy with 95% confidence level. Hypothesis testing is a concept related to CI so this test will strengthen our CI results as well. Towards the end we have tested the ability of the two tools to preserve the integrity of digital evidence.

Confidence Interval and Margin of Error. Estimating some point estimator for the population while dealing with a sample is merely a maximum likelihood estimator for the actual parameter of the population under consideration. For instance sample mean \bar{X} is a maximum likelihood estimator of the population mean μ . We know that \bar{X} will not exactly be equal to μ but it will be close. Basically, finding out the point estimator is of interest along with the determination of the interval within which the actual population parameter will lie (with a certain level of confidence) [36].

In the case being considered we are finding out both the estimations of MoE and CI based on the proportion of successful extractions. These measures will be useful in deciding the level of confidence in a specific tool. The higher the point estimates for the proportion of successful extractions, the lower the margin of error, and the narrower the range between upper and lower bounds of confidence interval, the better the tool is with respect to its performance and accuracy.

The equations to calculate CI are given below:

$$CI = p \pm MoE \dots\dots\dots\text{Equation 1}$$

$$MoE = z_{\alpha/2} * \sqrt{\frac{p(1-p)}{n}} \text{ When } n \geq 30 \dots\dots\dots\text{Equation 2}$$

$$MoE = t_{\alpha/2} * \sqrt{\frac{p(1-p)}{n}} \text{ When } n < 30 \dots\dots\dots\text{Equation 3}$$

$$p = x/n \dots\dots\dots\text{Equation 4}$$

Whereas:

CI = Confidence Interval

MoE = Margin of Error

p = Proportion of successful extractions

x = number of objects retrieved successfully

n = total number of objects populated

$z_{\alpha/2} = 1.96$ for 95% confidence level when $n \geq 30$ [37]

$t_{\alpha/2} = 2.05$ for 95% confidence level when $n < 30$ [37]

Hypothesis Testing and One Tailed Test: CI and MoE are calculated for each tool individually. Based on these performance measures, the investigator will still have to compare and eventually select a better tool manually. In order to overcome this problem, hypothesis testing as a formal comparison method is employed.

Testing a particular hypothesis concerning the unknown parameters of a population by using the sample data [38] was more interesting as compared to the explicit estimation of the unknown parameters. Hypothesis test is a “one tailed test” if the set of values lesser or greater than the critical value lies only on one side of the probability distribution, as shown in Figure 3 [39][40].

Rejection region, in case of left tailed test, lies below -1.645, hence z-score lesser than -1.645 will have enough evidence to not to accept H_0 with 95% confidence level. In this case we will have 0.05 probability of Type I Error [40]. Similarly, in the case of right tailed test the critical region lies on the right hand side of the probability distribution, with all the z-scores greater than 1.645. These values were used to interpret the hypothesis testing results.

The tests were done to compare the tools together for each category of digital evidence. This type of individual comparison is useful when an investigator has many tools at his disposal. This way, he can select different tools for different categories of digital evidences during the same investigative process e.g. UFED for call logs and XRY for SMS.

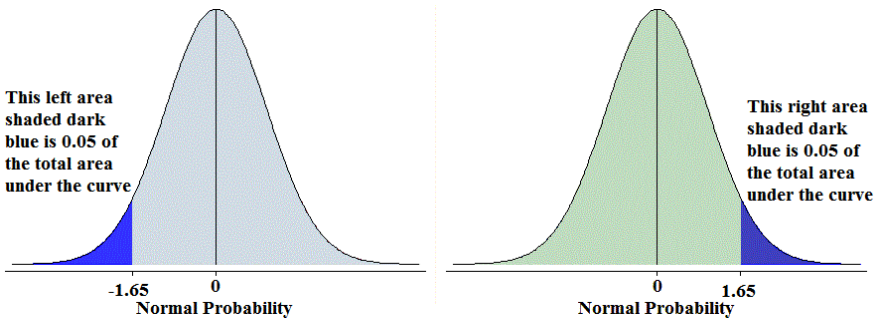


Fig. 3. One Tailed Hypothesis Testing [39]

The cumulative result for each tool is a combination of all the results in every category of digital evidence (assuming that every type of digital evidence is equally important and relevant). The hypothesis testing on the combined results comprehensively compared the tools. This type of analysis can help an investigator to select one tool for the entire investigative process based on the accuracy criterion.

Equations relevant to these statistics are the following:

$$z = \frac{p_1 - p_2}{S_{p_1-p_2}} \dots \dots \dots \text{Equation 5}$$

$$S_{p_1-p_2} = \sqrt{p * q \left(\frac{1}{n_1} + \frac{1}{n_2} \right)} \dots \dots \dots \text{Equation 6}$$

$$q = (1 - p) \dots \dots \dots \text{Equation 7}$$

$$p = \frac{(x_1 + x_2)}{(n_1 + n_2)} \dots\dots\dots\text{Equation 8}$$

$$p_1 = \frac{x_1}{n_1} \dots\dots\dots\text{Equation 9}$$

$$p_2 = \frac{x_2}{n_2} \dots\dots\dots\text{Equation 10}$$

Whereas:

x_1 , is the total number of objects retrieved by XRY

x_2 , is the total number of objects retrieved by UFED

$n_1 = n_2$, is the total number of objects populated in the mobile device

We have tested the following hypothesis:

Right Tailed Test:

$H_0 : p_1 \leq p_2$ Null hypothesis i.e. XRY does not perform better than UFED

$H_1 : p_1 > p_2$ Alternate hypothesis i.e. XRY performs better than UFED

Left Tailed Test:

$H_0 : p_1 \geq p_2$ Null hypothesis i.e. XRY performs better than UFED

$H_1 : p_1 < p_2$ Alternate hypothesis i.e. XRY does not perform better than UFED

With the theoretical aspects of the work outlined, the next step is to populate the mobile devices with potential digital evidences.

3.3 Population of Data Objects

The specifics of the data population process relative to mobile devices are well described in [30] and [11]. For the sake of reproducibility we will reiterate them in the following section.

Three different methodologies were used to populate data objects in the mobile devices.[41]

- 1) *Manual*: Using the normal handset interfaces only e.g. sending and receiving SMS via normal handset operations and a network of a mobile operator.
- 2) *Semi Manual*: Copying or moving data from a similar mobile device.
- 3) *Automatic*: Automated population of data objects e.g. with a tool or a software.

Since, timeline is critical in forensic science, so first of all the date and time was set in the sample mobile devices. The initial states were extracted and saved as “control states” to detect and eventually avoid possibility of any errors during the entire process.

1) *Sony Ericsson Xperia X1*

- a. *PIM Entries:* A total of 631 PIM (phonebook, calendar, note and task) entries were populated. Fifteen of them were manually deleted. We not only used both the mobile devices collaboratively to populate each other but also synchronized them with MS Office Outlook 2007. Contact entries include:
 - i. Special characters
 - ii. Blank entries
 - iii. Associated email addresses
 - iv. Associated picture or image
- b. *Message Entries:* Xperia X1 uses its internal memory to store all the types of messages. A total of 339 message entries were populated while 21 of them were manually deleted. We used two SIM cards by Lycatel³ and Tele2⁴ to manually populate the messages.
 - i. Lycatel provides free services for both SMS and EMS. So it was used to populate SMS (comprising both ASCII and Non-ASCII characters) and EMS entries (having both smileys and emoticons).
 - ii. Tele2 sim was used to populate MMS entries (containing audio, video and graphics).
- c. *Call Log:* A total of 295 call log entries were populated while 14 of them were manually deleted. Moreover, we also noted that switching off and then removing the SIM card does not affect the call logs in Xperia X1.
- d. *Emails:* A total of 444 email entries were populated while 399 of them were manually deleted. To populate emails, we connected Xperia X1 to our university WLAN and synchronized it with an existing email account (automated approach). We also used the mobile devices to create email entries via mobile operator's network (manual approach).
- e. *Internet History:* A total of 500 internet history entries were populated while 10 of them were manually deleted. We connected our mobile device to our university WLAN for accomplishing this task.
- f. *Standalone Files:* A total of 1629 standalone file entries, including audio, video and picture/graphic files, were populated while 386 of them were manually deleted (manual approach)
- g. *Application Files:* A total of 448 application file entries (including word, excel, power point, one note and pdf files) were populated while 5 of them were manually deleted.
- h. *GPS Entries:* GPS entries are also associated with pictures. So we used them to measure the performance of MoDeFo tools for GPS entries. We captured the pictures after enabling location services. These pictures were subsequently saved in the mobile device (manual approach) as standalone files of graphics type.

³ <http://www.lycatel.com/>

⁴ <http://www.tele2.se/>

2) *Nokia 5800 Express Music*

The approach to populate data objects in Nokia phone is the same as for SonyEricsson Xperia X1 with some minor difference in the total number of objects. The actual numbers are presented in Section 4.

3) *SIM Card:*

- a. *PIM:* A total of 246 PIM entries were populated. These entries were populated manually and also copied from the internal memory of our mobile devices.
- b. *Message:* A total of 30 message entries were populated while 10 of them were manually deleted.
- c. *Call Log:* A total of 11 call log entries were populated.

4 Results and Discussion

This section is about the results of the evaluation process. Initially, it deals with the results of CI and MoE. Then it proceeds with the formal comparison of the two tools by using hypothesis testing.

4.1 Margin of Error and Confidence Interval

The numbers, showed in four tables (1 through 4) depict:

1. Individual performance measures for each type of data objects.
2. Performance measure of MoDeFo tools for each class of data objects obtained by joining individual measures.
3. All the classes are also merged together to determine the cumulative performance measure of MoDeFo tools. It should be note that, merging the results in bullets 2 and 3 is based on an assumption that every object is equally important and relevant.

The tool with higher proportion of success, smaller MoE and thus higher confidence level is considered to be better and hence more appropriate to be used in a specific case.

Table 1 is about the evaluation results of both the MoDeFo tools for SonyEricsson Xperia X1 mobile device. Similarly, Table 2 is about the results of MoDeFo tools when applied on Nokia 5800 Xpress Music. The numbers in both the tables indicate that, in most of the cases examined, XRY is performing better than UFED.

4.2 Hypothesis Testing

The performance in terms of accuracy of the two MoDeFo tools is easily determined by comparing MoE and CI results. However, this type of comparison is not obvious, and it still has to done manually. Hypothesis testing as a formal method has a clear potential for automatic execution.

Tables 3, shows the hypothesis testing results for Xperia X1 with both the MoDeFo tools. It has a column with remarks showing whether we have sufficient evidence to reject the null hypothesis and to conclude that XRY performs better for a specific type

of data objects with 95% confidence level. Table 4, (just like Table 3) shows the hypothesis testing results for Nokia Xpress Music with both the MoDeFo tools.

Both the tables also show combined results for a specific class of data objects with an assumption that every type of data object is equally relevant and important. Similarly, all the classes are also joined together to perform hypothesis testing on all the data objects when seen together, again with same assumption as above.

Table 1. MoE and CI with Sony Ericsson Xperia X1 for both MoDeFo tools

		Sony Ericsson Xperia X1										
		Total Populated	Accurately Retrieved		Success Proportion (p)		Margin of Error		Confidence Interval			
			XRY	UFED	XRY	UFED	XRY	UFED	XRY		UFED	
								Lower	Upper	Lower	Upper	
PIM Entries	Phonebook/Contacts	307	292	292	0.951	0.951	0.024	0.024	0.927	0.975	0.927	0.975
	Calendar Entries	107	107	0	1.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000
	Memo/Notes	117	113	116	0.966	0.991	0.033	0.017	0.933	0.999	0.975	1.008
	Tasks/To-Do-Lists	100	95	0	0.950	0.000	0.043	0.000	0.907	0.993	0.000	0.000
	Total	631	607	408	0.962	0.647	0.015	0.037	0.947	0.977	0.609	0.684
Messages	SMS	185	185	179	1.000	0.968	0.000	0.026	1.000	1.000	0.942	0.993
	EMS	36	36	20	1.000	0.556	0.000	0.162	1.000	1.000	0.393	0.718
	MMS	118	30	0	0.254	0.000	0.079	0.000	0.176	0.333	0.000	0.000
	Total	339	251	199	0.740	0.587	0.047	0.052	0.694	0.787	0.535	0.639
Call Logs	Voice Calls	215	215	115	1.000	0.535	0.000	0.067	1.000	1.000	0.468	0.602
	Video Calls	80	80	80	1.000	1.000	0.000	0.000	1.000	1.000	1.000	1.000
	Total	295	295	195	1.000	0.661	0.000	0.054	1.000	1.000	0.607	0.715
Emails	Total	444	438	0	0.986	0.000	0.011	0.000	0.976	0.997	0.000	0.000
	URLs Visited	250	250	250	1.000	1.000	0.000	0.000	1.000	1.000	1.000	1.000
Internet History	Bookmarks/Favourites	250	150	0	0.600	0.000	0.061	0.000	0.539	0.661	0.000	0.000
	Total	500	400	250	0.800	0.500	0.035	0.044	0.765	0.835	0.456	0.544
	Audio	568	568	568	1.000	1.000	0.000	0.000	1.000	1.000	1.000	1.000
Standalone Files	Video	446	346	446	0.776	1.000	0.039	0.000	0.737	0.814	1.000	1.000
	Graphics/Pictures	615	515	615	0.837	1.000	0.029	0.000	0.808	0.867	1.000	1.000
	Total	1629	1429	1629	0.877	1.000	0.016	0.000	0.861	0.893	1.000	1.000
	Word	146	146	146	1.000	1.000	0.000	0.000	1.000	1.000	1.000	1.000
Application Files	Excel	42	42	42	1.000	1.000	0.000	0.000	1.000	1.000	1.000	1.000
	PowerPoint	130	130	130	1.000	1.000	0.000	0.000	1.000	1.000	1.000	1.000
	PDF	130	130	130	1.000	1.000	0.000	0.000	1.000	1.000	1.000	1.000
	Total	448	448	448	1.000	1.000	0.000	0.000	1.000	1.000	1.000	1.000
Grand Total		4286	3868	3129	0.902	0.730	0.009	0.013	0.894	0.911	0.717	0.743

Table 2. MoE and CI with Nokia 5800 Xpress Music for Both MoDeFo Tools

		Nokia 5800 Xpress Music										
		Total Populated	Accurately Retrieved		Success Proportion (p)		Margin of Error		Confidence Interval			
			XRY	UFED	XRY	UFED	XRY	UFED	XRY		UFED	
								Lower	Upper	Lower	Upper	
PIM Entries	Phonebook/Contacts	277	272	267	0.982	0.964	0.016	0.022	0.966	0.998	0.942	0.986
	Calendar Entries	107	107	0	1.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000
	Memo/Notes	66	66	0	1.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000
	Tasks/To-Do-Lists	105	95	0	0.905	0.000	0.056	0.000	0.849	0.961	0.000	0.000
	Total	555	540	267	0.973	0.481	0.013	0.042	0.959	0.986	0.440	0.523
Messages	SMS	147	142	142	0.966	0.966	0.029	0.029	0.937	0.995	0.937	0.995
	EMS	37	32	32	0.865	0.865	0.110	0.110	0.755	0.975	0.755	0.975
	MMS	133	133	0	1.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000
	Total	317	307	174	0.968	0.549	0.019	0.055	0.949	0.988	0.494	0.604
Call Logs	Voice Calls	238	235	0	0.987	0.000	0.014	0.000	0.973	1.002	0.000	0.000
	Video Calls	45	39	0	0.867	0.000	0.099	0.000	0.767	0.966	0.000	0.000
	Total	283	274	0	0.968	0.000	0.020	0.000	0.948	0.989	0.000	0.000
Emails	Total	389	389	0	1.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000
	URLs Visited	250	0	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Internet History	Bookmarks/Favourites	250	0	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Total	500	0	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Audio	626	621	112	0.992	0.179	0.007	0.030	0.985	0.999	0.149	0.209
Standalone Files	Video	462	412	412	0.892	0.892	0.028	0.028	0.863	0.920	0.863	0.920
	Graphics/Pictures	621	521	521	0.839	0.839	0.029	0.029	0.810	0.868	0.810	0.868
	Total	1709	1554	1045	0.909	0.611	0.014	0.023	0.896	0.923	0.588	0.635
	Word	145	143	0	0.986	0.000	0.019	0.000	0.967	1.005	0.000	0.000
Application Files	Excel	40	36	0	0.900	0.000	0.093	0.000	0.807	0.993	0.000	0.000
	PowerPoint	130	129	0	0.992	0.000	0.015	0.000	0.977	1.007	0.000	0.000
	PDF	152	151	0	0.993	0.000	0.013	0.000	0.981	1.006	0.000	0.000
	Total	467	459	0	0.983	0.000	0.012	0.000	0.971	0.995	0.000	0.000
Grand Total		4220	3523	1486	0.835	0.352	0.011	0.014	0.824	0.846	0.338	0.367

It is evident from Table 3 that for most of the data objects, we have sufficient evidence (z -score > 1.645) to reject the null hypothesis and to conclude that XRY performs better when Xperia X1 is used as a source of digital evidences. The tools (XRY and UFED) are equally good/bad in the case of:

1. Phonebook/Contacts, equally good.
2. Video Calls, equally good with 100% success proportion for both the tools.
3. URLs visited, equally good with 100% success proportion for both the tools.
4. Audio Files, equally good with 100% success proportion for both the tools.
5. Both the tools are equally good for all the types of application files with 100% success proportion.

Table 3. Hypothesis Testing with Xperia X1 for Both MoDeFo Tools

		SonyEricsson Xperia X1									
		Total Populated	Accurately Retrieved		Success Proportion		$p = \frac{X_1 + X_2}{n_1 + n_2}$	$q = (1 - p)$	Standard Deviation	Z-Score	Remarks
		$n = n_1 + n_2$	XRY	UFED	X_1	X_2					
PIM Entries	Phonebook/Contacts	307	292	292	0.951	0.951	0.951	0.049	0.017	0.000	Equally Good
	Calendar Entries	107	107	0	1.000	0.000	0.500	0.500	0.068	14.629	Yes
	Memo/Notes	117	113	116	0.966	0.991	0.979	0.021	0.019	-1.356	No
	Tasks/To-Do-Lists	100	95	0	0.950	0.000	0.475	0.525	0.071	13.452	Yes
	Total	631	607	408	0.962	0.647	0.804	0.196	0.022	14.119	Yes
Messages	SMS	185	185	179	1.000	0.968	0.984	0.016	0.013	2.470	Yes
	EMS	36	36	20	1.000	0.556	0.778	0.222	0.098	4.536	Yes
	MMS	118	30	0	0.254	0.000	0.127	0.873	0.043	5.863	Yes
	Total	339	251	199	0.740	0.587	0.664	0.336	0.036	4.227	Yes
Call Logs	Voice Calls	215	215	115	1.000	0.535	0.767	0.233	0.041	11.415	Yes
	Video Calls	80	80	80	1.000	1.000	1.000	0.000	0.000	NA	Equally Good
	Total	444	438	0	0.986	0.000	0.493	0.507	0.034	29.339	Yes
Emails	Total	389	389	0	1.000	0.000	0.500	0.500	0.036	27.893	Yes
Internet History	URLs Visited	250	250	250	1.000	1.000	1.000	0.000	0.000	NA	Equally Good
	Bookmarks/Favourites	250	150	0	0.600	0.000	0.300	0.700	0.041	14.639	Yes
	Total	500	400	250	0.800	0.500	0.650	0.350	0.030	9.945	Yes
	Standalone Files	Audio	568	568	568	1.000	1.000	1.000	0.000	0.000	NA
Video	446	346	446	0.776	1.000	0.888	0.112	0.021	-10.613	No	
Graphics/Pictures	615	515	615	0.837	1.000	0.919	0.081	0.016	-10.433	No	
Total	1629	1429	1629	0.877	1.000	0.939	0.061	0.008	-14.597	No	
Application Files	Word	146	146	146	1.000	1.000	1.000	0.000	0.000	NA	Equally Good
	Excel	42	42	42	1.000	1.000	1.000	0.000	0.000	NA	Equally Good
	PowerPoint	130	130	130	1.000	1.000	1.000	0.000	0.000	NA	Equally Good
	PDF	130	130	130	1.000	1.000	1.000	0.000	0.000	NA	Equally Good
	Total	448	448	448	1.000	1.000	1.000	0.000	0.000	NA	Equally Good
Grand Total	4380	3962	2934	0.905	0.670	0.787	0.213	0.009	26.836	Yes	

We cannot reject the null hypothesis and thus conclude that XRY performs better for just three types of data objects:

1. Memo/Notes: Here, UFED is actually performing a bit better. Nevertheless, we cannot conclude (by using left-tailed test) that UFED performs better than XRY with 95% confidence level.
2. Video: Using left-tailed test, we can conclude with 95% confidence level that UFED performs better than XRY for this type of data objects.

3. Graphics/Pictures: Using left-tailed test, we can conclude with 95% confidence level that UFED performs better than XRY for this type of data objects.
4. Standalone Files: We can conclude with 95% confidence level that UFED performs better than XRY for this class of digital evidences including audio, video and graphics files.

For the rest of the eight types of data objects, XRY performs better with 95% confidence level. We can reject the null hypothesis, with 95% confidence level, and conclude that XRY performs better than UFED for the combined performance measures of all the types of data objects in Table 3, with an assumption that every data object is equally important and relevant.

Similarly results in Table 4 provide enough evidence to reject the null hypothesis, with 95% confidence level, for most of the types of digital evidences, and to conclude that XRY performs better than UFED. If we merge the results of all the objects of a class together, with an assumption that all of them are equally important, then we can see that we still have enough evidence to reject the null hypothesis for all the classes of digital evidence (except Internet History) with 95% confidence level. Thus we can conclude that XRY performs better than UFED for all the classes of objects except “Internet History”. Both the tools did not extract even a single digital evidence of the “Internet History” class. This amounts to “equally bad” performance by both tools for this data class of digital evidence.

Table 4. Hypothesis Testing with Nokia 5800 for Both MoDeFo Tools

		Nokia 5800 Xpress Music									
	Total Populated	Accurately Retrieved		Success Proportion		Weighted Proportion		Standard Deviation	Z-Score	Remarks	
		XRV	UFED	XRV	UFED	$p = \frac{x_1 - x_2}{ n_1 + n_2 }$	$q = (1 - p)$				
	$n = n_1 + n_2$	X_1	X_2	$p_1 = \frac{x_1}{n_1}$	$p_2 = \frac{x_2}{n_2}$	$p = \frac{x_1 - x_2}{ n_1 + n_2 }$	$q = (1 - p)$	$S_{p,q} = \sqrt{pq \left(\frac{1}{n_1} + \frac{1}{n_2} \right)}$	$z = \frac{\hat{p} - \hat{q}}{S_{p,q}}$	H_0 Rejected	
										XRY is Better	
PIM Entries	Phonebook/Contacts	277	272	267	0.982	0.964	0.973	0.027	0.014	1.309	No
	Calendar Entries	107	107	0	1.000	0.000	0.500	0.500	0.068	14.629	Yes
	Memo/Notes	66	66	0	1.000	0.000	0.500	0.500	0.087	11.489	Yes
	Tasks/To-Do-Lists	105	95	0	0.905	0.000	0.452	0.548	0.069	13.171	Yes
	Total	555	540	267	0.973	0.481	0.727	0.273	0.027	18.394	Yes
Messages	SMS	147	142	142	0.966	0.966	0.966	0.034	0.021	0.000	Equally Good
	EMS	37	32	32	0.865	0.865	0.865	0.135	0.079	0.000	Equally Good
	MMS	133	133	0	1.000	0.000	0.500	0.500	0.061	16.310	Yes
	Total	317	307	174	0.968	0.549	0.759	0.241	0.034	12.345	Yes
Call Logs	Voice Calls	238	235	0	0.987	0.000	0.494	0.506	0.046	21.544	Yes
	Video Calls	45	39	0	0.867	0.000	0.433	0.567	0.104	8.296	Yes
	Total	283	274	0	0.968	0.000	0.484	0.516	0.042	23.046	Yes
Emails	Total	389	389	0	1.000	0.000	0.500	0.500	0.036	27.893	Yes
Internet History	URLs Visited	250	0	0	0.000	0.000	0.000	1.000	0.000	NA	Equally Bad
	Bookmarks/Favourites	250	0	0	0.000	0.000	0.000	1.000	0.000	NA	Equally Bad
	Total	500	0	0	0.000	0.000	0.000	1.000	0.000	NA	Equally Bad
Standalone Files	Audio	626	621	112	0.992	0.179	0.585	0.415	0.028	29.200	Yes
	Video	462	412	412	0.892	0.892	0.892	0.108	0.020	0.000	Equally Good
	Graphics/Pictures	621	521	521	0.839	0.839	0.839	0.161	0.021	0.000	Equally Good
	Total	1709	1554	1045	0.909	0.611	0.760	0.240	0.015	20.397	Yes
Application Files	Word	145	143	0	0.986	0.000	0.493	0.507	0.059	16.796	Yes
	Excel	40	36	0	0.900	0.000	0.450	0.550	0.111	8.090	Yes
	PowerPoint	130	129	0	0.992	0.000	0.496	0.504	0.062	16.001	Yes
	PDF	152	151	0	0.993	0.000	0.497	0.503	0.057	17.321	Yes
	Total	467	459	0	0.983	0.000	0.491	0.509	0.033	30.042	Yes
Grand Total	4220	3523	1486	0.835	0.352	0.593	0.407	0.011	45.142	Yes	

Moreover, both the tools performed equally good/bad for:

1. SMS, equally good.
2. EMS, equally good.
3. URLs visited, equally bad (0% success proportion for both the tools)
4. Bookmarks/Favorites, equally bad (0% success proportion by both the tools)
URLs visited and Bookmarks constitute Internet History class, so we can say that for this class of digital evidence both the tools performed equally bad.
5. Videos, equally good.
6. Graphics/Pictures, equally good.

There is just one object type of phonebook/contacts where there is not enough evidence to reject the null hypothesis. In this case, XRY is performing slightly better than UFED and its z-score is 1.309 which is slightly lesser than 1.645, thus we cannot reject the null hypothesis with 95% confidence level. However, when the confidence level is reduced to 90% then there is enough evidence to reject the null hypothesis, and conclude that XRY performs better than UFED.

With 95% confidence level, one can conclude for the rest of the twelve types of objects that XRY performs better than UFED. If we combine all the objects together, again with an assumption that every type of object is equally important and relevant, then we can say, on cumulative base, that XRY performs better than UFED with 95% confidence level.

Regarding SIM Card analysis, both tools had 100% success proportion for extracting the Contacts, SMS/EMS and Call Logs entries. This leads to just one conclusion i.e. both XRY and UFED are 100% accurate in this area. So, there was neither a need to calculate CI and MoE nor to perform hypothesis testing for SIM card analysis.

4.3 Integrity Preservation

The central ideas behind integrity preservation are (1) to preserve the data, and (2) to report on data modifications (if any). The procedure to examine integrity preservation feature in both tools is outlined below:

1. The images from the mobile devices with both the MoDeFo tools were extracted.
2. The contents of the images were modified by using WinHex 15.6. An entry in contacts was modified by changing a contact name from “Shamm” to “55ura”.
3. We reopened both image files with XRY and UFED.

XRY could not identify the modification and opened the file with the modified contact name appearing in its contact entries report window. On the other hand, UFED successfully identified the modification and reported with a “File Corrupted” error message. So in this regard, UFED came on the top.

The use of a secure platform in the form of smartcards to preserve the integrity of digital evidence is proposed as one of the plausible solutions for the above problem [33].

5 Conclusions and Future Work

Two mobile devices, Xperia X1 and Nokia 5800, are used to evaluate two MoDeFo tools i.e. XRY 5.0 and UFED Physical Pro 1.1.3.8.

5.1 Conclusion

The first step translated to computing MoE and CI in order to compare the performance of both tools. The results indicated that XRY is better than UFED for most of the object types, which we studied. But the comparison was neither obvious nor formal. The investigator has still to retain and manually compare the numbers to select a better tool with lesser margin of error, greater success proportion and better confidence level.

Finally, hypothesis testing was used to make the comparison process more obvious. The results of this process helped to conclude, with 95% confidence level, that XRY performs better than UFED for most of the object types. If we assume that all the object types are equally important and relevant than we can also reject the null hypothesis and make an overall conclusion that XRY performs better than UFED with 95% confidence level.

Comprehensive visualization of hypothesis testing results is provided in Figure 4. It shows that most of the vertical bars are above the threshold z-score value of 1.67 for the right tailed test. It provides enough evidence to reject the null hypothesis ($H_0 : p_1 \leq p_2$, XRY does not perform better than UFED) and therefore to accept the alternate hypothesis ($H_1 : p_1 > p_2$, XRY performs better than UFED) for most types of digital evidences found in the mobile devices.

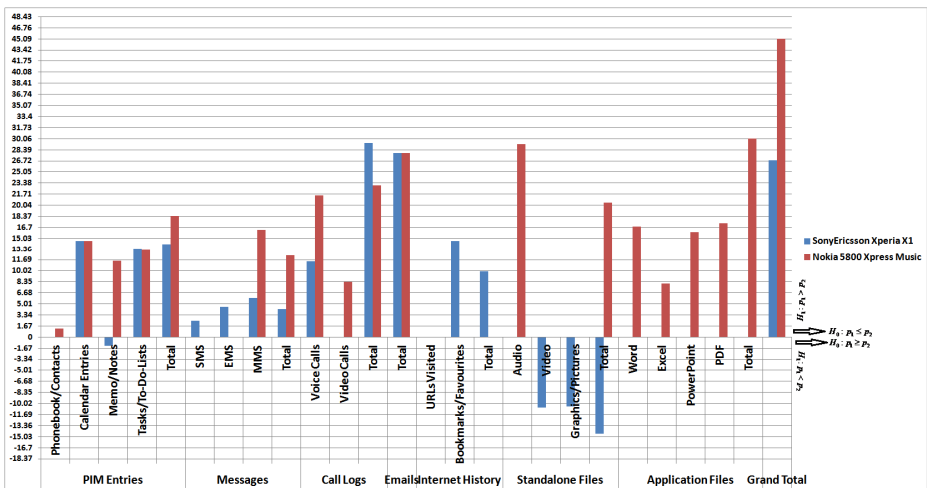


Fig. 4. Visualization of Hypothesis Testing Results

Figure 4 helps in a rapid selection of the appropriate tool for a particular type of digital evidence involving a specific type of mobile device. Another important observation in Figure 4 is that, if a tool performs better for a specific type of digital evidence for one mobile phone then it will also perform better for the same type of digital evidence with other mobile phone. However, at this stage, this rule cannot be generalized as it has an exception as well – the “Memo” type digital evidence.

UFED performs better than XRY as far as preserving the integrity of digital evidence is concerned.

In a nut shell, this paper is about a generic technique, which can be extended both vertically and horizontally. It means that any number of mobile devices and MoDeFo tools can be studied by this technique. Therefore, it will help in selecting the most appropriate MoDeFo tool for any specific incident.

5.2 Future Work

Although the results of CI, MoE and hypothesis testing can help in selecting a better tool for a particular type of digital evidence, in some way we may consider that cumulative comparison is somewhat false. This type of cumulative comparison asks for combining all the types of digital evidences with an assumption that all the types of digital evidences are equally important and relevant, which may not be true in most of the real life scenarios.

Despite the possible fallacy in the assumption the comparison results are necessary when an investigator has to choose just one tool for the entire investigative process. Especially in the circumstances when the relevance of different types of digital evidences in solving or furthering a particular case is known in advance. Hence, there must be a way to compare the MoDeFo tools by considering both performance and relevance of different types of digital evidences as two different criteria of quality. For these criteria, there must also be a mechanism to represent real life scenarios by mapping various degrees of importance and relevance.

In future, we will try to carve a generic model. The model will accommodate multiple criteria to obtain an overall ranking of the available MoDeFo tools by combining the results with varying degrees of importance and relevance. It will help in selecting the most appropriate MoDeFo tool, which may lead to the generation of better digital evidence.

References

- [1] Techsling, Personal Computers Outnumbered by Mobile Phones (2010), <http://www.techsling.com/2010/10/personal-computers-outnumbered-by-mobile-phones/> (accessed March 28, 2012)
- [2] International Telecommunication Union (ITU), ICT Data and Statistics (IDS) (2011), http://www.itu.int/ITU-D/ict/statistics/material/excel/2011/Mobile_cellular_01-11_2.xls (accessed March 28, 2012)

- [3] International Organization on Computer Evidence, IOCE - Guidelines for Best Practice in the Forensic Examination of Digital Technology (2002)
- [4] Brezinski, D., Killalea, T.: RFC 3227: Guidelines for Evidence Collection and Archiving (2002)
- [5] Daubert v. Merrell Dow Pharmaceuticals (92-102), 509 U.S. 579 (1993), <http://www.law.cornell.edu/supct/html/92-102.ZS.html> (accessed February 29, 2012)
- [6] Weissenberger, G., Duane, J.J.: Federal Rules of Evidence: Rules, Legislative History, Commentary, and Authority (2004)
- [7] Federal Evidence Review, Federal Rules of Evidence 2012 (2012), <http://federalevidence.com/downloads/rules.of.evidence.pdf> (accessed June 10, 2012)
- [8] National Institute of Standards and Technology (NIST), Smart Phone Tool Specification, Version 1.1 (2010)
- [9] National Institute of Standards and Technology (NIST), Smart Phone Tool Test Assertions and Test Plan, Version 1.1 (2010)
- [10] National Institute of Standards and Technology (NIST), CFTT- Mobile Devices, <http://www.nist.gov/itl/ssd/cs/cftt/cftt-mobile-devices.cfm> (accessed June 6, 2012)
- [11] Kubi, A., Saleem, S., Popov, O.: Evaluation of some tools for extracting e-evidence from mobile devices. *Application of Information and Communication Technologies* (10), 603–608 (2011)
- [12] Baryamureeba, V., Tushabe, F.: The enhanced digital investigation process model. In: *Proceedings of the 4th Annual Digital Forensic Research Workshop*, pp. 1–9 (2004)
- [13] Noblett, M.G., Church, F., Pollitt, M.M., Presley, L.A.: *Recovering and Examining Computer Forensic Evidence*. 2(4) (October 2000)
- [14] Palmer, G.: *A Road Map for Digital Forensic Research*, Utica, New York (2001)
- [15] Casey, E.: *Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet*, 3rd edn. Academic Press (2011)
- [16] United States Computer Emergency Response Team, *Computer Forensics US-CERT* (2008)
- [17] Meyers, M., Rogers, M.: Computer forensics: the need for standardization and certification. *International Journal of Digital Evidence* 3(2), 1–11 (2004)
- [18] Kent, K., Chevalier, S., Grance, T., Dang, H.: Guide to integrating forensic techniques into incident response, pp. 80–86. NIST Special Publication (August 2006)
- [19] Carrier, B.: An event-based digital forensic investigation framework. In: *Proceedings of Digital Forensic Research Workshop* (2004)
- [20] Jeong, R.S.C.: FORZA – Digital forensics investigation framework that incorporate legal issues. *Digital Investigation* 3, 29–36 (2006)
- [21] Beebe, N.L., Clark, J.G.: A hierarchical, objectives-based framework for the digital investigations process. *Digital Investigation* 2(2), 147–167 (2005)
- [22] Agarwal, A., Gupta, M., Gupta, S., Chandra, S.: Systematic Digital Forensic Investigation Model. *International Journal of Computer Science and Security* 5(1), 118–131 (2011)
- [23] Carrier, B.: Getting physical with the digital investigation process. *International Journal of Digital Evidence* 2(2), 1–20 (2003)
- [24] Reith, M., Carr, C., Gunsch, G.: An Examination of Digital Forensic Models. *International Journal of Digital Evidence* 1(3), 1–12 (2002)
- [25] National Institute of Justice, *Electronic crime scene investigation: A guide for first responders* (2001)

- [26] Noblett, M.G., Pollitt, M.M., Presley, L.A.: Recovering and examining computer forensic evidence. *Forensic Science Communications* 2(4), 102–109 (2000)
- [27] Carrier, B.: Defining digital forensic examination and analysis tools using abstraction layers. *International Journal of Digital Evidence* 1(4), 1–12 (2003)
- [28] Shin, Y.-D.: New Model for Cyber Crime Investigation Procedure. *Journal of Next Generation Information Technology* 2(2), 1–7 (2011)
- [29] Ramabhadran, A.: *Forensic Investigation Process Model For Windows Mobile Devices*. Tata Elxsi Security Group, pp. 1–16 (2007)
- [30] Appiah-Kubi, O.K.: *Evaluation of UFED Physical Pro 1.1.3.8 and XRY 5.0: Tools for Extracting e-Evidence from Mobile Devices*. Stockholm University (2010)
- [31] Bishop, M.: *Evaluating Systems*. In: *Computer Security: Art and Science*, p. 571. Addison-Wesley Professional (2002)
- [32] Radatz, J., Geraci, A., Katki, F.: IEEE standard glossary of software engineering terminology. IEEE Standards Board, New York, Standard IEEE std (1990)
- [33] Saleem, S., Popov, O.: Protecting Digital Evidence Integrity by Using Smart Cards. *Digital Forensics and Cyber Crime* 53, 110–119 (2011)
- [34] National Institute of Standards and Technology, “Computer Forensics Tool Testing (CFTT) Project, <http://www.cftt.nist.gov/> (accessed: February 26, 2012)
- [35] Attia, A.: Why should researchers report the confidence interval in modern research. *Middle East Fertility Society Journal* 10(1), 78–81 (2005)
- [36] Ross, S.M.: Interval Estimates. In: *Introduction to Probability and Statistics for Engineers and Scientists*, 3rd edn., pp. 240–241. Elsevier Academic Press (2004)
- [37] University of Leicester, *Online Statistics* (2000), <http://www.le.ac.uk/bl/gat/virtualfc/Stats/ttest.html> (accessed: June 16, 2012)
- [38] Ross, S.M.: Hypothesis Testing. In: *Introduction to Probability and Statistics for Engineers and Scientists*, 3rd edn., p. 291. Elsevier Academic Press (2004)
- [39] UCAL Academic Technology Services, What are the differences between one-tailed and two-tailed tests? http://www.ats.ucla.edu/stat/mult_pkg/faq/general/tail_tests.htm (accessed: June 16, 2012)
- [40] Easton, V.J., McColl, J.H.: *Statistics Glossary V1.1* (1997), <http://www.stats.gla.ac.uk/steps/glossary/index.html> (accessed: June 16, 2012)
- [41] Jansen, W., Delaitre, A.: *Mobile forensic reference materials: A methodology and reification*. US Department of Commerce, National Institute of Standards and Technology (2009)

Detection of Masqueraded Wireless Access Using 802.11 MAC Layer Fingerprints

Christer Idland, Thomas Jelle, and Stig F. Mjøl̄snes

Department of Telematics
Norwegian University of Science and Technology,
Trondheim
{christer.idland,thomas.jelle,sfm}@item.ntnu.no

Abstract. Many wireless Internet access operators prefer open local area network (WLAN) access because this reduces the need for user assistance for a variety of smaller devices. A 802.11 MAC spoofer masquerades as an authorized user and gains access by using an already whitelisted MAC address. We consider the scenario where the spoofer waits until the authorized user has finished the session, and then uses the still whitelisted MAC address for the network access. We propose and experiment with “implementation fingerprints” that can be used to detect MAC layer spoofing in this setting. We include eight different tests in the detection algorithm, resulting in 2.8 in average Hamming distance of the tests. Eleven different STA devices are tested with promising detection results. No precomputed database of fingerprints is needed.

Keywords: WLAN, 802.11, wireless, media access layer, masquerading, intrusion detection, network forensics, communication fingerprints.

1 Introduction

1.1 The Problem

Many wireless Internet access operators choose to provide a cryptographically unprotected wireless local area network (WLAN) link because this simplifies the user configurations for a variety of smaller devices, makes the wireless association faster, and reduces the cost of the user help-desk. Still, connecting to a wireless local area network for the first time may not be hassle-free for the user, because the setup also depends on user input for authentication, service selection, and payment. The operator can use a so-called *captive portal* for the purpose of user access control. A captive portal responds to any Hypertext Transfer Protocol (HTTP) client request (normally a web browser) with a special user authentication web page. All other uplink packets from the client will be blocked by the portal. The response page will give information about the internet access service, the operator, the access policy and accepted payment services. Once the client submits the proper credentials, then the Medium Access Control (MAC) address of the user’s WLAN network interface card (NIC) is whitelisted in the portal and subsequent packets are routed normally.

IEEE distributes and manages the allocation of the MAC addresses on a global basis. The manufacturers of the 802.11 NICs manage the assignment of a unique MAC to each device produced. This MAC identifier is stored in the hardware or firmware of the NIC, but can in many instances be modified by an attacker for the purpose of masquerading as an authorized user, for instance in the simple access control based on checking the MAC address performed by a captive portal. This is often called *MAC spoofing attack*.

Theoretically, this threat of masqueraded attacks does not come as a surprise, because only the user is authenticated by the captive portal, whereas the communicating devices and all their subsequent data communication are left without any authentication at all. The IEEE 802.11 standard provides security mechanisms for establishing a common symmetric authentication key between the client station (STA) and the access point (AP), where each link data frame is protected by a message authentication code that enables verification by the receiver. If a MAC spoofer attacker does not have access to the secret authentication key, then it will become computationally impossible to generate the correct message authentication codes, and the data frames from the spoofer will be rejected by the AP. In practice, the setup of the cryptographic keys will require extra user input, which works against user convenience and operator preferences.

Several techniques have been proposed for detecting a spoofing attack *while* the victim of the spoofing attack is actively connected to a cryptographically unprotected WLAN. It is an open problem whether it is possible to detect a MAC spoofing attack when the victim is no longer connected to the AP. The problem addressed in this paper is how to detect MAC spoofing when only the masquerading NIC is actively accessing the AP. An automatic detection of this type of spoofing attacks requires new algorithms for distinguishing between the authorized user and the masquerading attacker. Our distinguisher algorithms presented here are based on observing the heterogeneity of different implementation characteristics of the 802.11 protocol. We investigate how distinctive features of the various 802.11 implementations create NIC fingerprints, and how these can be used in the detection of MAC spoofing.

1.2 Motivation

The Internet access network *Wireless Trondheim* is a city wide wireless access network, mainly based on the IEEE 802.11a/b/g technologies (Wi-Fi). Currently, the network consists of approximately 500 access points. The geographical coverage is in the Trondheim city center outdoor area of about 1.5 km². In addition, the indoor area of all the buildings of the Municipality of Trondheim plus the coverage area of other central buildings. Wireless Trondheim provides Internet access service to about 15.000 unique users each month. Typically, 50% of the client terminals will use the WPA2 Enterprise solutions, while the other half will use the captive portal solution with authentication but with no encryption.

The main goal of Wireless Trondheim is to provide easy wireless Internet access for its users on a wide range of wireless equipment, including simple low-end devices, such as music players and simple mobile phones. Moreover,

the wireless network shall provide an arena for testing new services in a real environment (Living Lab). This implies that access control mechanisms must be as simple to operate and use as possible, keeping minimal requirements of the hardware and software of the wireless terminal.

On the other hand, cyber crime activities may be carried out by stealing user names and network addresses (IP, MAC) copied from other terminals. If a criminal act has been carried out and becomes investigated, then we want to make sure that the culprit is found and accused, and not some innocent third person. This implies the need for a strong authentication system, and works against the requirement of easy access for any user device.

Wireless Trondheim's motivation for finding solutions to the identity theft problem is to avoid that innocent users are wrongly accused of serious crime. A possible scenario is if the police or other authority requests information about the identity and activity of users during the process of serious crime investigations, where Wireless Trondheim can only provide possibly incriminating information about customers that in reality are innocent. This can obviously happen if an attacker hijacks the session of a user to hide his identity while doing criminal activity. A successful impersonator could perform criminal activities under another innocents users identity and access. The innocent user will be prosecuted and could eventually be found guilty in serious crime. The prime motivation of Wireless Trondheim for deploying a spoofing detection service is therefore to be able to check whether it is likely that there has been an identify theft before handing over user information to the police or other authorities. The algorithms presented here can also be used in an intrusion detection system (IDS) may block the attack, but this blocking will require reliable digital evidence because the costs of refusing a legitimate user are potentially high, the commercial and reputational risks for the service provider, as well as the denial-of-service and possible false allegations against the subscribed user.

2 Background and Related Work

The Norwegian University of Science and Technology and the company Wireless Trondheim started a research collaboration in 2008 taking on the practical problem of detection of masqueraded wireless access in the 802.11 networks. This research activity has spurred several master projects and theses at Department of Telematics, NTNU. In previous work, techniques have been proposed and tested for detecting MAC spoofing while the victim is active. In particular, we based the detection techniques on MAC sequence numbers and other logical properties of the 802.11 MAC layer [1].

This paper considers the scenario where the spoofer waits until the authorized user has finished the session, and then take advantage of the still whitelisted MAC address for the network access. Many of the results reported in this paper are based on the master thesis work and supervision of Idland [2]. Here we publish the main results, and put them in the wider problem context of operational architecture and practice.

Franklin et al. [4] exploit the fact that the channel scanning algorithm searching for available APs is not explicitly defined in the 802.11 protocol. They develop a method based on statistical analysis of the interframe timing of transmitted probe requests in order to identify a specific driver, and the conclusion is that the majority of wireless drivers do have a distinct fingerprint.

There are important differences in the processing of the Null Data frames in various implementations and we make good use of this observation. Gu et al. [7] create seven rules to identify different behavior regarding the Null Data frames. They focus on the fact that this distinguishing implementation feature may allow an attacker to recognize and determine the location of a client station. Location is in this case limited to an AP, for instance at a coffee shop, at home, or at school. These rules form a basis for the algorithm we present here, but we will use the rules to detect, and not aid attacks.

There exists several commercially available wireless intrusion detection systems (IDSs) that claim to detect and even prevent MAC spoofing attacks. One product in particular, the HSMX from fdXtende [8], uses a captive portal functionality. The manufacturer claims that HSMX will detect MAC address spoofing and prevent hijacking. It turns out that the hijacking prevention is based on an active SSL window technique that might not run on low-end devices. Furthermore, the MAC spoofing detection algorithm is based on not accepting more than one single IP address for a given MAC address, thus it does not defend against attacks where the victim is no longer online.

3 The Threat Model

3.1 The Wireless Access Network

Wireless Trondheim manages an open 802.11 access network where 802.11a/b/g are supported. The Wi-Fi Multimedia feature is enabled. Figure 1 presents a simplified overview of the network structure, depicting the relevant components for our purposes here. The AP functionality is split into two separate devices, which are the Lightweight Access Point (LAP) and the Wireless LAN Controller (WLC). One WLC entity can control several LAPs entities. The captive portal functionality is integrated in an Internet Gateway entity (Nomadix). The IDS server can receive the fingerprinting parameter values possibly from the WLC, or make the acquisition itself by eavesdropping on the wireless link directly. A whitelisted MAC address in the captive portal will remain whitelisted for up to 60 minutes after traffic has ceased. Obviously, a shorter time to white list flush will reduce the user's convenience by having to repeat a log-in after a break, and reduce the time available for the attacker.

3.2 MAC Spoofing

An easy way to fool an access control system based on whitelisting of MAC addresses is by performing a MAC spoofing attack. The theory is that the attacker

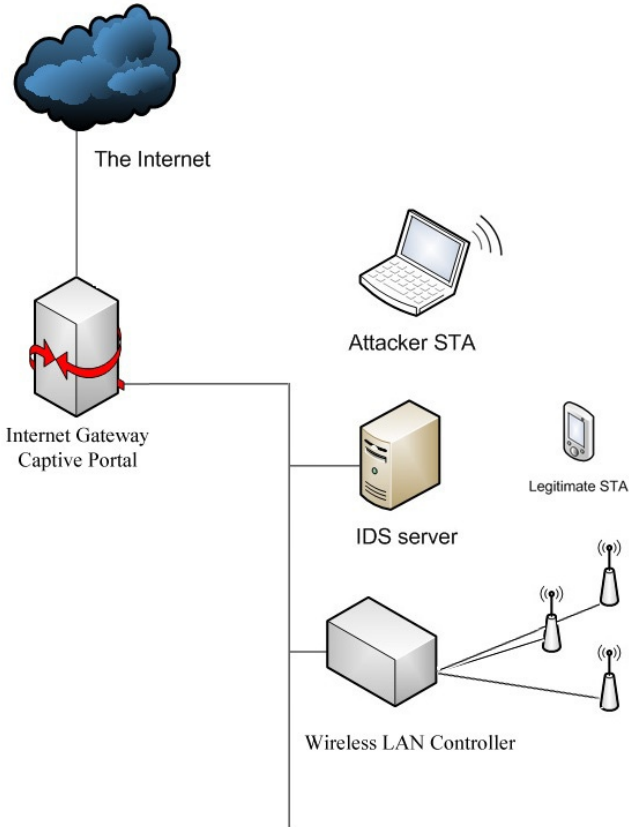


Fig. 1. The main network elements of the wireless access network. Several lightweight access points (LAP) are connected to a wireless LAN controller (WLC). The IDS server can receive fingerprinting properties from the WLC, or monitor the wireless links directly.

masquerades as a legitimate client, that is, a client that already has his MAC address whitelisted. Thereby the attacker gains access to the network. A suitable MAC address is easily obtained through eavesdropping of the victim's wireless communications.

Several methods for detecting attacks based on simultaneous access exists. The focus of this paper is on the MAC spoofing attack where the attacker and the victim do *not* need to be connected simultaneously. The main problem focus is on the wait-for-availability attack as it is the easiest to perform from the attacker's viewpoint. One can also argue that it is the most difficult attack to detect because the attacker does not force the legitimate client off the network. An attacker can force a victim off the network connection by sending a **deauthenticate** or **channel switch** frame to the client [3]. These MAC management frames can

easily be observed by an IDS, ergo it can easily be detected. Our attack scenario is where the attacker waits until there has not been any frame coming from the victim’s station (STA) for a significant period of time. Then, inferring that the victim has left, the attacker takes on the MAC address and tries to continue using the wireless access network.

4 Algorithms

4.1 Distinctive Features

The behavior of MAC layer STAs differs in many ways due to implementation differences of the 802.11 protocol. Most of these differences exist because the standard is not explicit, and therefore open to alternative interpretations. Some distinctive features are a result of different options and capabilities of the specific NIC in question. These distinctive features, called fingerprinting properties, are based on the rules for Null Data frame behavior found in Ref. [7]. We augment their list with several other distinctive features found during our research. Our list is presented in Table 1.

4.2 The Tests

The pseudocode for all algorithms described can be found in Appendix and Ref [2].

Test 1, PS-Poll Test 1 is based on the reported feature that some NICs use Null Data frames and others use PS-poll frames for power management. That is somewhat imprecise in relation to what we observed in the experiments. All STAs in the experiment used Null Data for changing power mode, but one STA used PS-Poll frames when the AP had buffered frames to send.

Therefore, the test is whether a STA use PS-Poll or not. If a STA use PS-Poll then the fingerprinting algorithm should observe a PS-Poll frame from the STA after the AP has announced that it has buffered frames for it. This PS-Poll frame should be observed within the `Listen interval` of that STA.

If the frame is a Beacon frame and the bit in the TIM corresponding to the STA in question is set then the `beacon_count` is incremented. If the `beacon_count` exceeds the `Listen interval` then the algorithm concludes that PS-Poll is not in use and returns suspect attack if it previously was in use. When a PS-Poll frame is observed the `beacon_count` is reset and the algorithm naturally concludes that PS-Poll is in use, if that was not the case before the algorithm will return “suspect attack.”

Test 2, Keep Alive. This algorithm tests whether a STA sends a Null Data frame if it has been idle for 10 seconds in order to keep the session alive. The timestamps are in milliseconds and thus testing for exactly 10 seconds would yield very few hits, therefore the test is implemented with a buffer, currently set to 0.15 seconds.

Table 1. Fingerprinting properties and their possible values for 802.11 network interface implementations [2]

Fingerprinting Property	Possible Values
PS-Poll	True / False
Keep Alive	True / False
Null before Probe	True / False
Mode changing Null Data	True / False
Fixed Interval	True / False
Null Data Type	Regular/QoS including TID
Duration Calculation	Pairs of (rate, duration) for each rate
Association Request Duration	{0...32767}
Listen Interval	{0...256}
Supported Rates	Set of up to eight integers $\in \{2...127\}$
Extended Supported Rates	Set of up to 255 integers $\in \{2...127\}$
QoS Capability	Present / Not Present
Vendor Specific	Type of vendor specific element

First the `time_delta` is calculated from the timestamp of the previous frame and the current frame. Then the algorithm checks if the `time_delta` is within the range of 10 seconds \pm the buffer of 0.15 second and in addition if the frame is a Null Data frame. If both of these conditions are true then the algorithm concludes that keep alive is in use and returns suspect attack if keep alive was not in use prior to this frame.

If the first if-conditions fail then a new if-statement checks if the `time_delta` is larger than 10.15 seconds (10 + buffer). If true then this indicates that the STA is not using keep alive, and thus the algorithm concludes so and returns suspect attack.

Test 3, Null before Probe. The rationale behind this test is that some STA sends a Null Data frame and enters PS mode before starting the channel scanning with Probe Request frames while other STAs do not do this.

First a set size and a minimum limit are defined. The set size is the number of probe request bursts that will be observed before any conclusion is made. The minimum limit is a number $\in [0, 1]$ representing the percentage of probe request bursts where the STA first sends a Null Data frame required in order for the algorithm to conclude that null before probe is in use.

If the current frame is a Probe Request frame then the algorithm continues. If the previous frame was a Null Data frame then null before probe is in use and the algorithm increments the `using_count` as well as the `total_count`. On the other hand, if the previous frame was not a Null Data frame or a Probe Request frame then null before probe is not in use and the algorithm only increments the `total_count`.

The last part of the algorithm checks if the percentage of times null before probe was in use is over the minimum threshold in order to conclude if it in fact is in use. Suspect attack is returned whenever the current conclusion differs from the previous conclusion.

Test 4, Mode Changing Null Data. Test 4 checks if the STA, when it has data frames to send, uses a Null Data frame to change mode or if it directly sends a regular data frame when changing power mode. Note that some STAs always use mode changing Null Data except when they have been in PS mode for a duration equal to their own `Listen interval`, this is therefore included in the test.

First a set size and a minimum limit are defined. The set size is the number of power mode changes that will be observed before any conclusion is made. The minimum limit is a number $\in [0, 1]$ representing the percentage of power mode changes that must be done by using Null Data frames in order for the algorithm to conclude that mode changing Null Data is in use.

The first if-statement checks whether the STA has been in PS mode longer than its `Listen interval`, and if so changes the recorded power mode of the STA to AM.

The next if-statement checks if the `chk_nxt_pkt` variable is set. The first time the algorithm is executed this is not the case and the algorithm continues to check if the STA is in PS mode. If the STA is in PS mode (before the current frame) the algorithm checks if the current frame is a Null Data frame that changes the power mode to AM. If this is the case, the `chk_nxt_pkt` variable is set. If it is not a Null Data frame with `pwr_mgt` bit `== 0`, but a regular data frame with `pwr_mgt` bit `== 0`, then the algorithm interprets this as the STA is not using null before probe and therefore increments the `total_count` without incrementing the `using_count`.

The reason for having the `chk_nxt_pkt` variable is that we are only interested in the power mode changes made when the next packet is a data packet. So, when the next packet is processed in the algorithm the `chk_nxt_pkt` variable is set and the algorithm checks if the current packet is a regular data packet. If this is the case the `using_count` is incremented as well as the `total_count`.

The last part of the algorithm works the same way as in Test 3, by returning suspect attack if the overall conclusion has changed since last time.

Test 5, Fixed Interval. This test checks the duration between Null Data frames with different values in the `Power Management` bit. If the STA is using mode changing Null Data then this interval would translate into the time the STA was in PS mode. The idea behind this test is that some STA stays in PS mode for a *fixed* interval, regardless of which data to transmit.

First a new Null Data frame is detected. If the `pwr_mgt` bit of the previous Null Data frame was 1 (PS) and the value in the current frame is 0 (AM) then this pair of Null Data frames will be examined further. If the measured time interval value between these two frames is within a preset normality range, then the counter for successful detection of pairs (`pair_ok_count`) is incremented.

The `pair_total_count` is incremented regardless of the result of the outcome of this comparison. In the experiments, the time interval average was computed from the first 10 time intervals of Null Data frames measured, and the normality range was heuristically set to $\pm 20\%$ of this average. The last part of the algorithm computes the fraction of `set_size` Null Data frame pairs where the time interval falls within the normality range. If this fraction is greater than a threshold parameter then the algorithm concludes that the STA does use a fixed time interval for the Null Data frames. The algorithm returns “suspect attack” if the result of the previous test run was the opposite of this run.

Test 6, Null Data Type. Test 6 is based on observations regarding Null Data behavior. Recall that the 802.11 standard has two types of Null Data frames; the regular one and the QoS Null Data frame. It turns out that in a network where QoS is enabled (as it is in Wireless Trondheim) some STAs use the QoS Null Data while others do not use it. Amongst those who use the QoS enabled there is differences in which QoS priority (TID class) they utilize. These implementation differences make Test 6 viable.

The algorithm for Test 6 is relative simple compared to the other test algorithms. Nevertheless, it identifies a viable fingerprinting property. The algorithm basically identifies the type of Null Data frame in use and its priority class (TID) if it was a QoS frame. The algorithm then returns suspect attack if the identified frame differs from the previous identified Null Data frame.

Test 7, Duration Calculation. Test 7 was motivated by the paper by Gopinath et al. [5]. The theory is that Null Data frames have the exact same size, and thus should have the same duration when the data rate is the same. A difference in the `Duration/ID` field would indicate different implementations in the calculation algorithm that again indicates two different STA.

It basically works by recording the duration taken from the `Duration/ID` field in the Null Data frames for each data rate used. Recall that the data rate is available in the radiotap header. If an inconsistency is found then the algorithm outputs suspect attack.

Test 8, Association Request. Vendor specific extensions as a fingerprinting source is mentioned in the paper by Gopinath et al. [5]. This was the motivation to further investigate the Association Request frame looking for possible sources for fingerprinting. Several potential fields were identified and they are: `Duration/ID`, `Listen interval`, `Supported Rates`, `Extended Supported Rates`, `QoS Capability` and `Vendor Specific`.

When the input frame is an Association Request frame, the relevant fields mentioned above, called implicit identifiers, are recorded. If an Association Request has been recorded for this MAC address before a check on each individual field is done and suspect attack is returned in case of any inconsistencies.

4.3 Creating a Compound Fingerprint

In order to avoid a high false reject rate when using fingerprinting it is important to rely on several different properties and tests that can flag suspicious behavior. The fingerprints created in our experiments are a composition of properties determined by the eight tests explained above. The first six tests result in one fingerprinting property each, while the two latter results in several properties. Each of the properties from the two latter tests can be used to detect suspicious behavior in their own right. See Table 1 for an exhaustive list of fingerprinting properties and their possible values that make up the compound fingerprint. For a more in-depth explanation of the possible values and their usage consult the 802.11 standard [6].

Our MAC spoof detection algorithm comprises a combination of each of the eight tests described above, and some additional logic. The logic to determine whether we are dealing with an attack based on the output from the tests has not been described. We want to do fingerprinting on the fly and not necessarily generate a complete fingerprint. The question of how many properties that are needed, and a selection of optimal parameters in order to achieve acceptable low false positive and negative rates is open for further work. Some of the tests might prove to be sufficient by itself, while others require at least one other test in combination in order to conclude attack with a high probability. The experiments and the following results will shed some light on these questions.

5 Experiments and Results

5.1 The Terminal Equipment Test

We implemented the fingerprinting algorithm in Perl in order to test the different distinctive features of the STAs. Table 2 shows an overview of the STAs that were used in the experiments. The list includes laptops, smartphones and music players, in addition to a typical attacker setup (laptop with Backtrack 5), and are all included in the test.

5.2 Test Scenarios

Scenario 1. General Usage This scenario is constructed to test a browsing behavior. Two behavioral patterns were considered important to include when creating this scenario. First, the STA should continuously generate data traffic for a period of time, resulting in little or no idle time. Second, the STA should have longer periods where there is no traffic to send. The rationale is that this is a realistic usage pattern, such as browsing the web, start reading or watching something, then continue the web browsing. A scenario including these patterns should also be able to elicit different power management and Null Data behavior. All STAs of Table 2 were tested in this scenario.

Table 2. Overview of the STA devices used in the experiments

No.	Name and Model	OS	NIC	Browser
S-1	Dell XPS	Windows 7	Intel Wi-Fi 1000bgn	IE 8
S-2	Lenovo S10-3S	Windows 7	Broadcom 802.11n	Opera 11
S-3	Acer Aspire 5745G	Windows 7	Broadcom 802.11n	IE 8
S-4	Dell Inspiron 9400	Windows 7	Intel P/W 3945abg	Chrome 11
S-5	iPhone 1. gen.	iOS 3.0.1	Not available	Safari
S-6	iPhone 4. gen.	iOS 4.3	Not available	Safari
S-7	Dell Latitude D610	Windows 7	Intel P/W 2915abg	IE 8
S-8	Acer Aspire 5670	Win. XP SP3	Intel P/W 3945abg	Firefox 3.6
S-9	iPod touch 1. gen	iOS 3.1.3	Not available	Opera Mini 5
S-10	HTC Hero	Android 2.2.1	Not available	"Internet"
S-11	Dell Inspiron 9400	Backtrack 5	Intel P/W 3945abg	Firefox 4

Scenario 2. Wait-for-availability Attack Here the user is operating for approximately eight minutes, with some periods of high traffic, and other periods with little or no traffic, similar to the behavior of Scenario 1. Then the user logs off and the attacker spoofs the user’s MAC address and performs the wait-for-availability attack. One pair of STA devices were randomly chosen, S-5 and S-7, and tested in this scenario.

Scenario 3. Concurrent Usage This scenario is similar to Scenario 2. The main difference is that now there are five additional STAs connected to the LAP, all generating concurrent traffic. In other words, a total of seven STAs are connected and generating traffic. The scenario is also shortened down to a total of eight minutes (four minutes before the attack and four minutes after). This fact certainly makes it harder for the algorithm to determine the fingerprints as it will be less communication data available for analysis. The question is whether four minutes of monitoring in an active network is enough, or if longer monitoring period, such as the ones in Scenario 1 and 2, are required.

5.3 Results

Each of the 7 first distinctive features corresponds to one test, the next 4 features are gathered in a single test because they all depend on the association request frame. That leaves us with 8 different test/ distinctive features. Figure 2 shows the hamming distance between any two of the STAs used in the experiments, note that the maximal distance of 8 could be obtained if two STAs differed on every single test.

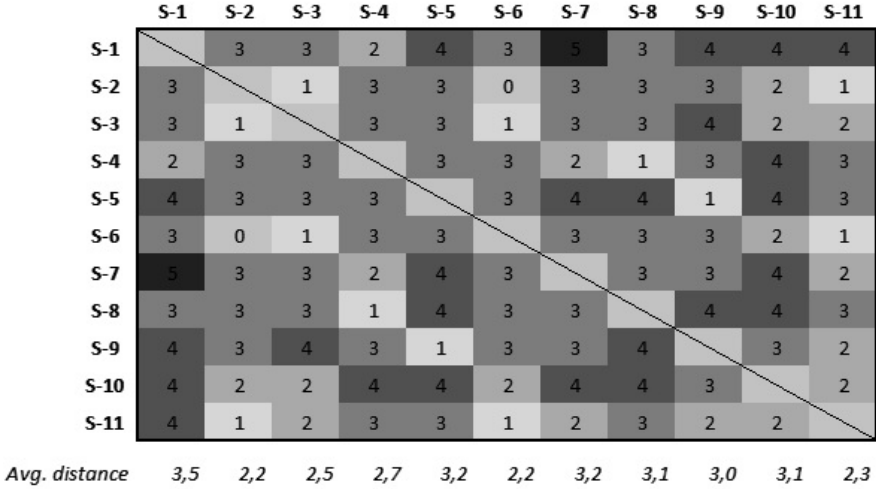


Fig. 2. Hamming Distance for the fingerprints in Scenario 1 [2]

Table 3. Results from Scenario 1 on tests 1-8 for S-1 to S-6. Undetermined entries are shown as "—".

Fingerprinting Property	S-1	S-2	S-3	S-4	S-5	S-6
PS-Poll	F	F	F	F	F	F
Keep Alive	T	F	F	—	F	F
Null before Probe	T	—	T	T	T	—
Mode changing Null	F	F	F	T	F	F
Fixed Interval	F	F	F	F	F	F
Null Data Type	QoS0	Reg.	Reg.	QoS0	QoS7	Regular
Duration Calculation	314	44	44	314	258	44
Ass. Req. Duration	60	314	213	314	314	314
Listen Interval	10	10	10	10	10	10
Supported Rates	s1	s2	s3	s3	s3	s2
Ext. Sup. Rates	e1	e2	e1	e1	e1	e2
QoS Capability	F	F	F	F	F	F
Vendor Specific	v1-v4	v1,v5	v1-v4	v1	v1	v1,v5

Scenario 1. The results from each of the eight tests performed are presented in Table 3 and Table 4. The first column presents the different fingerprinting properties, each of the other columns represents one STA. The first five fingerprinting features are either present (T) or not (F). Algorithms 1-5 determine these values.

Table 4. Results from Scenario 1 on tests 1-8 for S-7 to S-11. Undetermined entries are shown as "—".

Fingerprinting Property	S-7	S-8	S-9	S-10	S-11
PS-Poll	F	F	F	T	F
Keep Alive	F	F	F	F	F
Null before Probe	T	T	F	—	F
Mode changing Null	T	T	F	F	—
Fixed Interval	T	F	—	F	—
Null Data Type	Reg.	QoS0	QoS7	Reg.	Regular
Duration Calculation	44,314	44,314	258	44,213,223,258	44
Ass. Req. Duration	314	213	314	258	314
Listen Interval	10	10	10	3	5
Supported Rates	s3	s1	s3	s4	s1
Ext. Sup. Rates	e1	e1	e1	e3	e1
QoS Capability	F	F	F	T	F
Vendor Specific	v1	v1-v4	v1	v1	v1

Table 5. Results from Scenario 2 on tests 1-8 for S-5 and S-7

Fingerprinting Property	S-5	S-7
PS-Poll	F	F
Keep Alive	F	F
Null before Probe	T	T
Mode changing Null	F	T
Fixed Interval	F	F
Null Data Type	QoS7	Regular
Duration Calculation	258	44,223,258,314
Ass. Req. Duration	314	314
Listen Interval	10	10
Supported Rates	s3	s3
Ext. Sup. Rates	e1	e1
QoS Capability	F	F
Vendor Specific	v1	v1

Scenario 2. The eight distinctive features or fingerprints determined for each STA were identical to the ones determined in Scenario 1, with the exception of "Fixed Interval" for S-7. The fingerprints from this scenario are shown in Table 5.

Scenario 3. In both Scenario 2 and 3 the attacker’s Authentication Request frame was observed, making it easy to identify exactly when the attack occurred. The fingerprints from this scenario can be seen in Table 6.

Table 6. Results from Scenario 3 on tests 1-8 for S-5 and S-7

Fingerprinting Property	S-5	S-7
PS-Poll	F	F
Keep Alive	F	F
Null before Probe	—	T
Mode changing Null	—	—
Fixed Interval	—	—
Null Data Type	QoS7	Regular
Duration Calculation	258	314
Ass. Req. Duration	314	314
Listen Interval	10	10
Supported Rates	s3	s3
Ext. Sup. Rates	e1	e1
QoS Capability	F	F
Vendor Specific	v1	v1

6 Discussion and Conclusion

We have identified and experimented with some communication fingerprints of the IEEE 802.11 MAC layer that may serve to distinguish user stations. Our algorithms are able to detect spoofing attacks where the victim is not connected simultaneously with the attacker, something commercial IDS cannot do today. We have shown the feasibility of passively measuring the MAC layer fingerprints without specialized equipment, and that this can be done efficiently under realistic network access conditions. No precomputed database of fingerprints is necessary. The test data were acquired under realistic access scenario setups of 8-10 minutes, using 11 different devices. The communication fingerprints exhibited an average Hamming distance of 2.82. Even in the case of severely reduced communication data available (scenario 3), the tests show that the proposed algorithms are still able to distinguish between devices. The level of attacker skills required to avoid detection and evidence of session hijacking attacks in 802.11 can be raised considerably by using techniques presented here.

It remains to find out how to fine tune the selection of parameter values in the algorithms to gain optimal detection efficiency. Also, some of the fingerprints used are not easily altered as they are hard-coded in the firmware and drivers of the devices, while other fingerprints might be easier to alter or conceal, so assessing the difficulty of modifying or concealing a fingerprint are future work.

References

1. Holgernes, E.: Detecting Identity Thefts in Open 802.11e Enabled Wireless Networks. Masters thesis, Department of Telematics, NTNU, 109 pages (June 2010), <http://daim.idi.ntnu.no/masteroppgave?id=5476>

2. Idland, C.: Detecting MAC Spoofing Attacks in 802.11 Networks through Fingerprinting on the MAC Layer. Masters thesis, Department of Telematics, NTNU, 96 pages (June 2011), <http://daim.idi.ntnu.no/masteroppgave?id=6260>
3. Eian, M., Mjølunes, S.F.: The modeling and comparison of wireless network denial of service attacks. In: Proceedings of the 3rd ACM SOSP Workshop on Networking, Systems, and Applications on Mobile Handhelds. ACM (2011)
4. Franklin, J., McCoy, D., Tabriz, P., Neagoe, V., Van Randwyk, J., Sicker, D.: Passive Data Link Layer 802.11 Wireless Device Driver Fingerprinting. In: Proceedings of the 15th Conference on USENIX Security Symposium, vol. 15 (2006)
5. Gopinath, K.N., Bhagwat, P., Gopinath, K.: An empirical analysis of heterogeneity in IEEE 802.11 MAC protocol implementations and its implications. In: Proceedings of the 1st International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization (2006)
6. IEEE. IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems— Local and Metropolitan Area Networks— Specific Requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Technical Report. IEEE (2007)
7. Gu, W., Yang, Z., Que, C., Xuan, D., Jia, W.: On Security Vulnerabilities of Null Data Frames in IEEE 802.11 based WLANs. In: Proceedings of The 28th International Conference on Distributed Computing Systems, pp. 28–35. IEEE (2008)
8. fdXtended. HSMX - Internet Access Platform, Datasheet (June 9, 2011), <http://www.fdxextended.com/datasheets/HSMX-datasheet.pdf> (retrieved)

Appendix: The Algorithms

Input: *frame*

```

if frame == Beacon frame and bit in TIM is set then
    beacon_count++
    if beacon_count > listen_interval then
        use_PSPoll = false
        if use_PSPoll was true then
            return suspect attack
        end if
    end if
end if

if frame == PS-Poll frame then
    beacon_count = 0
    use_PSPoll = true
    if use_PSPoll was false then
        return suspect attack
    end if
end if

```

Algorithm 1. Test 1, PS-Poll

Input: *frame*

time_delta = timestamp previous frame - timestamp current frame

buffer is 0.15 sec

within_buffer = $9.85 < time_delta < 10.15$

if *within_buffer* **and** *frame* == Null Data **then**

use_keep_alive = **true**

if *use_keep_alive* was **false** **then**

return suspect attack

end if

else if *time_delta* > 10.15 **then**

use_keep_alive = **false**

if *use_keep_alive* was **true** **then**

return suspect attack

end if

end if

Algorithm 2. Test 2, Keep Alive

Input: *frame*

set_size = 5

min_limit = 0.80

if *frame* == Probe Request **then**

if previous frame was Null Data **then**

using_count++

total_count++

else if previous frame was **not** Probe Request **then**

total_count++

end if

end if

if *total_count* == *set_size* **then**

use_null_before_probe = *using_count* > *set_size* × *min_limit*

if *use_null_before_probe* changed value **then**

return suspect attack

end if

using_count = 0

total_count = 0

end if

Algorithm 3. Test 3, Null before Probe

Input: *frame*

set_size = 30

min_limit = 0.9

```

if frame == Beacon frame then
  if number of Beacons since data > listen_interval then
    pwr_mode = AM
  end if
end if

```

```

if chk_next_pkt == true then
  if frame type is data and fame ≠ Null Data then
    using_count++
    total_count++
  end if
  chk_next_pkt = false
else
  if pwr_mode == PS then
    if frame == Null Data and pwr_mgt bit == 0 then
      chk_next_pkt = true
    else if frame is DATA and pwr_mgt bit == 0 then
      total_count++
    end if
  end if
end if

```

pwr_mode = *pwr_mgt* bit (1 = PS, 0 = AM)

```

if total_count == set_size then
  use_mode_chng_null = using_count > set_size × min_limit
  if use_mode_chng_null changed value then
    return suspect attack
  end if
  using_count = 0
  total_count = 0
end if

```

Algorithm 4. Test 4, Mode changing Null Data

Input: *frame*

set_size = 50

threshold = 0.8

if *frame* == Null Data **then**

time_delta = time elapsed since previous Null Data frame

if *pwr_mgt_previous* == *PS* **and** *pwr_mgt_current* == *AM* **then**

if *average* exists **then**

if *time_delta* is within *range(average)* **then**

pair_ok_count++

end if

pair_total_count++

else

calculate *average* from first 10 pairs

end if

end if

if *pair_total_count* == *set_size* **then**

use_fixed_interval = (*pair_ok_count*/*set_size*) \geq *threshold*

if *use_fixed_interval* changed value **then**

return suspect attack

end if

pair_ok_count = 0

pair_total_count = 0

end if

end if

Algorithm 5. Test 5, Fixed Interval

Input: *frame*

if *frame* == Null Data **then**

categorize frame as QoS or regular

if *frame* was QoS type **then**

identify the priority class (TID)

end if

if *frame* differs from the previous Null Data frame **then**

return suspect attack

end if

end if

Algorithm 6. Test 6, Null Data Type

Input: *frame*

```

if frame == Null Data then
  get the duration value from the Duration/ID field
  get the data rate from the radiotap header

  compare duration for given data rate with previous value

  if differences in duration value for same data rate then
    return suspect attack
  end if
end if

```

Algorithm 7. Test 7, Duration Calculation

Input: *frame*

```

if frame == Association Request then
  record implicit identifiers
  if Ass. Req. for same MAC address is recorded before then
    compare implicit identifiers from current and previous frame
    if inconsistencies in implicit identifiers then
      return suspect attack
    end if
  end if
end if

```

Algorithm 8. Test 8, Association Request

Input: *frame*

```

run Test 1, PS-Poll
run Test 2, Keep Alive
run Test 3, Null before Probe
run Test 4, Mode changing Null Data
run Test 5, Fixed Interval
run Test 6, Null Data Type
run Test 7, Duration Calculation
run Test 8, Association Request

```

evaluate outputs from tests 1-8

return attack / no attack based on evaluation

Algorithm 9. The Fingerprinting Algorithm

BREDOLAB: Shopping in the Cybercrime Underworld

Daan de Graaf¹, Ahmed F. Shosha², and Pavel Gladyshev²

¹ National High Tech Crime Unit, Netherlands' Police Agency, The Netherlands
Daan.De.Graaf@nhtcu.nl

² University College Dublin, Ireland
Ahmed.Shosha@ucdconnect.ie, Pavel.Gladyshev@ucd.ie

Abstract. A recent emerging trend in the underground economy is malware dissemination as a service. Complex botnet infrastructures are developed to spread and install malware for third-party customers. In this research work, a botnet forensic investigation model is proposed to investigate and analyze large-scale botnets. The proposed investigation model is applied to a real-world law-enforcement investigation case that involves investigation of a large-scale malware dissemination botnet called BredoLab. The results of the forensic investigation show the effectiveness of the proposed model in assisting law-enforcement to conduct a successful forensic analysis of BredoLab botnet and its related resources.

Keywords: BredoLab, Botnets, Law-Enforcement Investigations, Malware Forensics, Forensic Investigation Models.

1 Introduction

Over the past few years, cybercrimes on the Internet have gradually transformed to profit-based crimes. A complex underground economy has emerged with complex divisions to manage various cybercriminal activities, e.g. financial crimes on the Internet, identity theft, attacking online services and dissemination of suspicious services, i.e. spam and phishing distribution. These illegal activities are supported with a solid networking infrastructure, such as bulletproof hosting through Virtual Private Network (VPN), to provide the cybercriminals a quality control and management for their malicious activities.

To manage the underground economy, well-defined organizations are established to rule the economic and technical aspects of the malicious service delivery through professional roles, such as, carders, scammers, financial cashiers, malware authors, spammers, spoof-website designers, money launders and botherders [1]. These organizations provide fee-based services on behalf of third-party customers to commit the customers' required criminal activities. In essence, these illegal services are mostly advertised on communication forums that are denoted as "Underground Forums" [2]. Such forums provide a secure communication channel between malicious services' providers and the services' customers through providing an infrastructure, e.g. communication-based dashboard to manage requested services and

to advertise newly developed malicious services. To commit previously aforementioned illegal acts, cybercriminals are usually assisted with botnets. A botnet is a collection of infected computers connected to the Internet and controlled by a botnet commander, usually denoted as bot-herder, and utilized to commit wide variety of cybercrimes, such as denial of Internet-based services [3].

As a bot controller, the bot-herder possesses the ability to download, update and execute malicious binaries on infected systems [3-4]. Fundamentally, he/she utilizes this functionality to update installed bots on the victim's computer with a newly developed malware sample, to allow execution of new binaries determined to commit different malicious activities. This process is denoted as "malware downloader" and describes the ability of a certain botnet to install other malware samples for different purposes. Since malware downloading is an important resource in botnets, bot-herders may offer the resource as a fee-based service to other cybercriminals. As a result, a cybercriminal use such paid service to commit a specific crime, i.e. malware downloader for banking fraud. A complex example of a botnet that was specifically used to offer the bots resources for spam activities and bank fraud is BredoLab.

Researchers and anti-virus companies first saw the BredoLab botnet in 2009. It is a complex downloading platform designed to facilitate malware spread on a massive, large-scale rate and used as fee-based service for installing malware to third-parties customers who could use infected machines (bots) to commit various cybercriminal activities.

From July 2010 till October 2010 the National High Tech Crime Unit of the Netherlands' Police Agency (NHTCU) did an investigation to a specific BredoLab botnet. The investigation has estimated that initial size of the botnet is, at least, three million infected machines. Following the investigation, NHTCU discovered that the networking infrastructure of this BredoLab botnet was running at a large-scale hosting provider in the Netherlands. Thus, on October 25, 2010 the NHTCU successfully took over the control of a BredoLab botnet and got access to servers that were directly connected to the network.

Because of the large-scale nature of BredoLab, traditional forensics investigation models were not sufficient to investigate and analyze the bot's resources. A forensics investigation model to investigate large-scale botnets was required.

In this research paper, a large-scale botnet's forensic investigation approach is proposed to analyze the botnet infrastructure. The proposed approach is applied in a real-world law-enforcement investigation of a BredoLab botnet. Finally, an analysis on BredoLab's resources is conducted to provide practical insights on the investigation of the malware selling botnets.

Paper Organization. Section two presents the proposed approach to forensically investigate large-scale botnets. Section three presents a case study of a law-enforcement investigation on the BredoLab botnet. Section four presents a research on the botnet data. Finally, section five concludes the paper.

2 An Approach to Analyze Large-Scale Centralized Botnets

Various forensic analysis and investigation approaches are proposed to analyze botnets, and investigate the underground economies. Most of currently proposed approaches, however, are limited to the analysis of malware samples found in acquired botnets, or focus on analysis of the communication channels to/from the cybercriminals. A comprehensive forensic investigation model to investigate large-scale infrastructure of botnets, however, is still missing.

In this section, we describe the infrastructure model commonly employed in centralized large-scale botnets, and propose a forensic investigation model to analyze the infrastructure. Note that, illustrated infrastructure is based on a real-world investigation case of a BredoLab botnet.

Generally, large-scale botnets are comprised of several working components, each of which is designated to one or more predefined functionalities. For example, main modules in a large-scale botnet such as BredoLab encompass C&C (Command and Control) servers, databases servers, bot-herder administration panels and customer control panels. The understanding of each module and the interrelationship between instances of each module is crucial for forensic investigation of the botnet's activities and is required for the proper disassembly of the botnet threats.

From a forensic investigation point of view, successful investigative model has to consider the modular nature of a botnet under investigation. Otherwise, investigation of interactions between the botnet's modules may conclude to insufficient results. As a result, the proposed botnet investigation model is decomposed into a set of subcomponents; each component is designated to investigate one or more specific module in the botnet's infrastructure. Note that, proposed model can be used when the botnet is identified and its servers are located.

In the proposed approach, three main investigation stages are defined to analyze the botnet's resources. The first stage includes forensic acquisition of the botnet's resources, i.e. forensic imaging of botnet hosting servers, etc. The second stage includes forensic evidence and data extraction of acquired forensic images and communication networks. Finally, the third stage includes analysis of malware samples found in botnet resources. A detailed analysis of the botnet backend and the re-building of the botnet infrastructure in a controlled environment to reconstruct a full view of the botnet resources is, also, proposed. This allows precise understating of the botnet threats.

As shown in figure 1, wiretap and net-flow components are designed to allow the acquisition of sampled network data from the botnet's communication. The analysis of botnet network data is essential to identify how the botnet modules are interacting and communicating with each other, which facilitates forensic investigation of interrelated modules. Forensic acquisition of the botnet's infrastructure, when the botnet is taken down, is accomplished using the forensic images component. This process includes forensic analysis of the images to extract forensic evidences and information to continue investigation, such as, botnet administrations and customers panels' related information, information about C&C servers, bots' database and downloader malware module.

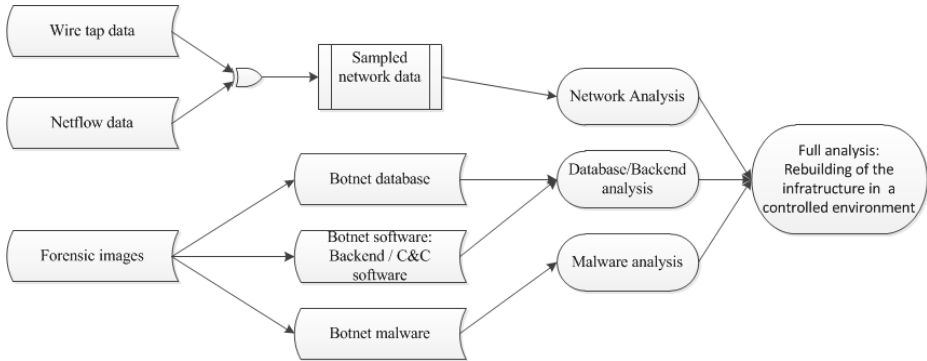


Fig. 1. Large-Scale Botnet Forensic Investigation Model

Based on extracted evidence from acquired forensic images a detailed behavior analysis of botnets resources, such as, malware samples behavior is required.

Determining the behavior of malware samples extracted from the botnet downloader database is accomplished through a dynamic malware analysis component. Practically, malware analysis component is a preconfigured controlled dynamic malware analysis environment [5-6] that is used to determine the behavior of malware samples used in botnets. The behavior analysis of malware samples includes identification of actions and activities invoked by the sample in infected bots, security assessment of used exploitation payload and spreading mechanism used by malware sample, and method used to hide their presence in infected bots, and techniques used to ensure persistence. Finally, to ensure that valid forensic investigation conclusions are resulted, a full forensic analysis based on re-building the botnet's resources in a controlled environment is developed. The workflow of service provided by the botnet is traced in the managed environment and are matched to conclusions resulting from the investigation to ensure validity and integrity of the conclusions.

3 Case Study: BredoLab Botnet Investigation

The first BredoLab exploits were seen in May 2009 by anti-virus companies [7-8]. The initial analysis of BredoLab reveals a complex threat posed by the botnet as results of employing different sophisticated attack vectors implemented in various malware samples. Particularly, BredoLab can install a wide range of malware families, e.g. password stealers, rootkits, backdoors, banking trojans, fake anti-virus software and spam malware [8].

In this section, a case study is provided about law-enforcement investigation of a BredoLab botnet. The case study includes procedures used to disassemble the botnet's threat and detailed forensic analysis on data extracted from the botnet for forensic investigation purposes based on the previously presented forensic investigation model.

3.1 NHTCU Investigation

Leaseweb, a large hosting provider located in the Netherlands, started in 2010 with the Community Outreach Project [9]. This project offers free servers and bandwidth in support to the organizations that monitor, identify and combat the sources of spam and crime on the Internet. One of the participants in the project is Abuse.ch, a non-profit organization that analyses different threats on the Internet, such as, Zeus and SpyEye [10]. Through Abuse.ch, Leaseweb was informed about a possible large-scale botnet infrastructure that intersects with their network, since Zeus malware was one of the malware samples spread by BredoLab. Normally, the regular security process of Leaseweb encompasses the immediate interaction with the servers' disseminating malware and blocking all communication channels. Instead, due to the large-scale nature of discovered botnet, Leaseweb decided to involve the National High Tech Crime Unit of the Netherlands' Police Agency (NHTCU) for further tracking of the botnet's resources. Initially, NHTCU acquired the net-flow data from the servers at Leaseweb for further network forensic analysis and investigation. Furthermore, the NHTCU placed additional wiretaps on eleven servers at Leaseweb to control the network communication involving BredoLab bots with a total amount of acquired wiretap data equals to four terabyte.

During the investigation, different malicious servers were fully identified as followed:

- A malware management server that used to hack and distribute newly developed malware samples.
- FTP grabber server used to authenticate credentials used by malware and distributed by the BredoLab botnet.
- A VPN server used for different purposes, such as, management of other servers, hacks to new proxy servers, Denial of Services (DDoS) attacks and communication with partners and personal customers.
- A database server used to store information about infected bots and malware samples distributed by the BredoLab botnet.
- A Jabber server to communicate with various malware samples i.e. commands to Zeus malware.
- Various C&C servers to control the bots.

In most adversarial legal systems, to establish a valid accusation, forensic analysts are required to prove that accused person has a knowledge and control - or what so-called "*mens rea*"¹- based on evidence and artifacts extracted from case under investigation. In the BredoLab investigation, forensic evidence that establish the knowledge and control of the criminal activity using BredoLab is obtained from the wiretaps of the database server and the VPN server. Various forensic evidence and supporting artifacts, such as, evidence to identify the botnet owners and customers, traces of

¹ *Mens rea* is Latin for "guilty mind". In criminal law, it is viewed as one of the necessary elements of a crime.

malware distribution for different purposes, evidence about launching DDoS attacks and hacking into various websites, are successfully collected during investigation based on wiretapping investigated servers and the analyses of the network traffic to/from identified servers.

3.2 BredoLab Botnet Termination Procedures

On October 25, 2010, BredoLab infrastructure was terminated and taken offline. The NHTCU successfully connected to the backend panel located on one of the C&C servers through exploiting a cookie extracted from wiretap artifacts of the VPN server. After controlling the backend panel, the NHTCU terminated all malicious activity, i.e. active malware distributions tasks. Intuitively, this action will only contribute to limit infecting new computers; however, to disrupt the ability of BredoLab, further actions are required to disinfect previously infected bots. Thus, NHTCU developed a program that is uploaded to all bots in the network and launched a standard browser on the victims' computers to allow infected users to read a press-warning message. This warning message has been viewed over 300,000 times.

BredoLab botnet was let active for a few days in order to reach as much victims as possible. After that, the network connections to all servers were terminated. Suspect servers were confiscated for further forensic investigation by the NHTCU. Additionally, suspect IP's, domain names and malware found during the investigation were distributed to the professional security communities.

During the investigation, the bot-herder was identified as results of wiretap data analysis of the VPN server, since the suspect bot-herder used to use the VPN server for other non-criminal activities, such as: accessing his personal Facebook account, e-mail accounts, and his WebMoney accounts. The NHTCU made an international arrest warrant, through which the suspect got arrested a few days later at the airport of Yerevan, Armenia. The suspect got convicted in Armenia for four years, based on the information provided by the NHTCU.

4 Analysis of BredoLab's Resources and Infrastructure

BredoLab resources' analysis and forensic investigation is accomplished using the forensic investigation model described earlier. Thus, in this section, the analysis's results of the different BredoLab resources, such as, the network data acquired from the communication infrastructure and the layout of the network used in a BredoLab botnet is presented. To reconstruct the network layout, a sampling of the wiretap data was captured and interrelation between captured network packets was reconstructed using a custom-built wiretap analysis tools developed by the NHTCU [11]. Each sampled data packet consists of: source and destination IP addresses, used networking protocol, source and destination port numbers. The sampled packets are correlated together to determine the communication network layout used to maintain BredoLab services. As shown in figure 2, when sampled network data are correlated together, the BredoLab network infrastructure layout can be identified.

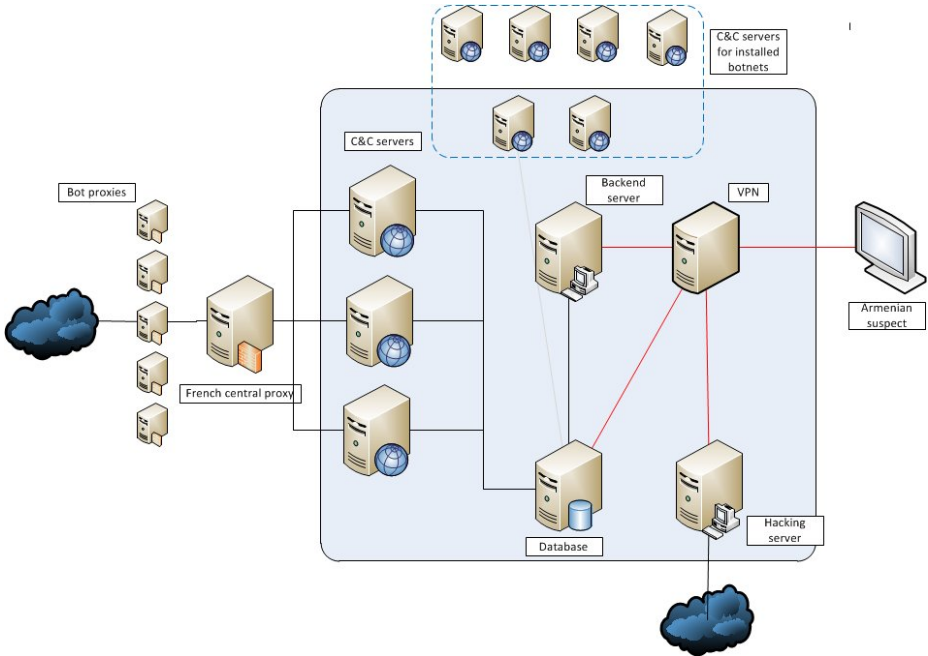


Fig. 2. The BredoLab Infrastructure Layout

The BredoLab infrastructure consisted of: a database server, a central proxy server, several proxies installed on bots, a backend server, a personal hacking server, a VPN server and several C&C servers. The blue square contains the servers that were wiretapped at Leaseweb. The red lines are traffic generated by the administrator from the VPN server. The data created by the previously mentioned wiretap analysis tools showed that all the traffic from the three C&C servers consists of: HTTP traffic going to a main proxy server in France, and plain MySQL code going to one central database server. The database server was used, as well, by one of the installed malware as FTPgrabber malware. Forensic investigation also showed that the VPN server has several encrypted connections to a suspect IP addresses in Armenia and was used to manage other servers.

4.1 Wiretap Data Analysis

Based on determined BredoLab's communication network infrastructure layout, three main C&C servers were identified as active servers. Forensic investigation of these servers' wiretaps revealed communication traffic to a domain called "worldhostdns.com". Further investigations resulted in specified domain is resolve to an IP address of a C&C server through a proxy server that is located in France. Analysis of wiretap data has, additionally, revealed that communication between infected bots and suspect domain is accomplished through various HTTP requests to a web page named "controller.php".

```

192.168.1.101 - - [06/Sep/2011:17:17:16 +0200]
"GEThttp://worldhostdns.com/new/controller.php?action=bot&entity_list=1272705710,1272796684,127875990,1281608998,1283317892&first=0&rnd=981633&uid=1&guid=2947510467 HTTP/1.0" - - "-" "-"

```

```

192.168.1.101 -- [06/Sep/2011:21:29:32 +0200] " GET
http://worldhostdns.com/new/controller.php?action=report&uid=1&guid=3985971469&rnd=123&entity=1259351490:unique_start;1259970379:unique_start;1271368047:unique_start;1278753990:unique_start;1283419228:unique_start;1283685805:unique_start HTTP/1.0" - - "-"
"-"

```

Fig. 3. A sample Communication Packet to a C&C Server

In essence, the communication to the C&C server through “controller.php”, as shown in figure 3, is defined in the following steps [8]:

- The infected bots connect to the suspect C&C server to update its infection status and to download newly developed malware through a GET request. Whereas, the C&C server could identify if connected host is an authorized bot, if the “Action” parameter in GET request is set “Bot”.
- The C&C server responds to communicate bots with a response message containing malware samples needed to install.
- The bot replies back if a malware installation task was handled successfully or not through specific GET request in which “Action” parameter is set to “Report”. Additionally, infected bots alert the server with the task starting timestamp via “Unique_Start” parameter.
- Finally, the C&C server reports back with an acknowledgement message.

4.2 Backend Panels Forensic Analysis

Forensic investigation of BredoLab’s network has identified two different backend panels; one is located on a separate backend server, and the other is located on the database server. The backend panel is called “BM Tx Edition v1.5.1.” such that, BM is abbreviation for **BredoLab Manager**. The Tx and the version numbering is a reference to installed BredoLab’s software version. Identified panels are developed in PHP and are used to manage different tasks, like: activate malware orders, start/stop of malware tasks, and manage installing and cleaning of malware. Besides management activities, the backend panels also maintain statistics about installed malware and infected bots. Below is a snapshot sample for information presented in a backend panel.

BM Tx Edition v1.5.1 Время сервера 08 09 10 11 47 25
 Статистика | Файлы | Пользователи file: Выход

Файлы
Добавить файл

Имя	Лимит загрузки	Способ загрузки	Пауза	Повтор	Только чтение	Фильтры разрешенные запрещенные	Размер URL	Дата изменения	Функции
<input checked="" type="checkbox"/> Scabbler	нет	RAM	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Регионы: нет нет нет Страны: нет нет нет	265728 байт	08 09 10 11 59 14	
<input checked="" type="checkbox"/> Click	нет	HDD	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Регионы: нет нет нет Страны: US CA RU UA нет нет	74792 байт http://d0w0nloa.d0w0nloa.net/81... http://d0w0nloa.d0w0nloa.net/81...	03 09 10 11 54 15	
<input checked="" type="checkbox"/> Ppgrabber	нет	RAM	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Регионы: нет нет нет Страны: нет нет нет	32788 байт	08 09 10 12 00 14	
<input checked="" type="checkbox"/> Cosma	нет	RAM	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Регионы: нет нет нет Страны: нет PT RU GB ES NL BE CA UA CH PL NZ IE нет нет	7158 байт	02 09 10 12 22 34	
<input checked="" type="checkbox"/> Ily	нет	RAM	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Регионы: нет нет нет Страны: нет нет нет	8192 байт	30 08 10 14 37 55	
<input checked="" type="checkbox"/> ORDERS - abc - US	нет	HDD	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Регионы: нет нет нет Страны: US нет нет	113664 байт	30 08 10 13 36 00	
<input checked="" type="checkbox"/> ORDERS - brusa@abber.cz	нет	HDD	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Регионы: нет нет нет Страны: ES нет нет	47104 байт	30 08 10 12 59 01	
<input checked="" type="checkbox"/> Fraer	нет	HDD	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Регионы: нет нет нет нет Страны: PT RU GB ES NL BE CA UA CH PL NZ IE нет нет	148480 байт	02 09 10 17 49 22	
<input checked="" type="checkbox"/> ORDERS - feedfood	нет	HDD	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Регионы: нет нет нет Страны: GB нет нет	116224 байт	08 09 10 15 28 01	

Легенда:
 - активный элемент.
 - скрыт по умолчанию.
 - если нажатие будет удалена вся статистика выбранного файла.
 - удалить файл и все его статистику на сервере.

Наверх ↑

Fig. 4. BredoLab Sample Backend Panel

An additional important panel is BredoLab’s customers and partners management panel. Customers’ panel which is resident in the database server was used to upload malware samples and to define preferred number of computers to be infected. Once a malware is uploaded through the panel, malware spread task is activated on the main backend panel and then malware is being disseminated. Finally, the panel, as well, keeps tracks of open payments and customer related information.

4.3 BredoLab Database Analysis

During post-mortem forensic investigation of the acquired servers’ forensic images, a BredoLab malware database was successfully extracted. Database forensic investigation has showed that a database named “BM” is used to store information about infected bots and different malware samples. Fundamentally, a number of essential database tables to assist BredoLab service delivery are identified as follow:

- Malware tasks: A table referencing stored information about infected bots and stored malware to manage outstanding malware infections tasks.
- Bots: A table to store the infected bots information.
- Users: A table that stores information about users and customers who are controlling the botnet.
- An administration table that is used by the backend panel and various C&C servers to assist the BredoLab administrator to control service requests, and payments from customers.

Tasks related to malware activities in the database are divided into two components; a component that is designated to malware management while the other is for statistics operations. The management component assists in 1) Upload new malware and place

it directly in the database as BLOB file, 2) Set country and regional variables to the location in which malware should be deployed, 3) Pause, load or reload spreading of a malware. While the statistics component, on the other side, is used to draw malware dissemination graphs, illustrate spread rates and spread dates in the backend panel. Every bot that is infected with a certain malware task gets stored in the statistics table. And for every infection, the backend panel kept track of IP address of the infected machine, date and timestamp of infection, and country and region.

The acquired statistics table contained information about 3,283,644 infected bots with 38 different malware tasks. However, the table counts, only, unique IP addresses. Hence, total infected IP addresses were 477,282 with the 38 different malware tasks. Note that, this number does not define the actual size of the BredoLab botnet, since it is possible that bots do not have active tasks at the database acquisition time.

Additionally, the wiretap from law-enforcement investigation of a C&C server showed that in a month, over one million infected unique IP addresses were identified. Thus, aforementioned statistics present a challenge in determining the actual size of a BredoLab botnet. Two tables in the investigated database are linked directly to the bots and loaders of BredoLab. These tables are used to assist monitoring of active infections of BredoLab via tracking information, such as: timestamps of infection and date of last connection to a C&C server, IP address, country, region and the infected bot GUID. The GUID resembles the serial number of the hard-drive where BredoLab resides. An administration table called “admtasks” is, also, identified and is used by the main backend panel and in the administration panel. The “admtasks” table is linked to the earlier mentioned malware tasks. “Admtasks” is a principal table in administrating active malware tasks and tracking customer information to every task. This includes, customer name, ICQ number of the customer, if found, and all payment details.

A number of 331 malware files dated from July 2009 till October 2010 are extracted from the “admtasks” table. Extracted malware files were tested using VTest software in cooperation with Norman IT Security [12]. VTest was recognizing 96% of extracted malware in the database.

Below, are a few examples of the malware samples being used in BredoLab:

- Eighteen malware files were recognized by as Tedroo. Tedroo is known as a malware that is being used to spread spam.
- Nine malware files were being recognized as Zeus or ZBot malware. Zeus is a well-known malware family that is being used for banking cybercrimes.
- More than twenty malware files were recognized as being a fake Anti-Virus program. Fake A/V are used to mislead users into paying money for fake Anti-Virus products.
- Eight files were recognized as BredoLab malware or BredoLab variants.

5 Conclusion and Discussion

Downloader botnets pose a significant challenge to the user and computer security on Internet. A prevalent example of downloader botnet is BredoLab. Although, a

significant effort to contain BredoLab's threats by security community and law-enforcement is spent, recent researches showed that BredoLab and its variants are still widely spread. Moreover, complex infection and spreading techniques that are found only in BredoLab malware samples are, unfortunately, now employed in other major malware families [13-14].

To this end, this practical research paper presented a large-scale forensic investigation model that is applied to BredoLab botnet investigation. The proposed model suggested different forensic investigation components to analyze the botnets resources and to extract necessary evidence to assist law enforcement. Furthermore, the paper illustrated the law enforcement procedures to forensically acquire and investigate BredoLab, and to develop required knowledge and control to support prosecution using proposed model. Most take-downs take months or even years of research to attempt a take-down [14,15]. In the case of the Bredolab investigation a wiretap on the VPN server from the main suspect proved to be enough to get the data needed to take-down the botnet and start prosecution. A combination of research in the security community together with law-enforcement might prove the best way to attack these kinds of botnets in the future.

Our future work includes, enhancement to the presented model and integration of security-related research with law-enforcement procedures to provide better countermeasures to cybercrime threats.

Acknowledgement. This work is a result of support provided by the National High Tech Crime Unit from the National Crime Squad part of the Netherlands' Police Agency.

References

1. Schiller, C., Binkley, J., Harley, D., Evron, G., Bradley, T., Willems, C.: Botnets, the killer web app., pp. 77–85. Syngress Publishing, Canada (2007)
2. Yip, M.: The Underground Economy Ecosystem (2011), <http://www.michaelyip.me.uk/blog/2011/08/the-underground-economy-ecosystem/>
3. Ianelli, N., Hackworth, A.: Botnets as a Vehicle for Online Crime. In: First International Conference on Forensic Computer Science. Carnegie Mellon University, Pittsburgh (2005)
4. Stone-Gross, B., Holtz, T., Stringhini, G., Vigna, G.: The Underground Economy of Spam: A botmaster's perspective of coordinating large-scale spam campaigns. In: 4th USENIX Workshop on Large-Scale Exploits and Emergent Threats. University of California, Santa Barbara (2011)
5. Ligh, M.H., Adair, S., Hartstein, B., Richard, M.: Malware Analyst's Cookbook and DVD, pp. 283–330. Wiley Publishing Inc., Canada (2011)
6. Ligh, M.H., Adair, S., Hartstein, B., Richard, M.: Malware Analyst's Cookbook and DVD, pp. 211–224. Wiley Publishing Inc., Canada (2011)
7. Sancho, S.: You Scratch My Back... Bredolab's Sudden Rise in Prominence. Trend Micro Inc. (2009)
8. Tenebro, G.: The Bredolab Files. Symantec Corporation (2009)

9. Leaseweb, <http://blog.leaseweb.com/2010/08/31/leaseweb-offers-free-web-hosting-to-fight-cybercrime/>
10. Abuse.ch The Swiss Security Blog, <http://www.abuse.ch>
11. National High Tech Crime Unit.: Replay Analyst Toolkit. KLPD, Driebergen (2011)
12. Norman ASA Norway, <http://www.norman.com>
13. February 2011 Intelligence Report, Bredolab, Zeus and SpyEye stage synchronized, integrated attacks. Symantec Corporation (2011)
14. Stone-Gross, B., Cova, M., Cavallaro, L., Gilbert, B., Szydowski, M., Kemmerer, R., Kruegel, C., Vigna, G.: Your Botnet is My Botnet: Analysis of a Botnet Takeover. In: 16th ACM conference on Computer and communications security, pp. 635–647. University of California, Santa Barbara (2009)
15. Dittrich, D.: So You Want to Take Over a Botnet... In: 5th USENIX Workshop on Large-Scale Exploits and Emergent Threats. University of Washington, Seattle (2011)

A Review and Comparative Study of Digital Forensic Investigation Models

Kwaku Kyei, Pavol Zavorsky, Dale Lindskog, and Ron Ruhl

Information Systems Security Department
Concordia University College of Alberta, Edmonton T5B 4E4, Canada
kwaku.kyei.f@gmail.com,
{pavol.zavorsky, dale.lindskog, ron.ruhl}@concordia.ab.ca

Abstract. In this paper we present a review and comparative study of existing digital forensic investigation models and propose an enhanced model based on Systematic Digital Forensic Investigation Model. One significant drawback in digital forensic investigation is that they often do not place enough emphasis on potential admissibility of gathered evidence. Digital forensic investigation must adhere to the standard of evidence and its admissibility for successful prosecution. Therefore, the techno-legal nature of this proposed model coupled with the incorporation of best practices of existing models makes it unique. The model is not a waterfall model, but iterative in nature helping in successful investigation and prosecution. The result of the study is expected to improve the whole investigation process including possible litigation.

Keywords: forensic investigation process, digital evidence, information sharing.

1 Introduction

Forensic computing and cybercrime investigation emerged as a result of increase in computer or digital crime due to the development of the Internet and proliferation of computer technology. The advancement in technology and the rise in online communication have not only brought about increase in criminal activity (with the use of the computer either a tool or target or both in committing crime) but also poses a challenge to law enforcement agencies on how to investigate these complex and sophisticated crimes. Various investigation models have been developed since 1984 (when the FBI laboratory and other law enforcement agencies began to develop programs to examine computer evidence). Some of these are for incident response and others are for court admissibility, but all were developed in an attempt to investigate and where necessary prosecute offenders. Unfortunately, not much has been achieved since the success rate for the prosecution is less than two percent [1].

The methods and procedural rules governing evidence gathering and investigation in these models vary from place to place. Since cybercrime is often transnational and borderless in nature offenders take advantage of these gaps to avoid arrest and prosecution [2]. Digital forensics is relatively new compared to other forensic

disciplines, and therefore there is no common standard of investigation. Each organization and country tend to adopt its own procedures, some focused on the technology aspect, and relegate legalities to the background [3], some focused on the data analysis portion of the investigation or other aspect of the process.

This paper presents a comparative study of the recent Systematic Digital Forensic Investigation Model [4] and other existing models based on the frame of reference (number of phases and activities in the existing models) and try to enhance it by filling in the gaps and omissions identified to make it more comprehensive and suitable for both investigation and prosecution.

2 Review of Previous Models

A number of digital forensic models have been developed for investigations since 1984; some of these focused on either incident response or investigation or emphasize a particular phase or activity of an investigation. Below are brief descriptions of the model development process from 2001 to 2012, see also Fig.1 – Fig.3.

A. *Digital Forensic Investigation Model 2001*

Kruse & Heiser (2001) came up with a model [5] which has three phases, namely acquiring evidence, authenticating the evidence and analyzing the evidence, popularly referred to as the three A's of digital forensics. This model is concerned with integrity of the evidence, and was designed for incident response.

B. *Digital Forensic Research Workshop 2001*

The DFRW model [6] is a collective document created at a Research Workshop organized in Utica USA in 2001. The model was made up of seven phases, namely Identification, Preservation, Collection, Examination, Analysis, Presentation and Decision. One significant feature of the model was that it was an improvement over previous models because it covered some of the stages others did not cover, such as the presentation stage. It also laid the foundation for digital forensic investigation and a framework for future research.

C. *Abstract Digital Forensic Model 2002*

Reith, Carr and Gunsch reviewed the DFRW and improved it by adding three more components, which were missing in the previous models. This model [7] was the most comprehensive of the three because it had all the activities of DFIM and DFRW and also added Preparation, Approach Strategy and Return of Evidence. Figure 1 shows the mapping of common elements in the three models and the additions are highlighted in ADFM.

D. *Integrated Digital Investigation Model 2004*

The Integrated Digital Investigation Model (IDIP) [8] has five phases, namely Readiness (Operational and infrastructural readiness), Deployment (Detection and

notification; and confirmation and authorization), Physical Crime Scene Investigation, Digital Crime Scene Investigation and Review. The model applied the normal traditional investigation approach and integrated it into digital forensic investigation. This was quite innovative, especially the reconstruction procedure in both physical and digital crime scene, which is a strategy used to detect cyber criminals [9].

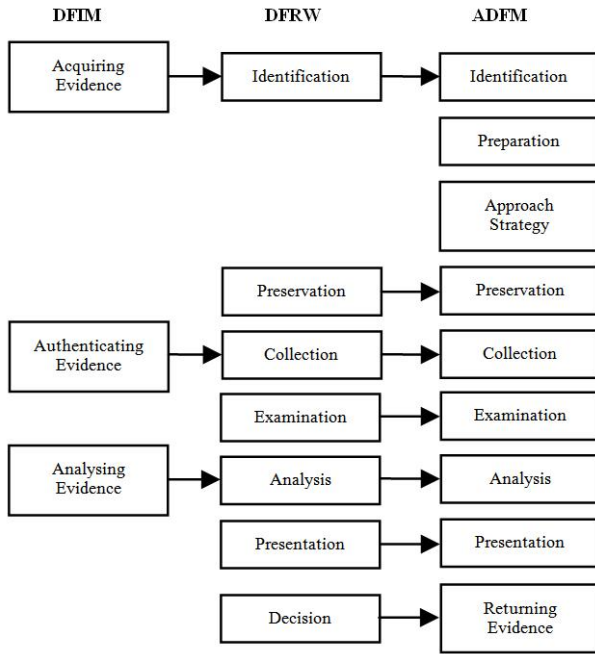


Fig. 1. The digital forensic investigation phases in the DFIM, DFRW and ADFM models

E. *Enhanced Digital Investigation Process Model 2004*

The Enhanced Digital Investigation Process Model (EDIP) [10] seeks to enhance integrated digital investigation process model by adding two additional steps: Trace back and Dynamite. Figure 2 shows mapping of common elements between the two models. Deployment phase in EDIP has physical and digital crime scenes, which are separate phases in IDIP and in addition introduced other useful activities like Detection & Notification, Confirmation and Submission. Trace back and Dynamite (reconstruction) would enable the investigator to trace the primary crime scene, from the footprint obtained from the secondary crime scene with the sole objective of identifying the possible suspect or criminal, which was a weakness in the earlier model.

F. *Extended Model of Cybercrime Investigation 2004*

The EMCI model [11] was developed by Seamus O Ciardhuain, who has considerable experience not only in cybercrime investigation but also as a researcher, network

administrator and developer of training for investigators in forensic computing. It is made up of thirteen (13) steps, namely Awareness, Authorization, Planning, Notification, Search for and identify evidence, Collection of evidence, Transport of evidence, Storage of evidence, Examination of evidence, Hypothesis, Presentation of hypothesis, Proof/Defense of hypothesis and Archive Storage (used for dissemination of information). The model provides a better understanding of the investigation process and captures most of the information flow for cybercrime investigation.

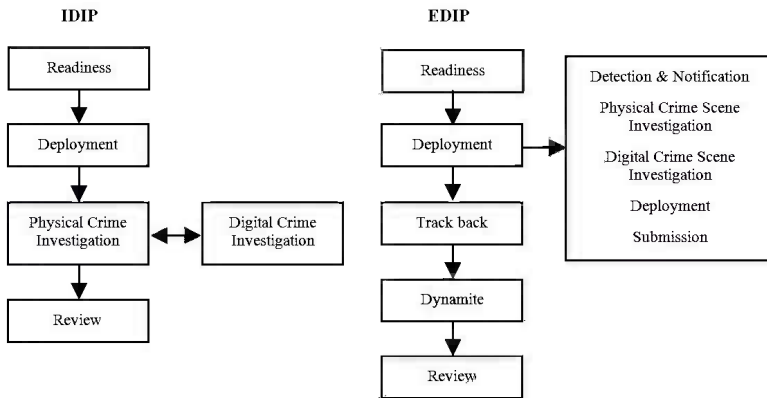


Fig. 2. Digital forensic investigation phases in IDIP and EDIP models

G. *Digital Forensic Model Based on Malaysian Investigation Process 2009*

In 2009, S. Perumal developed investigation model [12] based on cybercrime laws in Malaysia. The model consists of seven phases namely Planning, Identification, Reconnaissance, Transport & Storage, Analysis, Proof & Defense, and Archive Storage. It enhanced existing models by incorporating a live and static data acquisition process that focuses on volatile data. It also introduced data mining in the archive storage.

H. *Digital Forensic Model for Digital Forensic Investigation 2011*

Inikpi developed another model, (DFMDFI) [13] which was generalized into a 4-tier iterative approach. The first tier was made up of preparation, identification, authorization and communication. The second tier consisted of rules such as collection, preservation and documentation. The third tier was made up of rules like examination, exploratory testing and analysis and the fourth tier has result, review and report. What is significant about this model is that it is iterative, therefore one can revisit any activity or phase when it becomes necessary.

I. *The Systematic Digital Forensic Investigation Model 2011*

Agawal et al. (2011) [4] developed another model, the SDFIM, that organizes the digital forensic investigation process into eleven phases as outlined in Fig. 3.

Phase 1: Preparation

The preparation phase includes getting the initial understanding of the problem through assessment, and the right equipment. This phase is used to obtain authorization and approval, search warrant, and legal notice must also be given to those concerns and finally appropriate strategy should be developed.

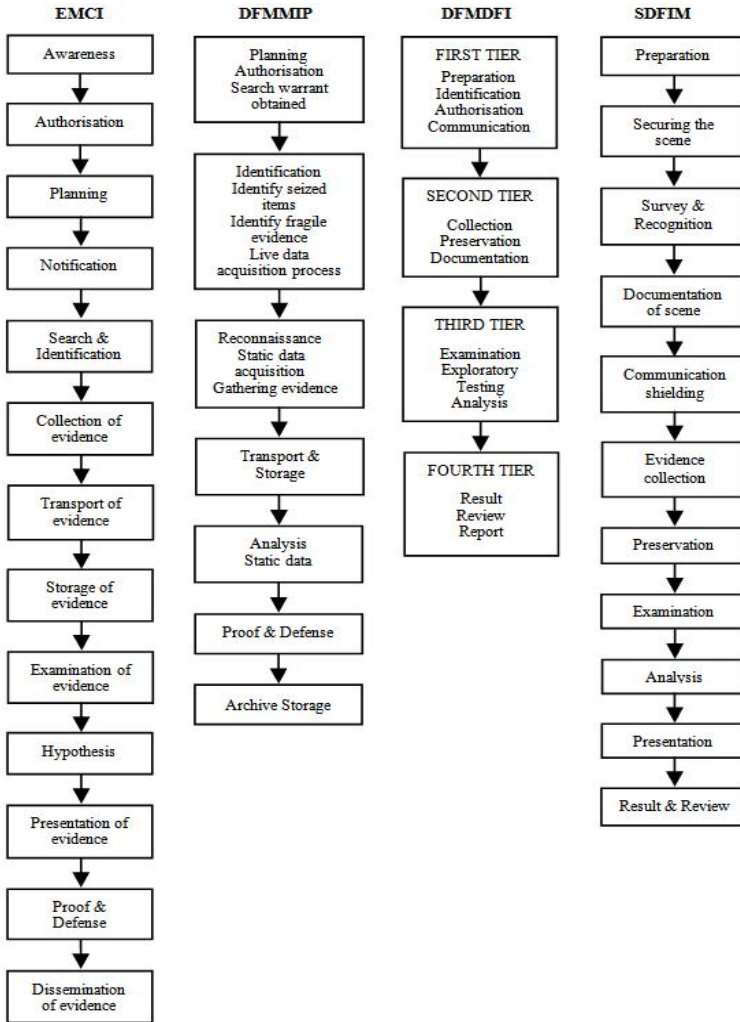


Fig. 3. Digital forensic investigation phases in EMCI, DFMMIP, DFMDFI and SDFIM models

Phase 2: Securing the Scene

The second phase primarily deals with securing the crime scene from unauthorized access and preserving the evidence from being contaminated.

Phase 3: Survey and Recognition

Survey and Recognition Phase involves an initial survey conducted by the investigators for evaluating the crime scene, identifying potential sources of evidence and formulating an appropriate search plan.

Phase 4: Documenting the Scene

Phase four involves proper documentation of both physical and digital crime scenes along with photographing, sketching, and crime-scene mapping.

Phase 5: Communication Shielding

Communication Shielding occurs prior to evidence collection. At this stage, all further possible communication options of the devices should be blocked. Even if the device appears to be in an off state, some communication features like wireless or Bluetooth may be enabled. This may result in overwriting the existing information and hence such possibilities should be avoided.

Phase 6: Evidence Collection

The evidence includes both volatile and non-volatile. The necessary precautionary measures must be taken to ensure its integrity.

Phase 7: Preservation

Preservation includes copies of digital evidence, packaging, transportation, and storage. Appropriate procedure and environmental conditions to maintain the chain of custody should be followed and documented to ensure the electronic evidence collected is not altered or destroyed.

Phase 8: Examination

Examination involves examining the content of the collected evidence by a forensic specialist and extracting information for presentation in court. This is made up of volatile and non-volatile evidence. According to the author, hashing technique like md5 must be used to authenticate the data.

Phase 9: Analysis

Analysis is more of technical review conducted by the investigative team on the basis of the result of the examination of the digital evidence and reconstructing the event data based on the guidelines recommended by the National Institute of Justice.

Phase 10: Presentation

Presentation phase is where a report consisting of detailed summary of the various steps taken during the investigation and the conclusion arrived at is presented to the appropriate authorities. It is presented to the court of law when a crime is committed or corporate management when it is an incident

Phase 11: Result and Review

At the final stage of the investigation, an evaluation is made and the result is used to update or improve any shortcoming experienced during the investigation.

Agawal et al (2011) performed a comparative analysis of some selected models and came out with a model that is probably one of the most detailed to date. The advantages of his model over others are listed in the following section.

3 Advantages and Limitations of the SDFIM

The model is not only comprehensive in scope because it captured almost all the important activities of the existing models but it is also based on forensic laws and the guidelines recommended by National Institute of Justice.

The model addresses the issue of collecting digital evidence from either volatile data or live response or both, which others with the exception of DFMMIP did not. This is a major concern for cybercrime investigation and equally important ingredient for prosecution.

In spite of these advantages, the model has the following limitations. For example it focused on the technical aspect of the investigation, (examination and analysis). However, all other aspects of the process both pre and post investigation processes must be considered equally if a comprehensive and detailed model is to be achieved.

The model revealed some similarities in some of the phases which could be regrouped to make it more coherent. For example, Survey and Recognition could be part of Preparation, Documenting the Crime Scene and Communication Shielding could also be part of Securing the Crime Scene, since these two independent phases in this model in reality are part of Securing Crime Scene. Examination and Analysis could also be combined. The model used these terms as separate activities but their definitions are not only similar but also complement each other and it can create confusion when separated.

SDFIM did not cover all aspects of cybercrime investigation as shown in Table 2 but mainly focuses on the process of obtaining digital evidence. According to Computer Crime Research Centre, [14] cybercrime is defined as crimes committed on the Internet using the computer as either a tool or a targeted victim. To effectively investigate such a crime, especially in a network environment which is a borderless or distributed system, one needs to trace the footprint from the secondary crime scene to determine the primary crime scene. [15] [9]. This was completely missing.

Even though the model is designed to investigate cyber-crime, in reality it can only be useful for computer crime (computer fraud) on a standalone machine where the computer is used as repository of evidence but not as a tool or target or both, due to the absence of Trace back and Dynamite [10] as explained earlier. Therefore, it cannot be applicable to a distributed system or complex architectures or network.

4 Gap Analysis Based Enhanced Digital Forensic Investigation Model

The weaknesses and limitations of the existing models are shown in Table 2. It is evident that the existing models did not address all the concerns or capture all the

activities necessary for investigating and prosecuting cybercrime from start to finish. Most of them focus too much on processing digital evidence or the investigation process at the expense of other steps. The motivation for an enhanced model is based on the fact that digital forensics and for that matter cybercrime investigation involve not just a single computer but multiple or distributed computers, and successful investigation of such crime requires access to evidence from various sources. However, the existing forensic models including the SDFIM, do not sufficiently take into consideration these various sources of evidence and the need to correlate them both for the purpose of reconstruction and prosecution.

The proposed model is made up of six phases and is depicted in the flow chart in Fig. 4. It fills in the relevant gaps that were omitted from the existing models (as indicated in column II of Table 2) and also introduces Information Sharing shown in Table 3, which is an important ingredient for effective investigation and prosecution.

One unique feature about the proposed model which is an improvement over existing models is that, it has all the advantages of the existing models but in addition addresses the limitations of SDFIM. For example, SDFIM has eleven phases some of which overlapped, as explained in the previous section. In the proposed model, the phases have been regrouped as shown in Table 4 for efficiency and consistency.

The inclusion of honeypots/honeynet, intrusion detection and prevention systems and like tools supporting traceability and reconstruction for ongoing investigation will enable the security investigators to trace the primary crime scene from the footprint obtained from the secondary crime scene with the sole objective of identifying the possible suspect(s) or criminal(s) in a distributed or borderless environment.

Technicalities alone as mentioned in the previous paragraph is not sufficient for successful investigation and prosecution unless is backed by forensic laws, cooperation and collaboration from law enforcement agencies from both the primary and secondary crime scenes. This is achieved through information sharing and criminal profiling which are very significant for they equipped the law enforcement agency not only to develop investigative strategy but also effective interviewing technique.

4.1 Proposed Model: Enhanced Systematic Digital Forensic Investigation Model (ESDFIM)

In this section, the proposed model will be discussed. The model consists of six major phases and the structure is illustrated in Fig. 4 and Fig.5.

A. Preparation Phase

Preparation phase is where all the work and activities that needs to be done before the actual investigation takes place. It includes but not limited to the studying applicable forensic laws and guidelines, obtaining search warrant, management support, planning, and setting up appropriate strategy and tools to be used. Monitoring devices like Intrusion Detection System, Intrusion Prevention systems, Honeypot/Honeynet and like tools may sometimes be used as detective and preventive techniques depending on the nature of the crime. These were completely missing in the existing models.

B. Acquisition and Preservation Phase

Acquisition and preservation phase is where the evidential life cycle starts from and the tasks performed include securing the crime scene, identifying and collecting both volatile and non-volatile evidence, labeling & packaging, transporting, image acquisition, storage and preservation of evidence. In general this phase is where relevant data are captured, stored and made available for the next phase. It is therefore important that every item searched and seized including access control, system and network architectures is legally obtained (plain view, search warrant, consent, etc.) and properly documented (chain of custody) in conformity to the evidential rule [16], [17], [18] [23] [24]. The existing models did not capture most of these activities.

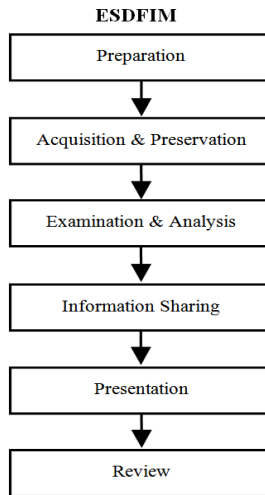


Fig. 4. Digital forensic investigation phases in the proposed model

C. Examination and Analysis Phase

Examination and analysis is where forensic examiners and experts look for digital evidence (Digital Evidence is defined by Carrier and Spafford [8] as digital data that supports or refutes a hypothesis about digital event or the state of digital data) by examining and analyzing the content of various digital devices which were legally seized and properly preserved. This is where the detail and technical job is done using approved guidelines and accredited forensic tools in order to identify the source of crime and ultimately trace whoever did it. The evidence to be generated will depend on the scope of engagement; the nature of the crime and also on the initial hypothesis and the result may or may not contradict the initial hypothesis, in order to prove culpability in the court of law [19].

D. Information Sharing Phase

Information sharing is the ability to exchange data between various countries, organizations, people, and technology (according to Techopedia.com). This weapon which is effectively used within the social networking sites and the hacking

community could be applicable in digital forensic investigation. [20]. The effectiveness of this tool however depends on certification of the information, mutual trust and understanding among law enforcement agencies, common cybercrime laws and investigation models being used in both countries else it will have a cascading effect on prosecution. One important advantage of information sharing is the ability to get full criminal profile of the suspect(s) [17] [21], which will effectively equip the law enforcement agencies to develop investigative strategy and effective interviewing techniques [22]. This form of cooperation and information sharing can contribute effectively towards successful prosecution.

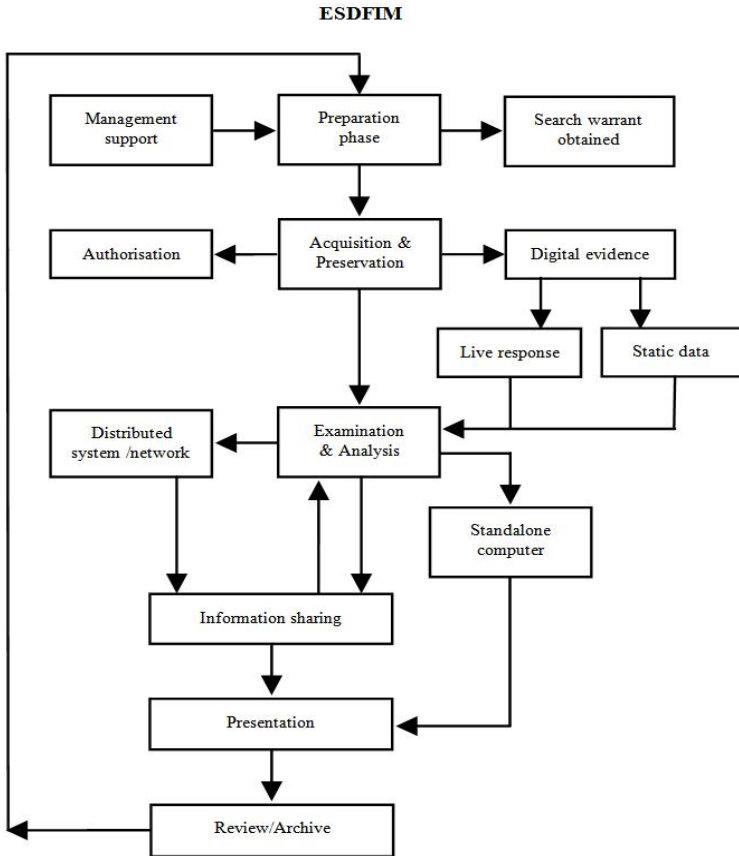


Fig. 5. Complete flow of a digital forensic investigation in the proposed model

E. Presentation Phase

The result of the examination and analysis phase is compiled and presented to the authority concerned. This is the critical stage of the investigation since the whole evidence can either be accepted or rejected. The admissibility of the evidence before the court of law for example depends on certain factors including but not limited to whether the evidence is materially and properly preserved, (chain of custody or

evidence), whether the evidence is relevant, properly identified and legally obtained, whether the language used in the presentation is simple and concise to be understood by the judge or the jury or whether the prosecution and his team can defend and prove intent, motive, identity or any error or mistake against the challenges and criticism of the accused/defendant's team. It is important to remember that the critical point in this phase is to present the findings to convince and prove your case before the trial judge or jury in a court of law.

F. Review Phase

The whole investigation is evaluated and areas of improvement identified. From the beginning of the investigation to court proceedings, and the result are used for future improvement. The experience gained and lessons learnt are shared and used to train new staff. Cases are also classified according to its status and remarks made in respect of whether the case is completed, suspended, pending and ongoing. This is done to guide future events such as a court appeal, reappearance of an acquitted person or for a reference. Evidence and exhibits which are returnable are given to their owners.

A unique feature of the proposed model is that it is not waterfall model but iterative in nature and therefore one has the ability to go back to the previous activity or phase when it becomes necessary that in doing so will help in the successful investigation and prosecution.

5 Comparison of the Proposed Model with Existing Models

A significant drawback in digital forensic investigation is that often not enough emphasis is placed on potential admissibility of the gathered evidence. Digital forensic investigation must adhere to the standard of evidence and its admissibility for successful prosecution. Therefore the techno-legal nature of the proposed model, coupled with the incorporation of best practices of existing models, will not only equip law enforcement agencies in their fight against computer criminals in both proactive and reactive ways but will also lead to successful prosecution. The following tables show our comparison of the proposed ESDFIM with the models discussed in this paper. Note that all relevant activities from previous models are included in the proposed model.

Table 1. Summary of phases and activities in existing digital forensic investigation models

Phase	Activities and considerations
Phase 1 Preparation	1. Preparation 2. Planning 3. Operational readiness 4. Infrastructural readiness 5. Survey 6. Awareness, 7. Communication 8. Assessment 9. Authorization and approval 10. Search warrant 11. Forensic laws
Phase 2 Acquisition & Preservation	1. Securing the crime scene 2. Identification and collection of evidence 3. Non-volatile evidence 4. Live response (volatile evidence) 5. Transport, labeling and packaging 6. Image acquisition 7. Storage and preservation 8. Documentation 9. Detection and notification 10. Reconnaissance
Phase 3 Examination & Analysis	1. Examination 2. Exploration testing 3. Hypothesis creation 4. Analysis 5. Tracing and reconstruction. 6. Dynamite
Phase 4 Presentation	1. Report 2. Testify 3. Proof and defense 4. Result 5. Presentation
Phase 5 Review	1. Review 2. Evaluation 3. Archival Storage 4. Return of evidence 5. Decision 6. Dissemination of information

Table 2. Comparison of the proposed ESDFIM model with existing digital forensic investigation models

Phase	Task/ Activities	ESDFIM	SDFIM	DFMDFI	DFMMIP	EDIP	IDIP	EMCI	ADFM	DFIM	DFRW
Preparation	Preparation	✓	✓	✓					✓		
	Planning	✓			✓			✓			
	Operation readiness	✓				✓	✓				
	Infrast. readiness	✓				✓	✓				
	Survey	✓				✓	✓				
	Awareness	✓	✓					✓			
	Assessment	✓	✓			✓	✓				
	Communication	✓		✓							
	Approach strategy	✓	✓						✓		
	Authorization & approval	✓	✓	✓	✓	✓	✓	✓			
	Forensic laws	✓	✓		✓						
	Search warrant	✓	✓		✓		✓	✓			
Honeypot/honeynet-like tools	✓										
Acquisition and Preservation	Securing crime scene	✓	✓			✓					
	Comm. shielding	✓	✓								
	Identification and collection	✓	✓	✓		✓	✓	✓	✓	✓	✓
	Reconnaissance	✓			✓						
	Deployment, detection & notification	✓				✓	✓	✓			
	Non-volatile evidence	✓	✓								
	Live response (volatile evidence)	✓	✓		✓						
	Documentation	✓	✓	✓		✓	✓				
	Transportation, labeling, packaging	✓	✓		✓			✓			
	Image acquisition	✓	✓			✓	✓				
Storage and preservation	✓	✓	✓	✓	✓	✓	✓	✓		✓	
Examination and Analysis	Examination	✓	✓	✓				✓	✓		✓
	Exploratory testing	✓		✓							
	Hypothesis creation	✓						✓			
	Analysis	✓	✓	✓	✓				✓	✓	✓
	Tracing and reconstruction	✓					✓	✓			
	Dynamite	✓					✓				
Information Sharing	Information sharing	✓				✓					
	Criminal profiling	✓									
	Interview techniques	✓									
	Interrogation	✓									
Presentation	Report/Result	✓	✓	✓	✓				✓		✓
	Presentation	✓	✓	✓		✓	✓	✓			✓
	Testify	✓									
	Proof and Defense	✓			✓			✓		✓	✓
Review	Review	✓	✓	✓		✓	✓				
	Evaluation	✓									
	Archival Storage	✓			✓						
	Return of evidence	✓							✓		
	Dissemination	✓						✓			

Table 3. Comparison of phases and objectives in the proposed ESDFIM model with the existing digital forensic investigation models

Model name	Authors	Year	No of phases	Frame of reference
ESDFIM	K. Kyei et al	2012	6	Effective for incidence response, cybercrime and computer fraud forensic investigation on both standalone and distributed systems with complex network architectures. Designed to lead to effective incidence prevention, incidence response, and successful prosecution.
SDFIM	A. Agawal et al	2011	11	Developed for helping forensic practitioners and organization for setting up appropriate policies and procedures in a systematic manner. Designed for computer fraud based on forensic laws
DFMFDI	I.O. Ademu et al	2011	4	The model identifies activities that facilitate and improve digital forensic investigation process
DFMMIP	S. Perumal	2009	7	Designed for cybercrime investigation based on Malaysia laws
IDIP	B. Carrier and E.H. Spafford	2004	5	This model integrates the physical crime scene into digital crime scene investigation to identify the perpetrator. It is suitable for both law enforcement and corporate investigations
EDIP	V. Baryamureeba and F. Tushabe	2004	5	Designed for cybercrime investigation and focuses on tracing all the way to the actual device used in by the criminal to commit the crime. It could also be adopted for incidence response.
EMCI	S.O. Ciardhuam	2004	13	The model provides a better understanding of the investigation process and captures most of the information flow of an entire cybercrime investigative process. Though generic it could be used for cybercrime investigation.
ADFM	M. Reith, C. Carr and G. Gunsch	2002	9	The basis of this model is using the ideals of traditional forensic evidence collection strategy. This model is an enhancement over previous models and useful for law enforcement agencies.
DFRWS	Palmer	2001	7	The model is considered as an enhancement compared to the previous models. Though it is only an investigative technique, it helps define the direction and challenges of digital forensic
DFIM	Kruse & Heiser	2001	3	Designed as an investigative technique

Table 4. New elements in the proposed digital forensic investigation model

Preparation	Intrusion detection and intrusion prevention systems, honeypots/honeynets for both prevention purposes and detection of suspicious activities. They are useful for both criminal investigation and organizational incidence response.
Information sharing	Cooperation, collaboration, trust, criminal profiling, interview and investigative techniques are essential ingredients in the proposed model for successful prosecution.

6 Conclusion

The objective of this paper is to review, analyze and identify gaps in the existing models in order to develop a holistic digital forensic investigation model which will enable law enforcement agencies to correctly investigate and successfully prosecute cybercriminals. It is believed that adoption of best practices from previous models and the inclusion of honeypot/honeynets etc, information sharing, criminal profiling as well as effective interview and interrogative techniques make it more detailed and comprehensive than the previous models. The new model, the enhanced systematic digital forensic investigation model, is expected to be not only useful to law enforcement agencies and organizations’ incident response teams, but will also provide a basis for the development of useful forensic tools.

References

1. Boateng, R., et al.: Cyber Crime and Criminality in Ghana: Its Forms and Implications. In: Proceedings of the 16th Americas Conference on Information Systems (2010)
2. Smith, R.G., Grabosky, P.N., Urbas, G.: Cybercriminals on trial. Cambridge University Press (2004) ISBN: 9780521840477
3. Kent, K., Chevalier, S., Grance, T., Dang, H.: NIST SP 800-86 Guide to Integrating Forensic Techniques into Incident Response (2006)
4. Agarwal, A., et al.: Systematic Digital Forensic Investigation Model (2011), <http://www.cscjournals.org/csc/manuscript/journals/IJCSS/Volume5/Issue1/IJCSS-438.pdf>
5. Kruse, W.J., Heiser, G.: Computer Forensics: Incident Response Essentials. Addison-Wesley (2002) ISBN 0-201-70719-5
6. Palmer, G.: A Road Map for Digital Forensic Research. Technical Report DTR-T001-01, DFRW, Report From the First Digital Forensic Research Workshop, Utica, NY (2001)
7. Reith, M., Carr, C., Gunsch, G.: An Examination of Digital Forensic Models. International Journal of Digital Evidence 1(3) (2002)
8. Carrier, B., Spafford, E.H.: Getting Physical with the Investigative Process. International Journal of Digital Evidence 2(2) (Fall 2003)
9. Lee, H., Palmbach, T., Miller, M.: Henry Lee's Crime Scene Handbook, Academic Press (2001) ISBN-13: 978-0124408302
10. Baryamureeba, V., Tushabe, F.: Enhanced Digital Investigation Process Model, Digital Forensic Research Workshop, Baltimore, MD, USA (2004)
11. Ciardhuáin, S.O.: An Extended Model of Cybercrime Investigations. In: International Journal of Digital Evidence 3(1) (Summer 2004)
12. Perumal, S.: Digital Forensic Model Based on Malaysian Investigation Process. IJCSNS International Journal of Computer Science and Network Security 9(8) (August 2009)
13. Ademu, I.O., Imafidon, C.O., Preston, D.S.: A New Approach of Digital Forensic Model for Digital Forensic Investigation. (IJACSA) International Journal of Advanced Computer Science and Applications 2(12) (2011)
14. Aghatise, E.J.: Computer Crime Research Center Cybercrime Definition (2006)
15. Carrier, B.: File System Forensic Analysis, Addison-Wesley (2005) ISBN 0-321-26817-2
16. Bunting, S.: Mastering Windows Network Forensic and Investigation, 1st edn. Sybex (2007) ISBN-13: 978-0470097625
17. Cressey, D.R.: Other People's Money: Study in the Social Psychology of Embezzlement. Wadsworth Publishing Company (1972) ISBN-13: 978-0534001421
18. Cosic, J., Baca, M.: A Framework to (Im)Prove "Chain of Custody" in Digital Investigation Process. In: Proceedings of the CECIIS, Varazdin, Croatia (2010)
19. Roger, M.K.: A social learning theory and moral disengagement analysis of criminal computer behavior: An exploratory study. University of Manitoba, Winnipeg (2001)
20. Biros, D.P., et al.: Information Sharing: Hackers vs. law enforcement. In: Proceedings of the 9th Australian Information Warfare and Security Conference, Perth, Australia (2008)
21. Stephenson, P.: Modeling of Post-Incident Root Cause Analysis. International Journal of Digital Evidence 2(2) (Fall 2003)
22. Turvey, B.: Criminal Profiling: An Introduction to behavioral evidence analysis, 4th edn. Elsevier (2012) ISBN 978-0-12-385243-4
23. ACFE Fraud Examiners Manual, Canadian Edition (2012)
24. Association of Chief Police Officers (ACPO): Good Practice Guide for Computer based Electronic Evidence (2006)

Author Index

- Al Jabri, Bedoor 91
Al Marzougy, Mohamed 239
Al-Safi, Deena 144
Al Shamlan, Moza 91
Al Zaabi, Ayesha 91
Aouad, Lamine M. 253
Appiah-Kubi, Oheneba Kwame 264
- BaAbdallah, Afrah 144
Baggili, Ibrahim 91, 144, 239
Baier, Harald 167
Beck, Marc B. 204
Breitinger, Frank 167
Bryan, Kevin 42
- Carthy, Joe 1, 22
- de Graaf, Daan 302
DiPippo, Lisa 42
Di Russo, Roberto 253
- Fasan, Oluwasola Mary 220
Fay-Wolfe, Victor 42
- Gent, Gerald 42
Gladyshev, Pavel 66, 183, 302
- Hannaway, Alan 66
- Idland, Christer 283
- James, Joshua I. 66
Jelle, Thomas 283
- Kechadi, Tahar M. 253
Koppen, Jeremy 42
Kramer, Jillian 42
Kyei, Kwaku 314
- Lindskog, Dale 314
Liu, Chen-Ching 66
- Marrington, Andrew 144, 239
McGrath, Niall 183
Mjøl̄snes, Stig F. 283
Mohamed, Abdallah 158
Moreland, Marquita 42
- Olivier, Martin S. 220
- Popov, Oliver 264
- Rouchka, Eric C. 204
Ruan, Keyun 1, 22
Ruhl, Ron 314
- Saleem, Shahzad 264
Seigfried-Spellar, Kathryn C. 81
Shosha, Ahmed F. 66, 302
Strikwerda, Litska 109
- Tennyson, Matthew F. 58
- Yampolskiy, Roman V. 158, 204
- Zavarsky, Pavol 314