

Linköping Studies in Science and Technology

Thesis No. 1361

# **Completing the Picture — Fragments and Back Again**

by

**Martin Karresand**

Submitted to Linköping Institute of Technology at Linköping University in partial  
fulfilment of the requirements for the degree of Licentiate of Engineering

Department of Computer and Information Science  
Linköpings universitet  
SE-581 83 Linköping, Sweden

Linköping 2008



# Completing the Picture — Fragments and Back Again

by

Martin Karresand

May 2008

ISBN 978-91-7393-915-7

Linköping Studies in Science and Technology

Thesis No. 1361

ISSN 0280-7971

LiU-Tek-Lic-2008:19

## ABSTRACT

Better methods and tools are needed in the fight against child pornography. This thesis presents a method for file type categorisation of unknown data fragments, a method for reassembly of JPEG fragments, and the requirements put on an artificial JPEG header for viewing reassembled images. To enable empirical evaluation of the methods a number of tools based on the methods have been implemented.

The file type categorisation method identifies JPEG fragments with a detection rate of 100% and a false positives rate of 0.1%. The method uses three algorithms, Byte Frequency Distribution (BFD), Rate of Change (RoC), and 2-grams. The algorithms are designed for different situations, depending on the requirements at hand.

The reconnection method correctly reconnects 97% of a Restart (RST) marker enabled JPEG image, fragmented into 4 KiB large pieces. When dealing with fragments from several images at once, the method is able to correctly connect 70% of the fragments at the first iteration.

Two parameters in a JPEG header are crucial to the quality of the image; the size of the image and the sampling factor (actually factors) of the image. The size can be found using brute force and the sampling factors only take on three different values. Hence it is possible to use an artificial JPEG header to view full of parts of an image. The only requirement is that the fragments contain RST markers.

The results of the evaluations of the methods show that it is possible to find, reassemble, and view JPEG image fragments with high certainty.

*This work has been supported by The Swedish Defence Research Agency and the Swedish Armed Forces.*





# Acknowledgements

This licentiate thesis would not have been written without the invaluable support of my supervisor Professor Nahid Shahmehri. I would like to thank her for keeping me and my research on track and having faith in me when the going has been tough. She is a good role model and always gives me support, encouragement, and inspiration to bring my research forward.

Many thanks go to Helena A, Jocke, Jonas, uncle Lars, Limpan, Micke F, Micke W, Mirko, and Mårten. Without hesitation you let me into your homes through the lenses of your cameras. If a picture is worth a thousand words, I owe you more than nine millions! I also owe a lot of words to Brittany Shahmehri. Her prompt and thorough proof-reading has indeed increased the readability of my thesis.

I would also like to thank my colleagues at the Swedish Defence Research Agency (FOI), my friends at the National Laboratory of Forensic Science (SKL) and the National Criminal Investigation Department (RKP), and my fellow PhD students at the Laboratory for Intelligent Information Systems (IISLAB) and the Division for Database and Information Techniques (ADIT). You inspired me to embark on this journey. Thank you all, you know who you are!

And last but not least I would like to thank my beloved wife Helena and our lovely newborn daughter. You bring happiness and joy to my life.

Finally I acknowledge the financial support by FOI and the Swedish Armed Forces.

*Martin Karresand*

Linköping, 14<sup>th</sup> April 2008



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Problem Formulation . . . . .	2
1.3	Contributions . . . . .	4
1.4	Scope . . . . .	5
1.5	Outline of Method . . . . .	6
1.6	Outline of Thesis . . . . .	6
<b>2</b>	<b>Identifying Fragment Types</b>	<b>9</b>
2.1	Common Algorithmic Features . . . . .	9
2.1.1	Centroid . . . . .	9
2.1.2	Length of data atoms . . . . .	10
2.1.3	Measuring Distance . . . . .	11
2.2	Byte Frequency Distribution . . . . .	11
2.3	Rate of Change . . . . .	14
2.4	2-Grams . . . . .	21
2.5	Evaluation . . . . .	22
2.5.1	Microsoft Windows PE files . . . . .	25
2.5.2	Encrypted files . . . . .	26
2.5.3	JPEG files . . . . .	27
2.5.4	MP3 files . . . . .	29
2.5.5	Zip files . . . . .	29
2.5.6	Algorithms . . . . .	30
2.6	Results . . . . .	30
2.6.1	Microsoft Windows PE files . . . . .	32
2.6.2	Encrypted files . . . . .	32
2.6.3	JPEG files . . . . .	33
2.6.4	MP3 files . . . . .	37
2.6.5	Zip files . . . . .	37
2.6.6	Algorithms . . . . .	40

<b>3</b>	<b>Putting Fragments Together</b>	<b>43</b>
3.1	Background . . . . .	43
3.2	Requirements . . . . .	46
3.3	Parameters Used . . . . .	47
3.3.1	Background . . . . .	47
3.3.2	Correct decoding . . . . .	49
3.3.3	Non-zero frequency values . . . . .	50
3.3.4	Luminance DC value chains . . . . .	51
3.4	Evaluation . . . . .	52
3.4.1	Single image reconnection . . . . .	53
3.4.2	Multiple image reconnection . . . . .	53
3.5	Result . . . . .	54
3.5.1	Single image reconnection . . . . .	54
3.5.2	Multiple image reconnection . . . . .	57
<b>4</b>	<b>Viewing Damaged JPEG Images</b>	<b>59</b>
4.1	Start of Frame . . . . .	59
4.2	Define Quantization Table . . . . .	66
4.3	Define Huffman Table . . . . .	67
4.4	Define Restart Interval . . . . .	70
4.5	Start of Scan . . . . .	72
4.6	Combined Errors . . . . .	75
4.7	Using an Artificial JPEG Header . . . . .	75
4.8	Viewing Fragments . . . . .	76
<b>5</b>	<b>Discussion</b>	<b>79</b>
5.1	File Type Categorisation . . . . .	79
5.2	Fragment Reconnection . . . . .	81
5.3	Viewing Fragments . . . . .	82
5.4	Conclusion . . . . .	83
<b>6</b>	<b>Related Work</b>	<b>85</b>
<b>7</b>	<b>Future Work</b>	<b>93</b>
7.1	The File Type Categorisation Method . . . . .	94
7.2	The Image Fragment Reconnection Method . . . . .	95
7.3	Artificial JPEG Header . . . . .	95
	<b>Bibliography</b>	<b>97</b>
<b>A</b>	<b>Acronyms</b>	<b>103</b>
<b>B</b>	<b>Hard Disk Allocation Strategies</b>	<b>105</b>
<b>C</b>	<b>Confusion Matrices</b>	<b>107</b>

# List of Figures

2.1	Byte frequency distribution of .exe . . . . .	13
2.2	Byte frequency distribution of GPG . . . . .	13
2.3	Byte frequency distribution of JPEG with RST . . . . .	14
2.4	Byte frequency distribution of JPEG without RST . . . . .	15
2.5	Byte frequency distribution of MP3 . . . . .	15
2.6	Byte frequency distribution of Zip . . . . .	16
2.7	Rate of Change frequency distribution for .exe . . . . .	18
2.8	Rate of Change frequency distribution for GPG . . . . .	18
2.9	Rate of Change frequency distribution for JPEG with RST . . . . .	19
2.10	Rate of Change frequency distribution for MP3 . . . . .	20
2.11	Rate of Change frequency distribution for Zip . . . . .	20
2.12	2-gram frequency distribution for .exe . . . . .	22
2.13	Byte frequency distribution of GPG with CAST5 . . . . .	25
2.14	ROC curves for Windows PE files . . . . .	33
2.15	ROC curves for an AES encrypted file . . . . .	34
2.16	ROC curves for files JPEG without RST . . . . .	34
2.17	ROC curves for JPEG without RST; 2-gram algorithm . . . . .	35
2.18	ROC curves for files JPEG with RST . . . . .	36
2.19	ROC curves for MP3 files . . . . .	38
2.20	ROC curves for MP3 files; 0.5% false positives . . . . .	38
2.21	ROC curves for Zip files . . . . .	39
2.22	Contour plot for a 2-gram Zip file centroid . . . . .	40
3.1	The frequency domain of a data unit . . . . .	45
3.2	The zig-zag ordering of a data unit traversal . . . . .	46
3.3	The scan part binary format coding . . . . .	49
4.1	The original undamaged image . . . . .	60
4.2	The Start Of Frame (SOF) marker segment . . . . .	60
4.3	Quantization tables with swapped sample rate . . . . .	62
4.4	Luminance table with high sample rate . . . . .	62
4.5	Luminance table with low sample rate . . . . .	64
4.6	Swapped chrominance component identifiers . . . . .	64
4.7	Swapped luminance and chrominance component identifiers . . . . .	65
4.8	Moderately wrong image width . . . . .	65

4.9	The Define Quantization Table (DQT) marker segment . . . . .	66
4.10	Luminance DC component set to 0xFF . . . . .	68
4.11	Chrominance DC component set to 0xFF . . . . .	68
4.12	The Define Huffman Table (DHT) marker segment . . . . .	69
4.13	Image with foreign Huffman tables definition . . . . .	71
4.14	The Define Restart Interval (DRI) marker segment . . . . .	71
4.15	Short restart interval setting . . . . .	71
4.16	The Start Of Scan (SOS) marker segment . . . . .	72
4.17	Luminance DC Huffman table set to chrominance ditto . . . . .	74
4.18	Complete exchange of Huffman table pointers . . . . .	74
4.19	A correct sequence of fragments . . . . .	78
4.20	An incorrect sequence of fragments . . . . .	78
5.1	Possible fragment parts . . . . .	82

# List of Tables

2.1	Camera make and models for JPEG with RST . . . . .	28
2.2	Camera make and models for JPEG without RST . . . . .	28
2.3	MP3 files and their encoding . . . . .	29
2.4	Algorithm base names used in evaluation . . . . .	31
2.5	File type information entropy . . . . .	32
2.6	Centroid base names used in evaluation . . . . .	41
2.7	2-gram algorithm confusion matrix . . . . .	41
2.8	Large GPG centroid 2-gram algorithm confusion matrix . . . . .	42
3.1	Multiple image reconnection evaluation files . . . . .	54
3.2	Results for the image fragment reconnection method; single images	55
3.3	Results for the image fragment reconnection method; multiple images . . . . .	57
4.1	Relation between image width and height . . . . .	66
B.1	Data unit allocation strategies . . . . .	106
C.1	Centroid base names used in evaluation . . . . .	107
C.2	Confusion matrix: 2-gram algorithm . . . . .	108
C.3	Confusion matrix: BFD with JPEG rule set . . . . .	108
C.4	Confusion matrix: BFD and RoC with JPEG rule set . . . . .	108
C.5	Confusion matrix: BFD and RoC with Manhattan dist. metric . .	108
C.6	Confusion matrix: BFD and RoC . . . . .	109
C.7	Confusion matrix: BFD . . . . .	109
C.8	Confusion matrix: BFD and RoC with JPEG rule set and signed values . . . . .	109
C.9	Confusion matrix: BFD and RoC using signed values and Man- hattan distance metric . . . . .	109
C.10	Confusion matrix: BFD and RoC using signed values . . . . .	110
C.11	Confusion matrix: RoC with JPEG rule set and signed values . .	110
C.12	Confusion matrix: RoC using Manhattan distance metric and signed values . . . . .	110
C.13	Confusion matrix: RoC using signed values . . . . .	110
C.14	Confusion matrix: RoC with JPEG rule set . . . . .	111

C.15 Confusion matrix: RoC using Manhattan distance metric . . . . .	111
C.16 Confusion matrix: RoC . . . . .	111



# Chapter 1

## Introduction

The work presented in this thesis is directed at categorising and reconnecting Joint Photographic Experts Group (JPEG) image fragments, because these capabilities are important when searching for illegal material. In this chapter we describe the motivation for our research, state the research problem and scope, present the contributions of our work, and finally draw the outline of the thesis.

### 1.1 Motivation

In a paper from the US Department of Justice [1, p. 8] it is stated that

The Internet has escalated the problem of child pornography by increasing the amount of material available, the efficiency of its distribution, and the ease of its accessibility.

Due to the increasing amount of child pornography the police need the possibility to scan hard disks and networks for potential illegal material more efficiently [2, 3, 4]. Identifying the file type of fragments from the data itself makes it unnecessary to have access to the complete file, which will speed up scanning. The next requirement, then, is to make it possible to determine whether an image fragment belongs to an illegal image, and to do that by viewing the partial picture it represents. In this way the procedure becomes insensitive to image modifications.

The fact that it is possible to make identical copies of digital material, something which cannot be done with physical entities, separates digital evidence from physical evidence. The layer of abstraction introduced by the digitisation of an image also simplifies concealment, for example by fragmenting a digital image into small pieces hidden in other digital data. A criminal knowing where to find the fragments and in which order they are stored can reconnect them. In this way he or she can recreate the original image without loss of quality, which is not true for a physical image.

The abstract nature of digital material makes it harder for the police to find illegal images, as well as connect such images to a perpetrator. The police are constantly trying to catch and prosecute the owners of illegal material, but are fighting an “uphill battle” [5, 6]. The new technology makes it easier for the criminals to create and distribute the material, and also provides a higher degree of anonymity.

Consequently there is a need for tools that are able to work with fragmented data, and especially digital image files, transported over a network or held on some kind of storage media. Regarding the imminent problem of keeping the amount of digital child pornography at bay the police need tools that are able to discover image data in all possible situations, regardless of the state of the image data, even at the lowest level of binary data fragments. Preferably the tools should be able to quickly and accurately identify the file type of data fragments and then combine any image fragments into (partial) pictures again.

## 1.2 Problem Formulation

The procedure for recovering lost data is called file carving and can be used in a wide range of situations, for example, when storage media have been corrupted. Depending on the amount of available file recovery metadata<sup>1</sup> the task can be very hard to accomplish. At the web page of the 2007 version of the annual forensic challenge issued by the Digital Forensic Research Workshop (DFRWS) it is stated that [7]:

Many of the scenarios in the challenge involved fragmented files where fragments were sequential, out of order, or missing. Existing tools could not handle these scenarios and new techniques had to be developed.

DFRWS reports that none of the submissions to the forensic challenge of year 2007 completely solve the problems presented.

The main characteristic of a fragmented data set is its lack of intuitive structure, i.e. the data pieces are randomly scattered. The structuring information is held by the file metadata. The metadata of files on a hard disk come in the form of a file system. The file system keeps track of the name and different time stamps related to the use of a file. It also points to all hard disk sectors making up a file. Since files often require the use of several sectors on a hard disk and the size of a file often changes during its lifetime, files are not always stored in consecutive sectors. Instead they become fragmented, even though current file systems use algorithms that minimise the risk of fragmentation [8] (see also Appendix B). When a hard disk crashes and corrupts the file system, or even when a file is deleted, the pointers to the sectors holding a file are lost, and hence the fragments that moments before together formed a file now only are pieces of random data.

---

<sup>1</sup>In this thesis we define file recovery metadata to be data indirectly supporting the file recovery procedure by structuring the data to be recovered.

## 1.2. PROBLEM FORMULATION

In the case of network traffic the structuring information is often held by the Transmission Control Protocol (TCP) header information, or possibly by a protocol at a higher layer in the network stack. When monitoring network traffic for illegal material a full TCP session is required to read a transferred file and its content, but the hardware and different protocols used in a network often fragments files. The routing in a network causes the packets to be transported along different paths, which limits the amount of data that is possible to collect at an arbitrary network node, because there is no guarantee that all fragments of a file passes that particular node. If parts of a TCP session are encountered it is possible to correctly order the network packets at hand, but parts of the transferred data are lost and hence we have the same situation as for a corrupted file system.

To the best of our knowledge current tools either use parts of a file system to find files and categorise their content or use metadata in the form of header or footer information. The tools are built on the assumption that files are stored in consecutive hard disk sectors. When a tool has identified the start and probable end of a file all hard disk sectors in between are extracted. In other words, the tools are built on qualified guesses based on high level assumptions about how the data would usually be structured, without really using the information residing in the data itself.

A JPEG image is a typical example showing the structure of a file. The image file consists of a file header containing image metadata and tables for decoding, a section of compressed picture data, and a finalising file footer. The picture data part consists of a stream of raw data corresponding to a horizontal and downwards traversal of the image pixels. To allow a JPEG image file to be correctly decoded the picture data has to be properly ordered. Since existing file carving tools are relying on file headers and that data is stored in consecutive order it is currently not possible to reassemble a fragmented JPEG image.

The problems related to file carving of JPEG images give rise to the following research questions:

- How can file fragments be categorised without access to any metadata?
- How can fragmented files be reassembled without access to any metadata?
- What techniques can be used to enable viewing of a reassembled JPEG image?
  - How much can be seen without access to a JPEG header?
  - Can an artificial JPEG header be used?
  - What requirements are there on a working JPEG header?

These research questions cover the process of identifying, reassembling and viewing fragmented JPEG images, without having access to anything but the fragments themselves.

### 1.3 Contributions

The overall aims of our research are to explore what information can be extracted from fragments of high entropy data<sup>2</sup>, what parameters govern the amount of information gained, and finally to use the parameters found to develop effective, efficient and robust methods for extracting the information. The research includes studies of the parameters needed to extract different amounts of information, and at which level of abstraction that can be done. The ultimate goal is to be able to reconnect any identified file fragments into the original file again, if at all possible.

The main contributions of our work lie within the computer forensics and data recovery area, exemplified by fragmented JPEG image files. We have developed and evaluated a method<sup>3</sup> to categorise the file type of unknown digital data fragments. The method currently comprises three algorithms, which are used both as stand alone algorithms and in combination. The algorithms are based on separate parameters found to increase the ability to find the file type of unknown data fragments.

To supplement the file type categorisation algorithms, we have studied parameters that enable reconnection of fragments of a file, making it possible to rebuild a fragmented image file. The method to reconnect fragments is accompanied by experimental results showing the importance of different parameters in the JPEG header fields for the viewing of a restored image, or even a partial image. This enables us to create generic JPEG file headers to be added to the recovered images, if there are no proper header parts to be found among the fragments.

The details of our contributions are as follows:

- We present two algorithms, Byte Frequency Distribution (BFD) [9] and Rate of Change (RoC) [10], which are using parameters in the form of statistical measures of single byte relations and frequencies to identify the file type of unknown digital data fragments. The algorithms do not require any metadata to function and executes in linear time. They are well suited for file types without any clear structure, such as encrypted files.
- A third algorithm, called the 2-gram algorithm [11, 12], which is using parameters in the form of statistical properties of byte pairs to identify the file type of unknown digital data fragments. The 2-gram algorithm is suitable for situations where a high detection rate in combination with a low false positives rate is preferable, and a small footprint and fast execution is of less importance. The algorithm does not need any metadata to function and is well suited for file types with some amount of structure. Its high sensitivity to structure can be used to find hidden patterns within file types.

---

<sup>2</sup>We define high entropy data to be binary files in the form of, for example, compiled source code, compressed data, and encrypted information.

<sup>3</sup>The method is called “the Oscar method” in our previously published papers [9, 10, 11, 12].

- We introduce a method to find and calculate a value indicating the validity of two JPEG data fragments being consecutive. Several parameters work in tandem to achieve this, the most important parameter being the DC luminance value chain of an image. The method is currently implemented for JPEG data with Restart (RST) markers, which are used to resynchronise the data stream and the decoder in case of an error in the data stream. By using the method repeatedly digital JPEG image files can be rebuilt as long as all fragments of the images are available. If only a sub-set of all fragments of an image is available, the correct order of the fragments at hand can be found in most cases.
- We explore what impact modifications to fields in a JPEG header have on a displayed image. The results of these experiments can be used to create a generic JPEG file header making it possible to view the result of the JPEG image fragment reconnection method, even for only partially recovered images.

The parameters the algorithms are built on are applied to fragmented JPEG image files in this thesis, to help improve the police's ability to search for child pornography. The parameters and methodologies are generalisable and may be applied to other file types as well with minor modifications. Our work therefore indirectly contributes to a wide area of practical applications, not only to JPEG images and child pornography scanning.

### 1.4 Scope

The scope of the work presented in this thesis covers identification of JPEG image fragments and reconnection of JPEG image fragments containing RST markers. The images used for the research are produced by digital cameras and a scanner. None of the images are processed by any image manipulation application or software, apart from the cameras' internal functions and the scanner's software.

The set of images has been collected from friends and family and are typical family photographs. The contributors have selected the images themselves and approved them to be used in our research. We have allowed technical imperfections such as blur, extreme low and high key, and large monochrome areas to help improve the robustness of the developed methods.

The cameras used for the research are typical consumer range digital cameras, including one Single Lens Reflex (SLR) camera. The built-in automatic exposure modes of the cameras have been used to a different extent; we have not taken any possible anomalies originating from that fact into account.

The JPEG images used to develop the methods presented in the thesis are of the non-differential Huffman entropy coded baseline sequential Discrete Cosine Transform (DCT) JPEG type. The images adhere to the International Telegraph and Telephone Consultative Committee (CCITT) Recommendation T.81 [13] standard, which is the same as the International Organization for

Standardization (ISO)/International Electrotechnical Commission (IEC) International Standard 10918-1, and their accompanying extensions [14, 15, 16, 17]. Consequently the file type categorisation and fragment reassembly methods also follow that standard, where applicable.

## 1.5 Outline of Method

The work in this thesis can be divided into three main parts:

- categorisation of the file type of unknown fragments,
- reassembly of JPEG image fragments, and
- the requirements put on an artificial JPEG image header.

The method comprised by the above mentioned steps can be compared to the method used to piece together a large jigsaw puzzle. First the pieces are sorted into groups of sky, grass, sea, buildings, etc. Then the individual features of the pieces are used to decide how they should be fitted together. Lastly the wear and tear of the pieces governs if it is possible to correctly fit them together.

The method for file type categorisation utilises the byte frequency distribution, the frequency distribution of the derivative of a sequence of bytes, and the frequency distribution of byte pairs to categorise the file type of binary fragments. A model representing a specific file type is compared to an unknown fragment and the difference between the model and the fragment is measured and weighted by the standard deviation of each frequency value. Several models are compared to the unknown sample, which is then categorised as being of the same file type as the closest model.

Successful reassembly of a set of JPEG image fragments is achieved by using repetitions in the luminance DC value chain of an image. When two fragments are correctly connected vertical lines are repeated at an interval equalling  $\frac{1}{8}$  of the image width in pixels. The reassembly method also uses parameters related to the decoding of the image data. By measuring the validity of a joint between two fragments and then minimising the cost of a sequence of connected fragments, a correct image can be rebuilt with high probability.

The third part identifies the parameters which are crucial to a proper decoding of an image. These parameters are then used to form an artificial JPEG image header. The effects of different kinds of manipulation of the header fields are also explored.

## 1.6 Outline of Thesis

**Chapter 1:** The current chapter, motivating the research and outlining the research problems, contributions, scope, outline of the work and organisation of the thesis.

- Chapter 2:** Presents a method to identify the file type of unknown data fragments. The three different algorithms used by the file type categorisation method, BFD, RoC, and 2-grams, are presented and evaluated.
- Chapter 3:** The JPEG image fragment reconnection methodology for reconnecting JPEG fragments is presented and evaluated.
- Chapter 4:** The possibility of viewing images damaged in some way is discussed. The damage can come from erroneous header information or only parts of the scan being reconnected, hence leading to a mismatch between the header and scan parts.
- Chapter 5:** The research and results are discussed in this chapter, focusing on alternative methods, limitations, problems and their (possible) solutions.
- Chapter 6:** This chapter gives an overview of the related work in the file carving and fragmented image file reconstruction arenas. The chapter also covers the similarities and differences between the related work and the research presented in this thesis.
- Chapter 7:** Presents the conclusions to be drawn from the material presented in the thesis, together with subjects left as future work.





## Chapter 2

# Identifying Fragment Types

This chapter presents three algorithms, Byte Frequency Distribution (BFD), Rate of Change (RoC), and 2-grams, which are part of the method for file type categorisation of data fragments. The algorithms are all based on statistical measures of frequency (mean and standard deviation). They are used to create models, here called *centroids* [18, p. 845], of different file types. The similarity between an unknown fragment and a centroid is measured using a weighted variant of a quadratic distance metric. An unknown fragment is categorised as belonging to the same type as the closest centroid. Alternatively a single centroid is used together with a threshold; if the distance is below the threshold the fragment is categorised as being of the same file type as the centroid represents.

Hard disks store data in *sectors*, which are typically 512 bytes long. The operating system then combines several sectors into *clusters*, sometimes also called *blocks*. Depending on the size of the partition on the hard disk, clusters can vary in size, but currently the usual cluster size is 4 KiB, which is often also the size of *pages* in Random Access Memory (RAM). The file type categorisation method is currently set to use data blocks of 4 KiB, but this is not a requirement, the method should be able to handle fragments as small as 512 B without becoming unstable.

## 2.1 Common Algorithmic Features

The three algorithms share some basic features, which will be discussed in this section. These features are the use of a centroid as a model of a specific file type, small sized data atoms, and a weighted (quadratic) distance metric.

The BFD and RoC algorithms have been tested with some alternative features. These features are presented together with each algorithm specification.

### 2.1.1 Centroid

The centroid is a model of a selection of characteristics of a specific file type and contains statistical data unique to that file type. In our case the centroid

consists of two vectors representing the mean and standard deviation of each byte's or byte pair's frequency distribution, or the frequency distribution of the difference between two consecutive byte values.

To create a centroid for a specific file type we use a large amount of known files of that type. The files are concatenated into one file. The resulting file is then used to calculate the mean and standard deviation of the byte value count for a 4 KiB large fragment. The 2-gram method uses 1 MiB large fragments and the results are scaled down to fit a 4 KiB fragment.

The aim is to model a real world situation as closely as possible. We therefore collect all files used in the experiments from standard office computers, the Internet, and private consumer digital cameras. Some files were cleaned of unwanted impurities in the form of other file types. This is described in detail in the following sections.

### 2.1.2 Length of data atoms

An  $n$ -gram is an  $n$ -character long sequence where all characters belong to the same alphabet of size  $s$ , in our case the American Standard Code for Information Interchange (ASCII) alphabet giving  $s = 256$ . The byte frequency distribution is derived by finding the frequency with which each unique  $n$ -gram appears in a fragment. These frequencies then form a vector representing the sample to be compared to a centroid. The data the file type categorisation method uses is in the form of 1-grams and 2-grams, i.e. bytes and byte pairs.

The order of the bytes in the data is not taken into consideration when using a 1-gram method. This fact gives such methods a higher risk of false positives than methods using longer sequences of bytes. By considering the order of the bytes we decrease the risk of false positives from disk clusters having the same byte frequency distribution as the data type being sought, but a different structure. Using larger  $n$ -grams increases the amount of ordering taken into consideration and is consequently a nice feature to use. However, when using larger  $n$ -grams the number of possible unique grams,  $b$ , also increases. The size of  $b$  depends on whether the  $n$ -grams have a uniform probability distribution or not. The maximum value  $b = s^n$ , is required when the distribution is uniform, because then all possible  $n$ -grams will be needed. Therefore the execution time and memory footprint of the 2-gram algorithm is increased exponentially, compared to the 1-gram BFD and RoC algorithms, for high entropy file types.

The BFD and RoC algorithms use a 256 character long alphabet, which is well suited for 4 KiB large data fragments. Using 2-grams may require  $b = 65536$ , otherwise not all possible 2-grams can be accounted for. Consequently training on data blocks less than 64 KiB in size affects the quality of the 2-gram centroid. We therefore use 1 MiB large data blocks when collecting the statistics for the 2-gram centroid. The mean and standard deviation values are then scaled down to match a fragment size of 4 KiB. Since the scan part of a JPEG has a fairly even byte frequency distribution this method gives a mean value close to 1 for most of the 2-grams. Although the frequency count is incremented integer-wise, and thus every hit will have a large impact on the calculations of the distance, the

## 2.2. BYTE FREQUENCY DISTRIBUTION

2-gram algorithm performs well for JPEG and MPEG 1, Audio Layer-3 (MP3) file fragments.

### 2.1.3 Measuring Distance

The distance metric is the key to a good categorisation of the data. Therefore we use a quadratic distance metric, which we extend by weighting the difference of each individual byte frequency with the same byte's standard deviation. In this way we focus the algorithm on the less varying, and hence more important, features of a centroid.

The metric measures the difference between the mean value vector,  $\vec{c}$ , of the centroid and the byte frequency vector,  $\vec{s}$ , of the sample. The standard deviation of byte value  $i$  is represented by  $\sigma_i$  and to avoid division by zero when  $\sigma_i = 0$  a smoothing factor  $\alpha = 0.000001$  is used. The metric is described as

$$d(\vec{s}, \vec{c}) = \sum_{i=0}^{n-1} (s_i - c_i)^2 / (\sigma_i + \alpha). \quad (2.1)$$

The advantage of using a more computationally heavy quadratic-based metric over a simpler linear-based metric is the quadratic-based method's ability to strengthen the impact of a few large deviations over many small deviations. Assuming the vector sums,  $\|\vec{c}\|_1$  and  $\|\vec{s}\|_1$ , are constant, Equation (2.1) gives lower distance values for two vectors separated by many small deviations, than for two vectors separated by a few large deviations. A linear-based method gives the same distance, regardless of the size of the individual deviations, as long as the vector sums remain the same. Since we are using 4 KiB blocks of data the vector sums  $\|\vec{c}\|_1 = \|\vec{s}\|_1 = 4096$  and consequently we have to use a quadratic distance metric to achieve decent categorisation accuracy.

Some file types generate fairly similar histograms and it is therefore necessary to know which byte codes are more static than others to discern the differences in spectrum between, for example, an executable file and a JPEG image. We therefore have to use the more complex method of looking at the mean and standard deviation of individual byte codes, instead of calculating two single values for the vector.

## 2.2 Byte Frequency Distribution

The byte frequency distribution algorithm counts the number of occurrences of each byte value between 0 and 255 in a block of data. We use the mean and standard deviation of each byte value count to model a file type. The mean values are compared to the byte count of an unknown sample and the differences are squared and weighted by the standard deviations of the byte counts in the model. The sum of the differences is compared to a predefined threshold and the sample is categorised as being of the same type as the modelled file if the sum of the differences is less than the threshold.

The size of the data blocks are currently 4 KiB to match the commonly used size of RAM pages and disc clusters. The method does not require any specific data block size, but a block size of 512 bytes is recommended as a lower bound to avoid instability in the algorithm.

A special JPEG extension is used together with the BFD and RoC algorithms. This extension utilises some of the JPEG standard specifics regulating the use of special marker segments in JPEG files. By counting the number of occurrences of some markers we can significantly improve the detection rate and lower the false alarm rate of the algorithm. The extension is implemented as a rule set to keep track of marker sequences. If the algorithm finds any pair of bytes representing a disallowed marker segment in a fragment, the fragment will not be categorised as JPEG.

In Figures 2.1–2.6 the byte frequency distribution of the centroids of five different file types, Microsoft Windows Portable Executable (PE) files (.exe), GNU Privacy Guard (GPG) encrypted files (GPG), JPEG images (JPEG), MP3 audio files (MP3), and Zip compressed files (Zip), are shown. Please observe that the JPEG histogram is based only on the data part of the files used for the centroid, the other centroids are based on complete, although cleaned, files. Also note that the scaling of the Y-axis differs between the figures. This is done to optimise readability. As can be seen, the histogram of the executable files are different from the compressed GPG, JPEG, MP3, and Zip files. Even though the histograms of the compressed files are fairly similar, they still differ, with the most uniform histogram belonging to the GPG type, then the Zip, MP3, and last the JPEG.

The executable files used to create this centroid were manually cleaned from JPEG, Graphics Interchange Format (GIF), and Portable Network Graphics (PNG) images and Figure 2.1 should therefore give an idea of what the executable parts of Windows PE files look like. There are however both tables and string parts left in the data used to create the centroid, which makes the centroid more general and less exact. The significantly higher rates of  $0x00$ <sup>1</sup> and  $0xFF$  for the executable file centroid is worth noticing. The printable ASCII characters remaining in the compiled code can be seen in the byte value range of 32 to 126. There is a segment with lower variation in the approximate byte value range of 145 to 190.

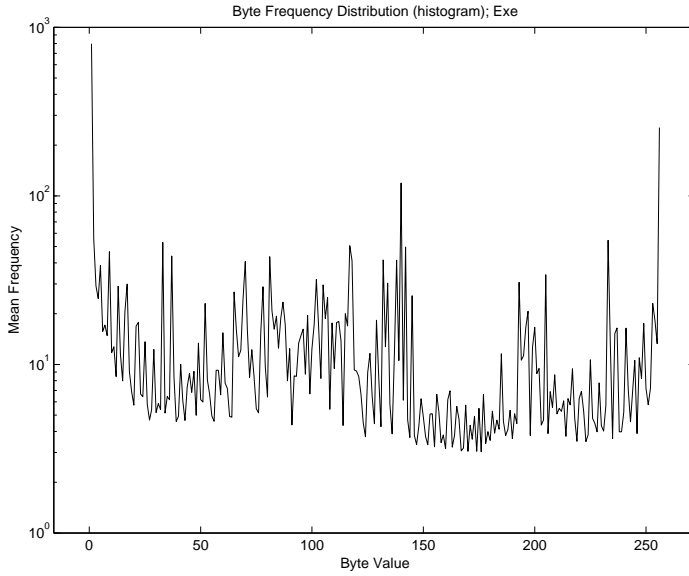
Figure 2.2 shows a diagram of the byte frequency distribution of an encrypted file. In this case the file is encrypted using GPG 1.4.6 and the Advanced Encryption Standard (AES) algorithm with a 128 bit long key. The frequency distribution is almost flat, which is an expected feature of an encrypted file.

The JPEG histogram in Figure 2.3 shows an increase in the number of  $0xFF$  bytes, because of the RST markers used. The JPEG centroid without RST markers in Figure 2.4 does not show the same increased frequency level at the highest byte value as can be seen in Figure 2.3. There is also a noticeable larger amount of  $0x00$  in both centroids. This is because of the zero value required to follow a non-marker  $0xFF$ . Hence the byte values are fairly evenly distributed in the raw

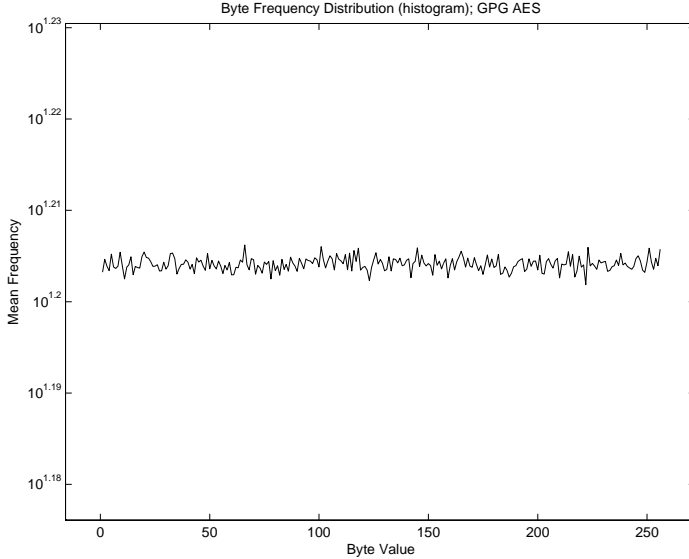
---

<sup>1</sup>Throughout the thesis hexadecimal values will be denoted as  $0xYY$ , where  $YY$  is the hexadecimal value of a byte.

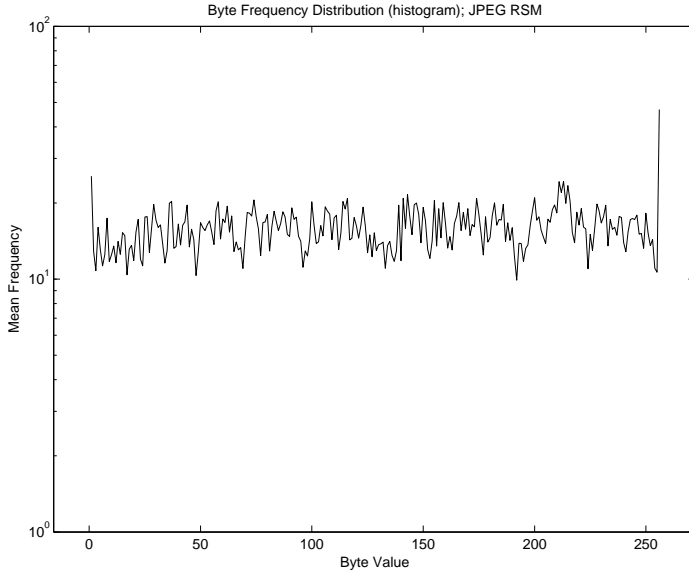
## 2.2. BYTE FREQUENCY DISTRIBUTION



**Figure 2.1:** The byte frequency distribution of a collection of executable Microsoft Windows PE files. A logarithmic scale is used for the Y-axis and the frequency values correspond to 4 KiB of data.



**Figure 2.2:** The byte frequency distribution of a large file encrypted with GPG using the AES algorithm with a 128 bit long key. A logarithmic scale is used for the Y-axis and the frequency values correspond to 4 KiB of data.



**Figure 2.3:** The byte frequency distribution of the data part of a collection of JPEG images containing RST markers. A logarithmic scale is used for the Y-axis and the frequency values correspond to 4 KiB of data.

JPEG coding and the extra zeroes added are clearly visible in a histogram.

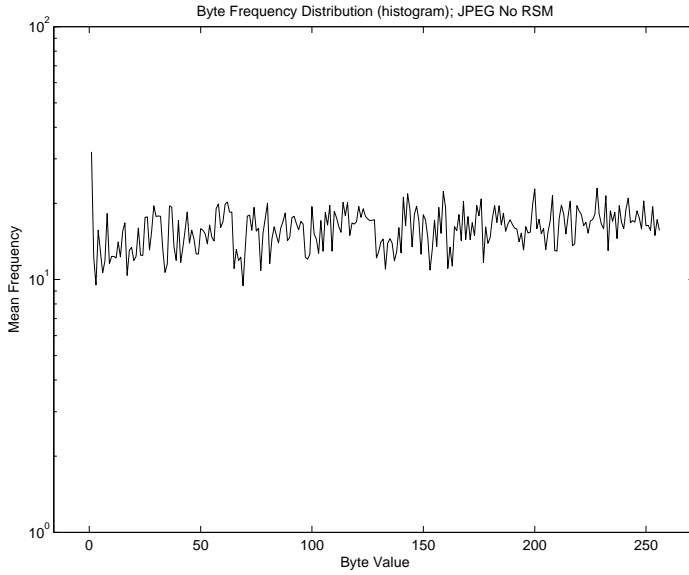
The appearance of a MP3 file centroid can be seen in Figure 2.5. The byte frequency distribution diagram was created from 12 MP3 files stripped of their IDentify an MP3 (ID3) tags. The use of a logarithmic scale for the Y-axis enhances a slight saw tooth form of the curve.

The byte frequency diagram for a Zip file centroid, which can be seen in Figure 2.6, has a clearly visible saw tooth form, which is enhanced by the scaling; all values in the plot fit into the range between 14.3 and 18.6. The reason for the saw tooth form might be the fact that the Huffman codes used for the DEFLATE algorithm [19, p. 7] are given values in consecutive and increasing order. The peaks in the plot lay at hexadecimal byte values ending in  $0xF$ , i.e. a number of consecutive 1s. The Huffman codes have different lengths, where shorter are more probable, and our experience is that they more often combine into a number of consecutive 1s rather than 0s.

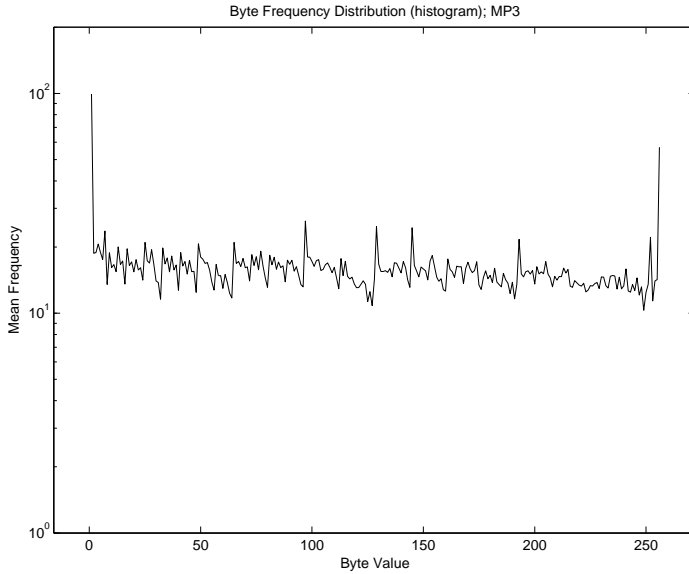
## 2.3 Rate of Change

We define the Rate of Change (RoC) as the value of the difference between two consecutive byte values in a data fragment. The term can also be defined as the value of the derivative of a byte stream in the form of a data fragment. By utilising the derivative of a byte stream, the ordering of the bytes is to some

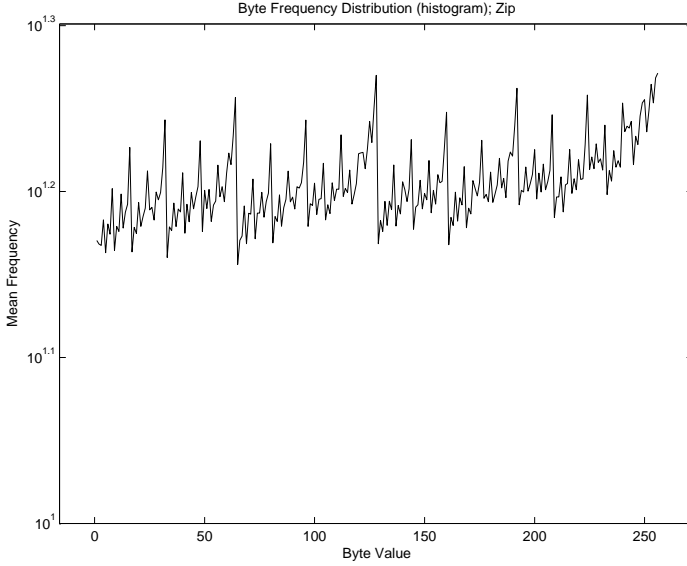
### 2.3. RATE OF CHANGE



**Figure 2.4:** The byte frequency distribution of the data part of a collection of JPEG images without RST markers. A logarithmic scale is used for the Y-axis and the frequency values correspond to 4 KiB of data.



**Figure 2.5:** The byte frequency distribution of a collection of MP3 files. A logarithmic scale is used for the Y-axis and the frequency values correspond to 4 KiB of data.



**Figure 2.6:** The byte frequency distribution of a collection of compressed Zip files. A logarithmic scale is used for the Y-axis and the frequency values correspond to 4 KiB of data.

extent taken into consideration, but the algorithm cannot tell what byte values give a specific rate of change, apart from a rate of change of 255, of course.

The frequency distribution of the rate of change is used in the same way as the byte frequency distribution in the BFD algorithm. A centroid is created, containing a mean vector and a standard deviation vector. The centroid is then compared to a sample vector by measuring the quadratic distance between the sample and the centroid, weighted by the standard deviation vector of the centroid, as described in Equation (2.1).

The original RoC algorithm [10] looks at the absolute values of the difference in byte value between two consecutive bytes. Hence the algorithm cannot tell whether the change is in a positive or negative direction, i.e. a positive or negative derivative of the byte stream. The number of rate of change values of a model and a sample are compared using the same weighted sum of squares distance metric as the BFD algorithm uses (see Equation (2.1)). We also experiment with an alternative distance metric using the 1-norm of the differences between a model and a sample. We have further extended the RoC algorithm by using signed rate of change values (difference values), as well as added the JPEG rule set extension from the BFD algorithm (see Section 2.2).

The reason for originally using the absolute value of the byte value difference and not a signed value is that since the difference value range is bounded, the sum of the signed byte value differences will be equal to the difference between the first and last byte in a sequence. By using the absolute value of the differences we



### 2.3. RATE OF CHANGE

get an unbounded sequence difference sum, and can in that way use the value as a similarity measure, if needed. We therefore have made the decision that the loss of resolution induced by the absolute value is acceptable. Another reason is that the higher resolution of a signed value difference possibly affects the statistical metrics used, because the statistics are based on half the amount of data. There is also a risk of false negatives if the centroid becomes too specialised and strict.

To improve the detection ability of the RoC algorithm the standard deviation of the centroid can be used. If we assume a data fragment with a fully random and uniform byte distribution it would have a histogram described by  $y = 256 - x$ , where  $y$  is the number of rate of changes of a certain value  $x$  for  $x = [1, 2, \dots, 255]$ . When the byte distribution becomes ordered and less random the histogram would become more disturbed, giving a standard deviation  $\sigma$  larger than zero. Therefore the value of  $\sigma$  for the sample and centroid vectors could be used as an extra measure of their similarity.

Figure 2.7 to Figure 2.11 shows the frequency distribution of the rate of change for five different file types. As can be seen in the figures, there are differences between the curves of the compressed (JPEG, GPG, MP3 and Zip) files, and the executable Windows PE files. It is also possible to see that the Zip file has a smoother curve, i.e. lower standard deviation, than the JPEG file. The reasons for the bell shaped curves are the logarithmic scale of the Y-axis together with the fact that, assuming a fully random and uniform byte distribution, the probability of difference  $x$  is described by

$$p(x) = \frac{256 - x}{\frac{(256+1) \cdot 256}{2}}; \quad x = [0, 1, \dots, 255].$$

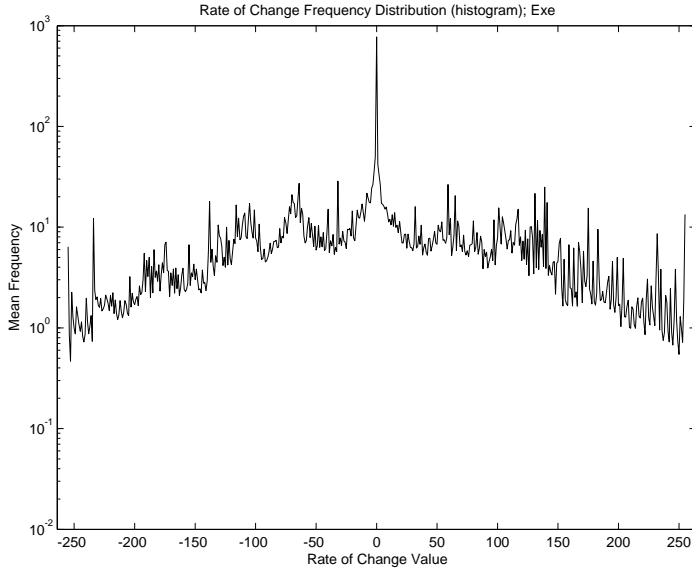
Hence there are more ways to get a difference value close to zero, than far from zero. The sum of the negative RoC frequencies cannot differ from the sum of the positive values by more than 255, i.e.

$$-255 \leq \sum_1^{255} \text{RoC}^- - \sum_1^{255} \text{RoC}^+ \leq 255$$

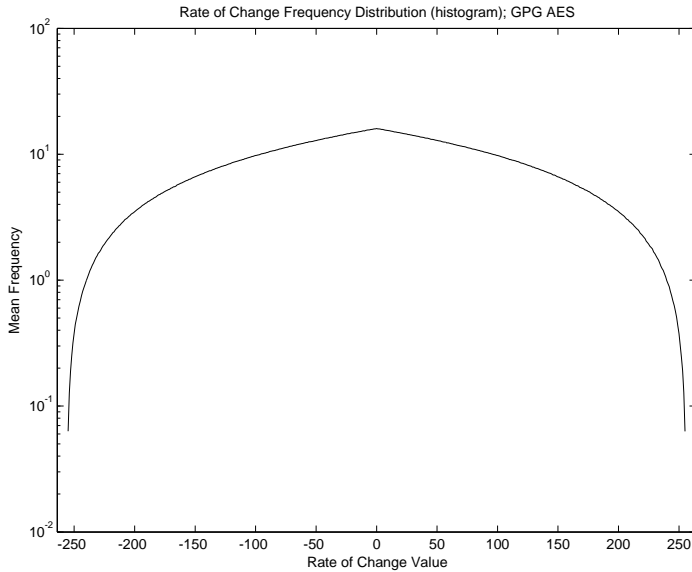
Executable files contain padding where the same byte value is used repeatedly. This can be seen in Figure 2.7 as the peak at RoC value 0. There are also two noticeable large negative RoC values, which are mirrored to a lesser extent on the positive side. The large positive value frequencies are more evenly spread over the spectrum than the corresponding negative values, but otherwise the RoC frequency plot is balanced.

The encrypted file RoC plot in Figure 2.8 shows a perfect bell curve, which also is the expected result. Had there been any disturbances in the plot they had been an indication of a weakness or bug in the implementation of the encryption algorithm.

In Figure 2.9 we can see the large amount of `0xFF00` in a JPEG stream. The plot shows a stream containing RST markers, which are indicated by the small peak starting at -47 and ending at -40. The four peaks at the positive side are

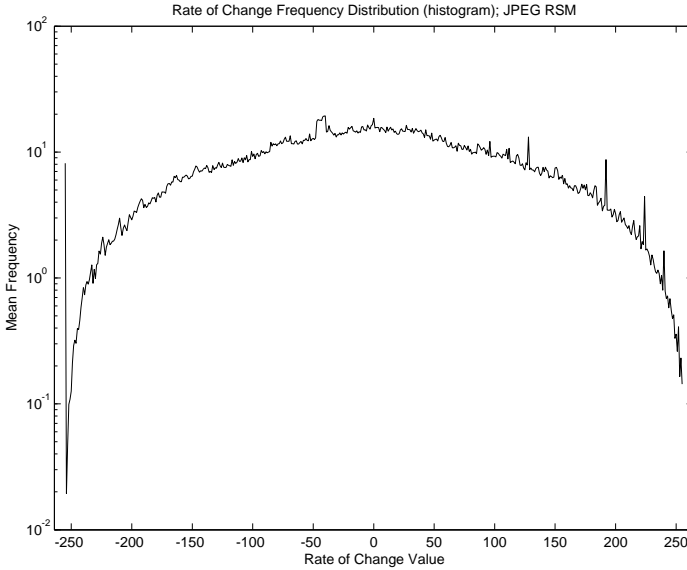


**Figure 2.7:** The frequency distribution of the Rate of Change for a collection of executable Windows PE files. The Y-axis is plotted with a logarithmic scale and the frequency values correspond to 4 KiB of data.



**Figure 2.8:** The frequency distribution of the Rate of Change for a large file encrypted with GPG. The Y-axis is plotted with a logarithmic scale and the frequency values correspond to 4 KiB of data.

### 2.3. RATE OF CHANGE



**Figure 2.9:** The frequency distribution of the Rate of Change for the data part of a collection of JPEG images containing RST markers. The Y-axis is plotted with a logarithmic scale and the frequency values correspond to 4 KiB of data.

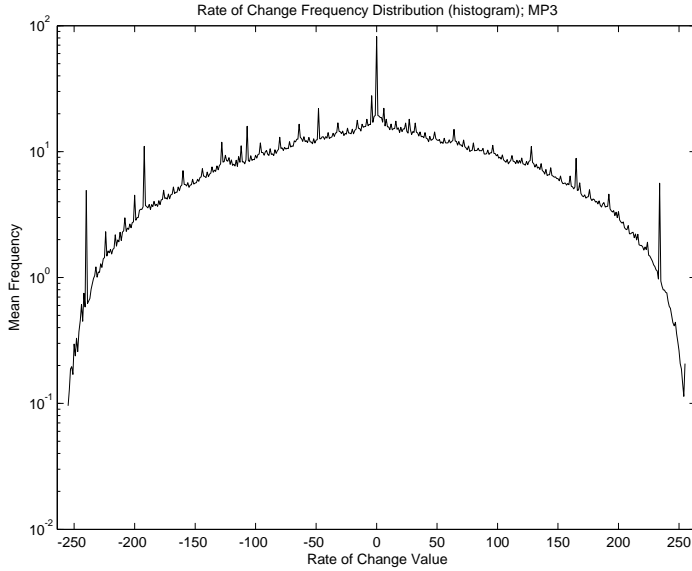
indications of the bit padding of 1s preceding an  $0xFF$  value. They give the preceding byte a limited number of possible values and consequently there are an increased amount of a number of RoC values.

The MP3 centroid in Figure 2.10 has significantly deviating RoC values in positions -240, -192, -107, -48, -4, 0, 6, and 234. By manually checking the centroid we found that the large RoC value at 0 is due to padding of the MP3 stream with  $0x00$  and  $0xFF$ . The value at -192 comes from the hexadecimal byte sequence  $0xC000$ . Another common sequence is  $0xFFFB9060$ , as is the sequence  $0x0006F0$ . The first sequence gives the difference values -4, -107, and -48 and the latter sequence has the difference values 6 and 234. Consequently the noticeable RoC values are generated by a few byte sequences and are not evenly distributed over all possible difference values.

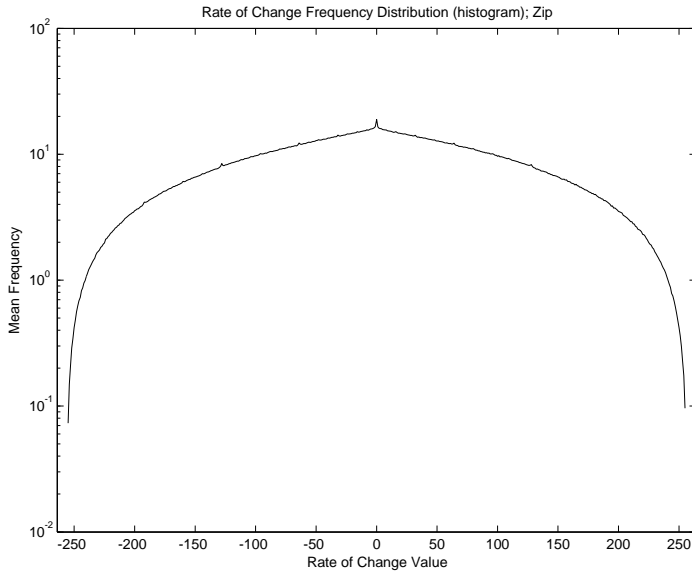
The only noticeable variation in the RoC value plot of Zip files in Figure 2.11 is at position 0. Otherwise the curve is smooth, which is expected since the file type is compressed.

The RoC algorithm is meant to be combined with the BFD algorithm used by the file type categorisation method. It is possible to use the same quadratic distance metric for both algorithms and in that way make the method simpler and easier to control. The combination is made as a logical AND operation, because our idea is to let the detection abilities of both algorithms complement each other, thus cancelling out their individual weaknesses.

Depending on the similarity of the sets of positives of the two algorithms the



**Figure 2.10:** The frequency distribution of the Rate of Change for a collection of MP3 files. The Y-axis is plotted with a logarithmic scale and the frequency values correspond to 4 KiB of data.



**Figure 2.11:** The frequency distribution of the Rate of Change for a collection of compressed Zip files. The Y-axis is plotted with a logarithmic scale and the frequency values correspond to 4 KiB of data.

improvement due to the combination will vary. The detection rate decreases if one of the algorithms has a close to optimal positive set, i.e. few false negatives and false positives, while at the same time the positive set of the other algorithm is less optimal. If we want to prioritise the detection rate we shall use a logical OR operation instead.

## 2.4 2-Grams

The 2-gram algorithm was developed to explore the use of byte pairs for file type categorisation. There are both advantages and disadvantages to using 2-grams. The disadvantages come from the exponential increase of the centroid's size, which mainly affects the execution speed of the algorithm, but also the memory footprint. In theory the effect is a 256-fold increase in process time and memory footprint compared to the 1-gram algorithms.

The increase in size of the memory footprint of the 2-gram algorithm can be ignored due to the amount of RAM used in modern computers; the requirement of the current implementation is a few hundred KiB of RAM. The effect of the increase in execution time of the algorithm can be lowered by optimisation of the code, but the decrease is not bounded.

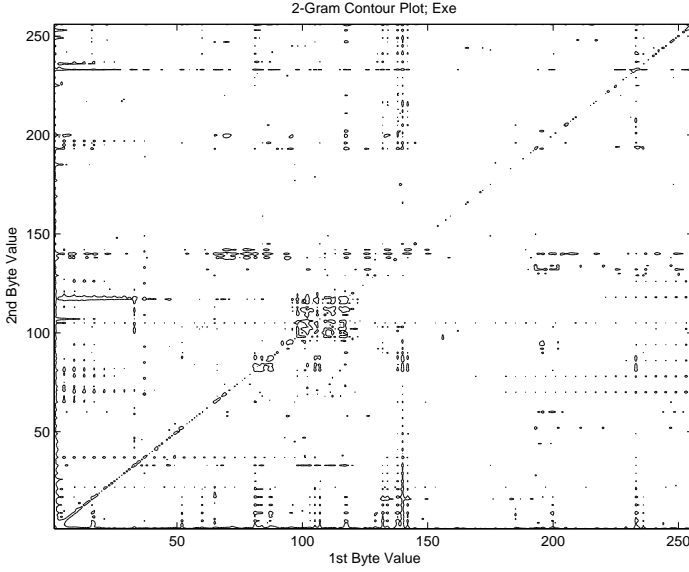
The main advantage of the algorithm is the automatic inclusion of the order of the bytes into the file type categorisation method, signalling that an unknown fragment does not conform to the specification of a specific file type and thus should not be categorised as such. A typical example is the appearance of certain JPEG header byte pairs disallowed in the data part. Another example is special byte pairs never occurring in JPEG files created by a specific camera or software.

The centroid modelling a file type is created by counting the number of unique 2-grams in a large number of 1 MiB blocks of data of the file type to identify. The data in each 1 MiB block are used to form a frequency distribution of all 65536 possible 2-grams and the mean and standard deviation of the frequency of each 2-gram is calculated and stored in two  $256 \times 256$  matrices. The values are then scaled down to correspond to a block size of 4 KiB.

The reason for using 1 MiB data blocks is to get a solid foundation for the mean and standard deviation calculations. When using a block size less than the possible number of unique 2-grams for creation of a centroid the calculations inevitably become unstable, because the low resolution of the incorporated values creates round-off errors.

A typical 2-gram centroid can be seen in Figure 2.12. This particular centroid represents a Windows PE file in the form of a contour plot with two levels. To lessen the impact of large values the height of the contour has been scaled logarithmically. It is possible to see the printable characters, as well as the padding of `0x00` and `0xFF` in the lower left and upper right corners.

When the similarity between a sample fragment and a centroid is measured the same type of quadratic distance metric as for the 1-gram file type categorisation method is used. Equation (2.2) describes the metric, where matrix  $S$  depicts the sample 2-gram frequency distribution and  $C$  the mean value matrix of the



**Figure 2.12:** A contour plot of the frequency distribution of the 2-gram algorithm for a collection of Windows PE files (.exe). The Z-axis is plotted with a logarithmic scale and the frequency values are scaled down to correspond to 4 KiB of data.

centroid. The standard deviation of 2-gram  $ij$  is represented by  $\sigma_{ij}$ . There is also a smoothing factor  $\alpha = 0.000001$ , which is used to avoid division by zero when  $\sigma_{ij} = 0$ .

$$d(S, C) = \sum_{i=0, j=0}^{i=j=255} (s_{ij} - c_{ij})^2 / (\sigma_{ij} + \alpha). \quad (2.2)$$

The reason for not using a more advanced distance metric is execution speed. The current maximum size of a single hard disk is 750 GiB and hence a forensic examination can involve a tremendous amount of data to scan.

## 2.5 Evaluation

To evaluate the detection rate versus the false positives rate of the different algorithms for file type categorisation we use real world data in the form of files collected from standard office computers and the Internet. If necessary the collected files are concatenated end-to-end into a single large file, which is then manually cleaned of foreign data, i.e. sections of data of other file types. The resulting files will be called *cleaned full size files* or simply *full files* throughout the thesis.

The cleaned full files are truncated to the length of the shortest such file, in this case 89 MiB, and used for the evaluation. We will call the evaluation files for *uniformly sized files* or simply *uni-files*.

## 2.5. EVALUATION

The goal of the experiments is to measure the detection rate versus the false positives rate of each of the algorithm and centroid combinations. There are 15 algorithms (variations on the three main algorithms) to be tested on 7 centroids (of 5 main types). The following algorithms are used:

**BFD** The standard Byte Frequency Distribution algorithm with a quadratic distance metric.

**BFD with JPEG rule set** A variant of the standard BFD extended with a JPEG specific rule set.

**RoC** The standard Rate of Change algorithm with a quadratic distance metric and absolute valued rate of changes.

**RoC with JPEG rule set** A variant of the standard RoC algorithm extended with a JPEG specific rule set.

**RoC with Manhattan distance metric** A variant of the standard RoC algorithm where a simple Manhattan (1-norm) distance metric is used.

**RoC with signed difference values** A variant of the standard RoC algorithm using signed values for the differences.

**RoC with signed values and JPEG rule set** A variant of the standard RoC algorithm with signed difference values and extended with a JPEG rule set.

**RoC with signed values and Manhattan distance** A variant of the RoC algorithm using signed difference values and a simple Manhattan distance metric.

**BFD and RoC combined** A combination of the standard BFD and standard RoC algorithms giving the intersection of their detected elements, i.e. performing a logical AND operation.

**BFD and RoC with Manhattan distance metric** A combination of the standard BFD algorithm and the RoC algorithm using a Manhattan distance metric.

**BFD and RoC with JPEG rule set** A combination of the BFD and RoC algorithms using a JPEG specific rule set.

**BFD and RoC with signed values** A combination of the standard BFD algorithm and the RoC algorithm using signed difference values.

**BFD and RoC with signed values and JPEG rule set** The BFD algorithm in combination with the RoC algorithm using signed difference values and a JPEG specific rule set.

**BFD and RoC with signed values and Manhattan distance** A combination of the standard BFD algorithm and a RoC algorithm using signed difference values and a Manhattan distance metric.

**2-gram** The 2-gram (byte pair) frequency distribution with a quadratic distance metric.

The algorithm variants are combined with centroids for the following file types. The creation of each centroid and its underlying files is described in the coming subsections.

- Windows PE files
- Zip files
- AES encrypted file using GPG
- CAST5 encrypted file using GPG
- JPEG image data parts with RST markers
- JPEG image data parts without RST markers
- MP3 audio files without an ID3 tag

The reason for using two different encryption algorithms is the fact that GPG uses packet lengths of 8191 bytes when using the older CAST5 algorithm. This results in the `0xED` value being repeated every 8192 bytes, giving rise to a clearly visible deviation in the centroid, which can be seen in Figure 2.13. This marker is also used for all available algorithms in GPG when it is possible to compress the source file, or the size of the final encrypted file is not known in advance [20, pp. 14–16].

A higher amount of `0xED` in an encrypted file therefore indicates

1. an older algorithm being used, or
2. a compressible source file type, and
3. that GPG does not know the length of the encrypted file in advance.

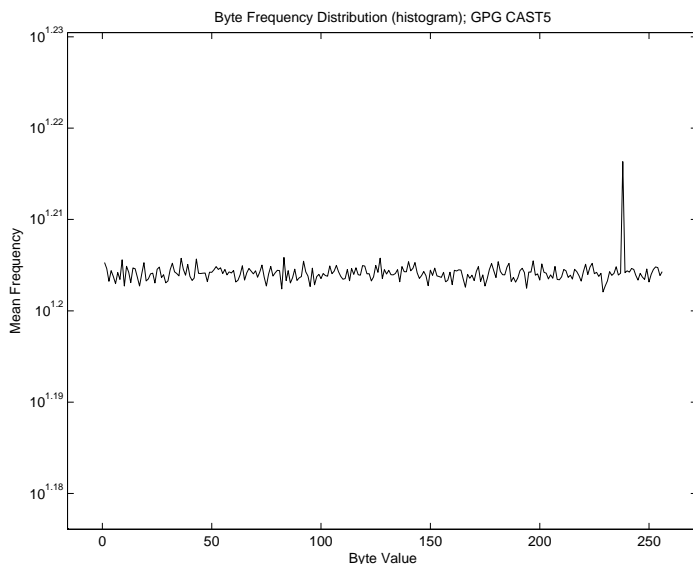
Case 1 becomes trivial if the header is available and is therefore less important, but cases 2 and 3 means that some information is leaking, although not very much.

The JPEG images are divided into two groups according to whether they come from cameras or applications using RST markers or not. This partitioning is made to test if it is possible to enhance the detection rate for such images. The RST markers are frequent and easy to detect, so there is no problem creating clean image data to train and test on.

The files used for the evaluation are truncated to form equally sized data sets. We do this to eliminate the risk of affecting the evaluation results by having centroids based on different amounts of data. The truncation method does not give rise to artificial byte combinations, which a method where non-consecutive segments of a large file are joined together will do. This is important when the algorithm is taking the ordering of the bytes into consideration.



## 2.5. EVALUATION



**Figure 2.13:** The byte frequency distribution of a large file encrypted with GPG, using the CAST5 algorithm. A logarithmic scale is used for the Y-axis and the frequency values correspond to 4 KiB of data.

The downside of using a fixed amount of data from the first part of a large file is that the composition of the data in the excluded part might differ from the included part and in that way bias the centroid. When creating the full files, which are used to make the uni-files, we try to make an evenly distributed mix of the included files and to concatenate them in an acceptable order.

The evaluation of the detection rate uses half of the data from a uni-file for creating a centroid and the other half to test on. The halves are then swapped and the evaluation process is repeated. The reason for this is to avoid testing on the same data as was used to create the centroid. Performing the evaluation twice on different parts of the data also guarantees that the same amount of data is used for evaluation of the detection rate and the false positives rate.

Unfortunately the splitting of files, giving a lower amount of data for centroid creation, affects the quality of the centroids. For file types with an almost uniform byte frequency distribution the decrease in quality is noticeable.

### 2.5.1 Microsoft Windows PE files

We use 219 files<sup>2</sup> of Microsoft Windows PE format extracted from a computer running Windows XP SP2. The *file* utility of Linux categorises the files into three main groups:

---

<sup>2</sup>The raw data files and source code can be found at <http://www.ida.liu.se/~iislab/security/forensics/material/> or by contacting the author at [g-makar@ida.liu.se](mailto:g-makar@ida.liu.se)

- MS-DOS executable PE for MS Windows (console) Intel 80386 32-bit
- MS-DOS executable PE for MS Windows (GUI) Intel 80386
- MS-DOS executable PE for MS Windows (GUI) Intel 80386 32-bit

The executable files are concatenated by issuing the command

```
cat *.exe > exes_new.lst
```

which treats the files in alphabetical order, no case-sensitivity. The resulting `exes_new.lst` file is manually cleaned from JPEG, GIF, and PNG images by searching for their magic numbers, keeping track of the look of the byte stream and ending the search when an end marker is found. We check for embedded thumbnail images to avoid ending our cleaning operation prematurely. The GIF images are found by their starting strings “GIF87a” or “GIF89a”. They end in `0x003B`, which can also occur within the image data, consequently the correct file end can be hard to find. We use common sense, there are often a number of zeros around the hexadecimal sequence when it represents the end of a GIF image. PNG images start with the hexadecimal byte sequence `0x89504E470D0A1A0A` and are ended by the string “IEND”.

The cleaned full file is called `exes_clean_full.raw` and the final step is to truncate the full file to form a 89 MiB long file. The new file is called `exes_clean_new.uni` and is used for the evaluation.

### 2.5.2 Encrypted files

We create two encrypted files<sup>3</sup> from the Zip full file using two symmetric encryption algorithms, CAST5 and AES with 128 bit keys. Due to the discovery of the `0xED` feature in GPG, the CAST5 file is given a name related to its source file, not the encryption algorithm name. The framework we use is GPG 1.4.6, which is called by

```
gpg --cipher-algo=cast5 \
--passphrase=gpg_zips_full.raw \
--output=gpg_zips_full.raw -c zips_new_full.raw
```

and

```
gpg --cipher-algo=aes --passphrase=gpg_zips_full.raw \
--output=gpg_aes_full.raw -c zips_new_full.raw
```

The full file of the Zip type is used because it is not further compressible and consequently the `0xED` packet length marker is not used for AES or other modern algorithms<sup>4</sup>.

<sup>3</sup>The raw data files and source code can be found at <http://www.ida.liu.se/~iislab/security/forensics/material/> or by contacting the author at [g-makar@ida.liu.se](mailto:g-makar@ida.liu.se)

<sup>4</sup>The modern algorithms available in our version of the GPG software are AES, AES192, AES256, and Twofish.

The encrypted full files are truncated to form two 89 MiB long files called `gpg_aes_new.uni` and `gpg_zips_new.uni`. These are then used for the evaluation.

### 2.5.3 JPEG files

We create two centroids for the JPEG file type, one for images using RST markers and one for images without RST markers<sup>5</sup>. The data files are named accordingly, `jpegs_rsm_full.raw` and `jpegs_no_rsm_full.raw`.

The images contained in the source files used for our research are taken from our database of standard, private (amateur) JPEG images captured using consumer grade cameras and in typical private life conditions. This selection of images is meant to give a broad and well balanced spectrum of images to work with. We have not excluded any technically poor images, because our intention is to keep the evaluation as close as we can to a real-life situation, without sacrificing control of the test environment.

All image donors have approved the selection of images and given us oral permission to use them in our research. The fact that we only use the data part of the images, i.e. the header is stripped off, makes it harder to recreate the images, or in other ways identify any objects in the images.

The `jpeg_rsm_full.raw` file consists of 343 images stripped of their headers, together giving 466 MiB of data. In Table 2.1 the number of images for each camera make, model and image size included in the file is shown. Images of portrait and landscape orientation are shown on separate lines, because their image data is differently oriented. This should not affect the encoding at a low level (byte level in the files), but at the pixel level it might matter.

The `jpeg_rsm_full.raw` file is made up of 272 images giving a total of 209 MiB. The images in this file do not contain any RST markers and come from the camera makes and models given in Table 2.2.

The two full files are truncated to form two equally large (89 MiB) uni-files, called `jpegs_rsm_new.uni` and `jpegs_no_rsm_new.uni`. These uni-files are used in the evaluation.

The compression levels of the image files differ, which affects the centroid's ability to correctly model a generic JPEG image. The compression level depends on the camera settings when the photograph was taken. A lower compression level means that the level of detail retained in the image files are higher and there are longer sequences of image code between each RST marker, if used. Longer sequences slightly changes the byte frequency distribution, hence the centroid is affected. We have not investigated to what extent the compression level affects the centroid, but our experience is that the effect is negligible.

---

<sup>5</sup>The raw data files and source code can be found at <http://www.ida.liu.se/~iislab/security/forensics/material/> or by contacting the author at [g-makar@ida.liu.se](mailto:g-makar@ida.liu.se)

**Table 2.1:** The number of images coming from different camera makes and models, and their sizes in pixels. Portrait and landscape mode images are separated. All images contain RST markers.

# images	Camera	Image size
25	Canon DIGITAL IXUS 400	1600x1200
25	Canon DIGITAL IXUS v	1600x1200
25	Canon EOS 350D DIGITAL	3456x2304
25	Canon EOS 400D DIGITAL	3888x2592
4	Canon PowerShot A70	1536x2048
46	Canon PowerShot A70	2048x1536
25	CASIO COMPUTER CO.,LTD EX-Z40	2304x1728
25	Kodak CLAS Digital Film Scanner / HR200	1536x1024
24	Konica Digital Camera KD-310Z	1600x1200
1	Konica Digital Camera KD-310Z	2048x1536
11	KONICA MINOLTA DiMAGE G400	1704x2272
14	KONICA MINOLTA DiMAGE G400	2272x1704
25	NIKON D50	3008x2000
6	NIKON E3500	1536x2048
19	NIKON E3500	2048x1536
3	Panasonic DMC-FZ7	2112x2816
22	Panasonic DMC-FZ7	2816x2112
18	SONY DSC-P8	2048x1536

**Table 2.2:** The number of images coming from different camera makes and models, and their sizes in pixels. Portrait and landscape mode images are separated. There are no RST markers in the images.

# images	Camera	Image size
73	FUJIFILM FinePix2400Zoom	1280x960
9	FUJIFILM FinePix2400Zoom	640x480
95	FUJIFILM FinePix E550	2848x2136
73	OLYMPUS IMAGING CORP. uD600,S600	1600x1200
2	1 OLYMPUS IMAGING CORP. uD600,S600	2816x2112
1	OLYMPUS IMAGING CORP. uD600,S600	640x480

**Table 2.3:** The MP3 files included in `mp3_no_id3_full.raw`. The individual encoding parameters are shown for each file.

File name	Bitrate	Encoder
01_petra_ostergren_8A6F_no_id3.mp3	variable	iTunes v7.1.1.5
02_david_eberhard_192DE_no_id3.mp3	variable	iTunes v7.1.1.5
03_boris_benulic_30DF_no_id3.mp3	variable	iTunes v7.1.1.5
04_dick_kling_81F0_no_id3.mp3	variable	iTunes v7.1.1.5
05_maria_ludvigsson_3A96_no_id3.mp3	variable	iTunes v7.2.0.34
06_hakan_tribell_C59F_no_id3.mp3	variable	iTunes v7.1.1.5
07_anders_johnson_0558_no_id3.mp3	variable	iTunes v7.1.1.5
08_marie_soderqvist_4B92_no_id3.mp3	variable	iTunes v7.1.1.5
09_erik_zsiga_D020_no_id3.mp3	128 KiB/s	-
10_carl_rudbeck_F7CB_no_id3.mp3	variable	iTunes v7.1.1.5
11_nuri_kino_7CFA_no_id3.mp3	112 KiB/s	-
12_johan_norberg_4F96_no_id3.mp3	112 KiB/s	-

#### 2.5.4 MP3 files

We use 12 MP3 files<sup>6</sup> featuring a pod radio show in the evaluation. The files contain ID3 tags [21], which are deleted using the Linux tool `id3v2 0.1.11`. We use a script to clean the files and then check each file manually to verify the cleaning. The resulting files are then concatenated and saved in a file called `mp3_no_id3_full.raw`, in the order shown in Table 2.3. The resulting full file is truncated to a size of 89 MiB and called `mp3_no_id3_new.uni`. All included files are MP3 version 1 files. They are all sampled at 44.1 kHz as Joint Stereo. Their individual encoding can be seen in Table 2.3.

The reason for deleting the ID3 tags is that they contain extra information. There is for example a field for attaching pictures [22, Sect. 4.15]. It is even possible to attach any type of data using a label called “General encapsulated object” [22, Sect. 4.16], hence a MP3 file can contain executable code. Since we want to use nothing but the audio stream in our evaluation we have to delete the ID3 tag.

#### 2.5.5 Zip files

The file `zips_new_full.raw`, consists of 198 zipped text files<sup>7</sup> from the Gutenberg project [23]. Among them is a zipped version of the Swedish bible from 1917 in the form a 1.5 MiB compressed file. All files from the Gutenberg project are compressed with Zip version 2 or above. The text in the files is ISO-8859-1 encoded. We also include a zipped `.iso` (ReactOS-0.3.1-REL-live.zip) and

<sup>6</sup>The raw data files and source code can be found at <http://www.ida.liu.se/~iislab/security/forensics/material/> or by contacting the author at [g-makar@ida.liu.se](mailto:g-makar@ida.liu.se)

<sup>7</sup>The raw data files and source code can be found at <http://www.ida.liu.se/~iislab/security/forensics/material/> or by contacting the author at [g-makar@ida.liu.se](mailto:g-makar@ida.liu.se)

a zipped Preboot Execution Environment (PXE) network XWin edition of the Recovery Is Possible (RIP) 3.4 Linux rescue system (zipped with a software version  $\geq 1.0$ ). The files are concatenated, sorted by file name in ascending order with the text files first, then the ReactOS, RIP, and finally a zipped version of the `exes_clean_full.raw` file. The full file is added to extend the amount of data for the creation of the encrypted file centroid (see Section 2.5.2). The original executable file collection is 89 MiB unzipped and 41 MiB zipped and is compressed using Linux Zip 2.32.

The `zips_new_full.raw` file is truncated to a 89 MiB long file, which is used for the evaluation. The new file is called `zips_new_new.uni`.

### 2.5.6 Algorithms

We test the amount of false positives and the detection ability of each algorithm by creating a centroid for each file type and then measuring the distance between the centroid and every fragment of each file type. The uni-files are used as test files, the distances are sorted and the test file giving the lowest distance is recorded for each combination of fragment, centroid and algorithm.

When measuring the detection rate of the centroids, i.e. having them detect fragments of their own type, we partition the uni-files in two and use one for training and the other for testing. We perform each test twice, swapping the roles of the file parts.

Because of the quality decrease of centroids based on smaller amounts of data, especially for almost uniformly distributed file types (see Section 2.6.6), we make a small extra evaluation using an alternative data source for the encrypted file type centroids. The data comes from a 644 MiB large CD iso file, which is downloaded from one of the Debian mirror sites [24]. The file is compressed using Zip and then two encrypted files are created in same way as described in Section 2.5.2, one using AES and one using CAST5. The results are then used to create a confusion matrix.

## 2.6 Results

The results of the evaluation will be presented using Receiver Operating Characteristic (ROC) curves [25]. A ROC curve plots the true positives against the false positives while the detection threshold is varied.

There is little meaning in plotting detection rate values below 50%, or false positives values above 50%, because outside those boundaries the results are getting close to guessing. Strictly speaking that happens when the ROC curve falls below the diagonal where the false positives rate equals the detection rate. Therefore we have limited the plots to the upper left quarter of the ROC curve plotting range area, although the consequence might be that some results are not shown

## 2.6. RESULTS

**Table 2.4:** The following base names, possibly together with a prefix or suffix, are used in the evaluation. The corresponding algorithm names are given as a translation.

Base name	Algorithm
2-gram_ny	2-gram
map-bf	BFD
map-bf_jpeg_spec	BFD with JPEG rule set
map-bf_n_roc	BFD and RoC combined
map-bf_n_roc_jpeg_spec	BFD and RoC with JPEG rule set
map-bf_n_roc_norm	BFD and RoC with Manhattan distance metric
map-roc	RoC
map-roc_jpeg_spec	RoC with JPEG rule set
map-roc_norm	RoC with Manhattan distance metric
map_noabs-bf_n_roc	BFD and RoC with signed values
map_noabs-bf_n_roc_jpeg_spec	BFD and RoC with signed values and JPEG rule set
map_noabs-bf_n_roc_norm	BFD and RoC with signed values and Manhattan distance
map_noabs-roc	RoC with signed difference values
map_noabs-roc_jpeg_spec	RoC with signed values and JPEG rule set
map_noabs-roc_norm	RoC with signed values and Manhattan distance

in the figures. What we miss are the *conservative*<sup>8</sup> and *liberal*<sup>9</sup> parts of the curves in the graphs.

The algorithm ROC curves for each centroid used in the evaluation will be presented in the following sections, one for each file type. We choose to present the results from the centroid point of view because in that way the algorithms can easily be compared against each other. The legend of each ROC curve plot contains algorithm base names, which are given together with the corresponding algorithm name in Table 2.4.

The file type categorisation method and its algorithms are generalisable, which the evaluation shows. It is applicable to different types of compressed or encrypted files, which have a high amount of entropy. Information entropy is a measure of the uncertainty of a signal or stream of bytes. It can be calculated using Equation (2.3), where  $X$  is a discrete random variable existing in the set  $\Omega$ , with the probability function  $f(x)$ . The symbol “ $^2\log$ ” means the base 2

<sup>8</sup>Defined as giving few positives, neither true nor false. They are conservative in the sense that they need strong evidence to give a positive categorisation. [25]

<sup>9</sup>Defined as giving many positives, both true and false. They are liberal in the sense that they need only weak evidence to give a positive categorisation. [25]

**Table 2.5:** The information entropy values for some file types.

File type	Entropy [bits]
Windows PE	6.581
MP3	7.938
JPEG w. RST	7.971
Zip	7.998
GPG AES	8.000

logarithm, which gives  $H(X)$  the unit *bit*. [26]

$$H(X) = - \sum_{x \in \Omega} f(x) (\log f(x)) \quad (2.3)$$

The entropy values for some of the file types included in the evaluation can be found in Table 2.5. They are calculated using Equation (2.3) and the values for the mean byte frequency of the centroids. The executable file differs from the other files by its approximately 17% lower value, but all values are close to the size of a byte, rendering their byte streams nearly random in appearance.

### 2.6.1 Microsoft Windows PE files

The only algorithm able to properly detect Windows PE files is the 2-gram algorithm. The ROC curve can be seen in Figure 2.14 and starts to level out at 60% detection rate and a false positives rate of 0.05%. It then slowly increases towards a new plateau at 75% detection rate and 10% false positives rate.

The reason for the non-smooth curve is probably the fact that executable files can be divided into several sub-parts, where the main parts are machine code and tables. Tables are very structured and easy to categorise manually, while machine code looks more random and unstructured. But machine code contain specific byte pairs that are more frequent than others, and that is detected by the 2-gram algorithm. The tables does not necessarily have a small number of more frequent byte pairs, and hence is not detected until the detection threshold is considerably higher than for machine code. We have not confirmed this experimentally, we have yet only formed a hypothesis.

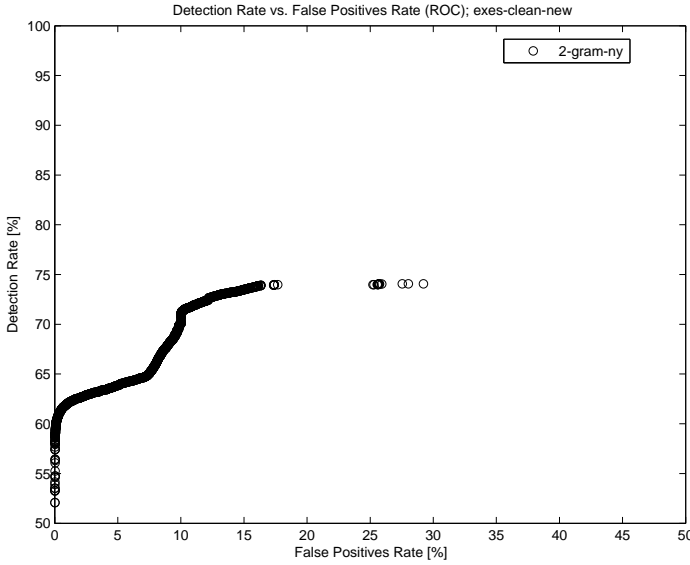
### 2.6.2 Encrypted files

We use two different centroids for the encrypted file experiments, because we found a deviation of the frequency of byte 0xED in some cases (see Section 2.5 and Figure 2.13). This deviation is too small to be seen in the ROC curves without a high magnification. Therefore we only show the results for the AES encrypted file centroid.

Figure 2.15 shows the results, where four clusters of algorithms are visible. At the 50% detection rate level the clusters are, from left to right:



## 2.6. RESULTS



**Figure 2.14:** The results for the Windows PE files centroid for algorithms scoring above 50% detection rate and below 50% false positives rate.

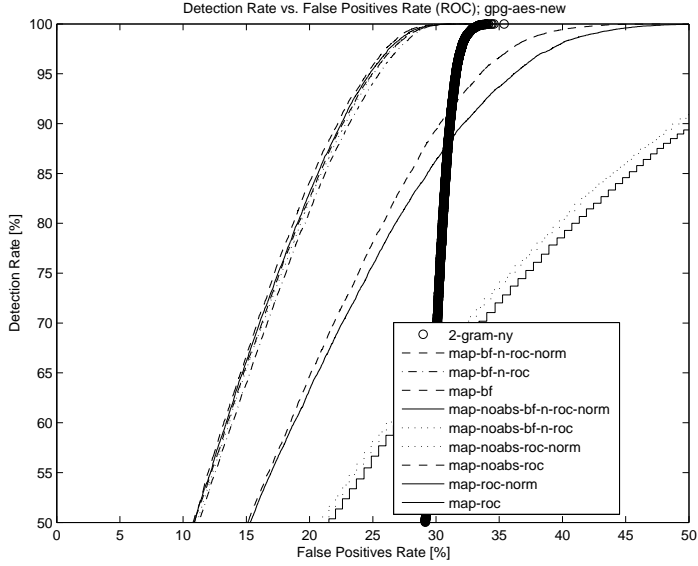
1. BFD, and BFD and RoC, at 11% false positives.
2. RoC with a quadratic distance metric, at 15% false positives.
3. RoC with a Manhattan distance metric, at 21% false positives.
4. 2-gram, at 29% false positives.

As can be seen in Figure 2.15 the 2-gram algorithm has the steepest inclination, but the best algorithms are still the BFD, and the BFD and RoC combination, reaching 100% detection rate at between 31.4% and 34.9% false positives. The 2-gram algorithm is a close second (or maybe more correctly, a close sixth) at 35.4% false positives.

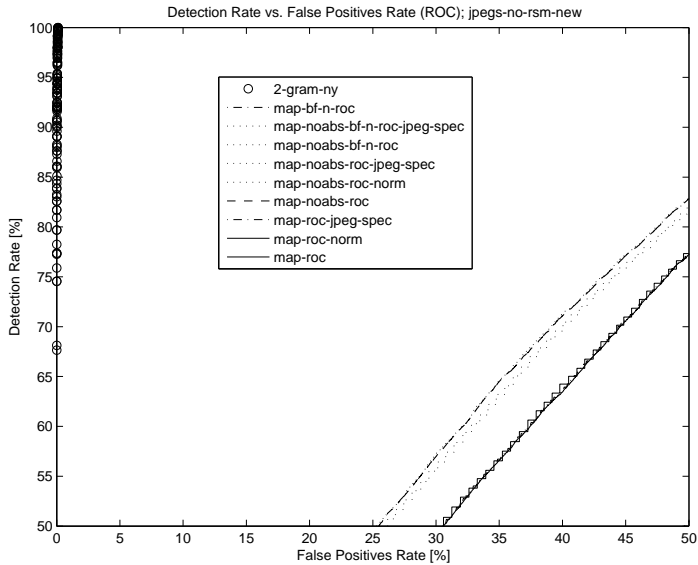
The difference in results between the centroids for the AES encrypted file and the CAST5 encrypted file is greatest at 50% detection rate. The best algorithm is the BFD in both cases, for CAST5 the false positives rate is 10.5% and for AES it is 10.8%.

### 2.6.3 JPEG files

The ROC curve for JPEG files without RST markers in Figure 2.16 is clustered into three curve areas, with a fourth area just touching the bottom right corner. The fourth cluster is not visible in the plot, but is there. It has a false positives rate of approximately 49.7%.

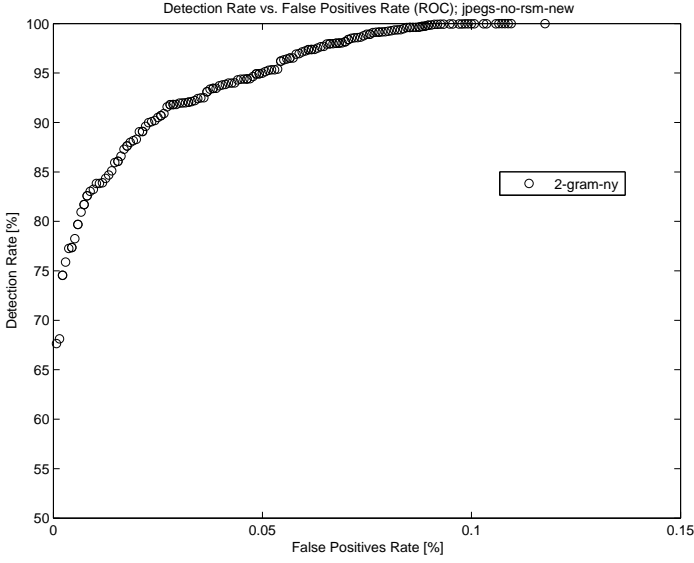


**Figure 2.15:** The results for the AES encrypted file centroid for algorithms scoring above 50% detection rate and below 50% false positives rate.



**Figure 2.16:** The results for the centroid for JPEG without RST markers for algorithms scoring above 50% detection rate and below 50% false positives rate.

## 2.6. RESULTS



**Figure 2.17:** The results for the centroid for JPEG without RST markers for the 2-gram algorithm.

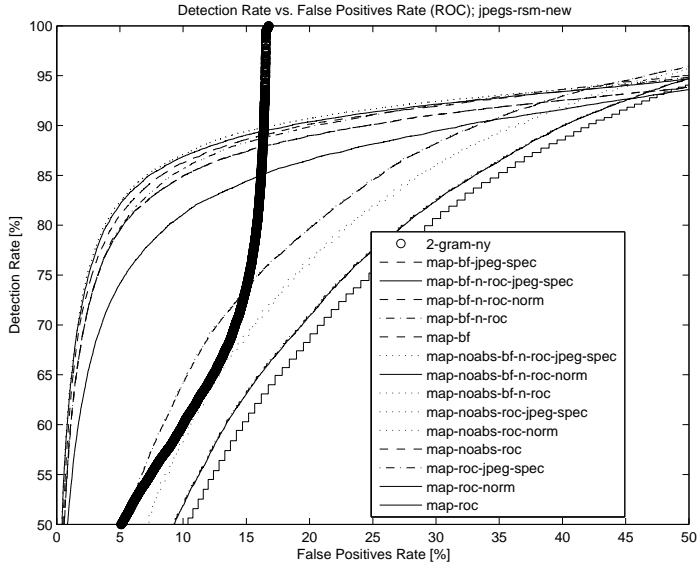
The clustered areas are related to the following algorithms, with the given false positives rate at a detection rate of 50%, counting from left to right:

1. 2-gram, false positives rate 0%
2. RoC with signed difference values, false positives rate 26%
3. RoC with absolute difference values, false positives rate 31%
4. BFD and RoC with quadratic distance metric, false positives rate 50%

The BFD and RoC combinations with a simple Manhattan distance metric, as well as the single BFD algorithm, give false positives rates higher than 50% and are not shown in the figure.

The 2-gram algorithm gives outstanding results compared to the other algorithms. The centroid for JPEG images lacking RST markers used together with the 2-gram algorithm give a 100% detection rate at a false positives rate of less than 0.12%, as can be seen in Figure 2.17. For detection rates below 67.6% there are no false positives. Please observe the logarithmic scale of the (false positives) X-axis.

The centroid for JPEG images with RST markers, which can be seen in Figure 2.18, does not give results as good as the previous JPEG centroid. The best results are achieved together with the BFD and RoC combinations, and the single BFD algorithms. All these algorithms give a false positives rate of less than 1% at a detection rate of 50%. Best, from a false positives rate point of view, is



**Figure 2.18:** The results for the centroid for JPEG with RST markers for algorithms scoring above 50% detection rate and below 50% false positives rate.

the BFD and RoC combination using signed difference values, a quadratic distance metric, and a JPEG specific rule set. This algorithm reaches a detection rate of 83.8% at a false detection rate of 6%. It then levels out and eventually reach a 100% detection rate at 95.7% false positives rate. The 2-gram algorithm is the first to reach a detection rate of 100% at a false positives rate of 16.8%. It starts at a false positives rate of 5.1% at 50% detection rate.

The clusters of algorithms that can be seen in Figure 2.18 comprise the following algorithms, from left to right at 80% detection rate:

1. BFD and RoC with signed difference values, absolute valued BFD an RoC (except for the JPEG specific rule set algorithm), and the BFD algorithms, 4% false positives.
2. BFD and RoC with absolute difference values and a JPEG specific rule set, 8% false positives.
3. 2-gram algorithm, 16% false positives.
4. RoC with signed difference values and quadratic distance metric, 20% false positives.
5. RoC with signed difference values and Manhattan distance metric, 23% false positives.
6. RoC with absolute difference values and quadratic distance metric, 27% false positives.

## 2.6. RESULTS

7. RoC with absolute difference values and Manhattan distance metric, 29% false positives.

Since the centroid for JPEG images with RST markers does not require, only allow, RST markers in a data fragment, it is more prone to false positives than the other JPEG centroid. This explains the differences in the results for the 2-gram algorithm, which is especially good at utilising fixed parameters that are required by a file type.

### 2.6.4 MP3 files

The results for the MP3 file centroid are shown in Figure 2.19. The 2-gram algorithm is the best and is accompanied by three clusters at 85% detection rate. The clusters are, from left to right:

1. 2-gram, 0.1% false positives.
2. BFD and RoC combinations, the signed difference valued slightly to the right, 2% false positives.
3. RoC, 7% false positives.
4. BFD, 10% false positives.

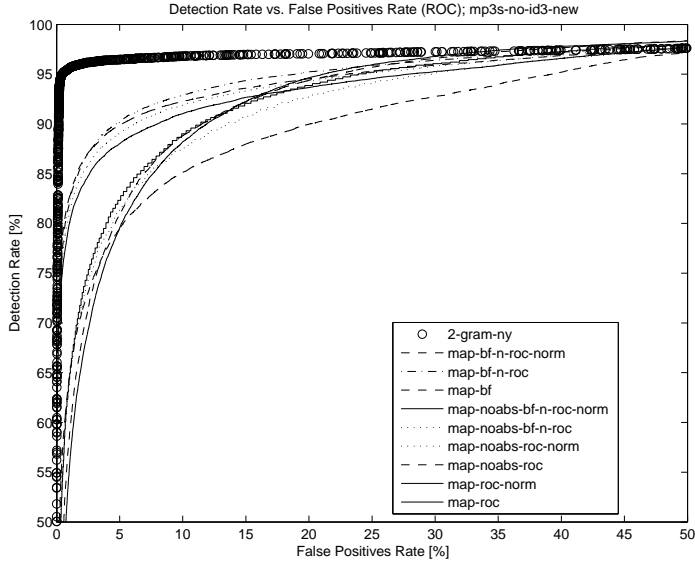
Even though the RoC algorithm with a quadratic distance metric is above the 2-gram algorithm curve when reaching false positive levels of 40% or higher in Figure 2.19, the 2-gram algorithm is the first to reach a 100% detection rate, with a false positives rate of 86%. The second best is actually the BFD algorithm, but it reaches 100% detection at 99.3% false positives.

The detailed results, presented for a maximum false positives rate of 0.5%, can be seen in Figure 2.20. The result of the 2-gram algorithm is levelling out at a detection rate of 94% at 0.24% false positives. The BFD and RoC combinations level out at approximately 70% detection rate, with the BFD and RoC combination using a Manhattan distance metric a few percent above the other three.

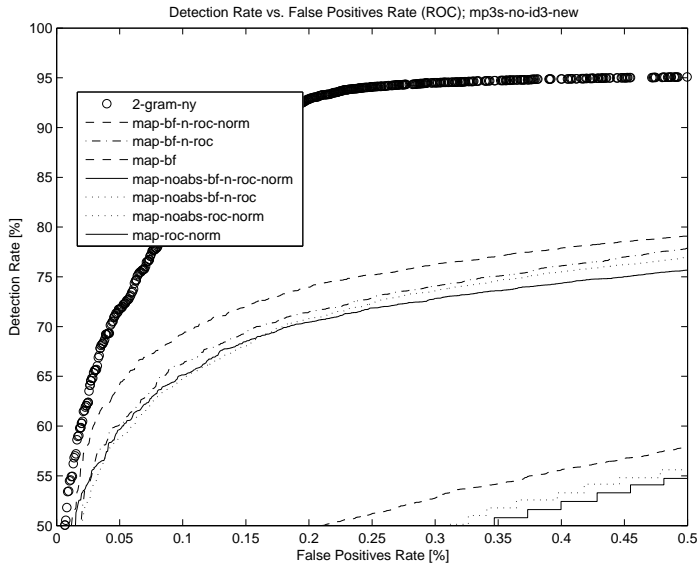
### 2.6.5 Zip files

In Figure 2.21 the results of the Zip file centroid can be seen. The straighter curves belong to the RoC algorithms and the S-shaped curves belong to the BFD and RoC combination algorithms, plus the single BFD algorithm. The clusters of curves shown in the figure belong to, from left to right at the 65% detection rate level, the following algorithms:

1. RoC with quadratic distance metric and signed difference values, 29.8% false positives rate.
2. RoC with quadratic distance metric and absolute difference values, 31.9% false positives rate.

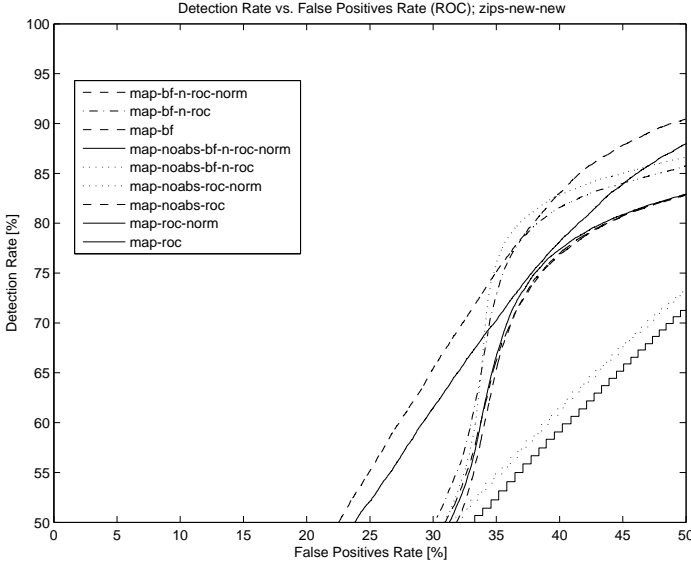


**Figure 2.19:** The results for the MP3 file centroid for algorithms scoring above 50% detection rate and below 50% false positives rate.



**Figure 2.20:** The results for the MP3 file centroid for algorithms scoring above 50% detection rate and below 0.5% false positives rate.

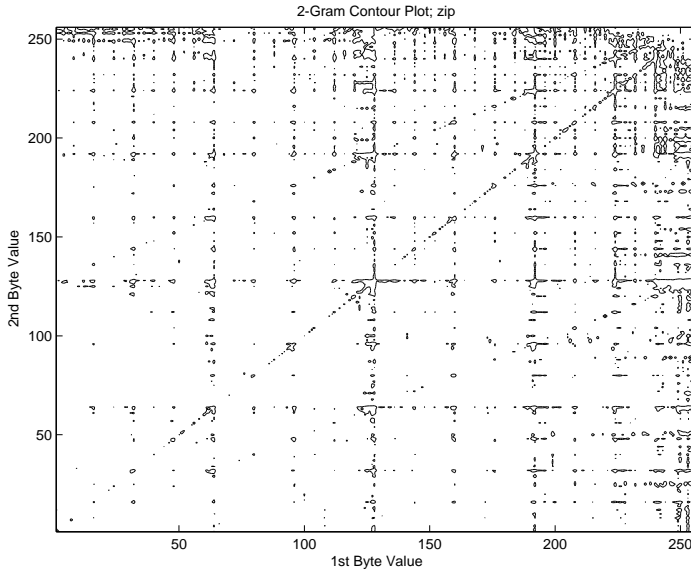
## 2.6. RESULTS



**Figure 2.21:** The results for the Zip files centroid for algorithms scoring above 50% detection rate and below 50% false positives rate.

3. BFD and RoC combinations with quadratic distance metric, 33.8% false positives rate.
4. BFD, and BFD and RoC combinations with Manhattan distance metric, 35% false positives rate.
5. RoC with Manhattan distance metric and signed difference values, 42.9% false positives rate.
6. RoC with Manhattan distance metric and absolute difference values, 44.5% false positives rate.

Worth noticing is that the 2-gram algorithm is not included in Figure 2.21. At a detection rate of 50% it has 68% false positives, and reaches 100% detection at 99% false positives. This might depend on the regular patterns, which makes the centroid resemble a structured file type, in an otherwise almost uniform distribution. The regular patterns can be seen in Figure 2.22. The figure shows that the 2-gram frequencies are slowly increasing towards higher byte pairs. There is one line corresponding to similarly valued 2-grams, and two lines representing byte pairs where one of the byte values is twice the value of the other, i.e.  $x = 2y$  or  $2x = y$ . There is also a pattern of squares of bytes either starting or ending with  $0xF$ .



**Figure 2.22:** A contour plot for a Zip file centroid of 2-grams. The almost evenly distributed centroid has clearly visible patterns.

### 2.6.6 Algorithms

One way to represent the performance of a categorisation algorithm is to use a confusion matrix. In our case the matrix shows the number of fragments categorised as different file types when using a specific centroid. Hence the rows represent different centroids, the actual classes, and the columns represent categorisations by the algorithms, meaning the predicted classes.

The short form of the centroid names used in the confusion matrices can be found in Table 2.6.

The best results are achieved using the 2-gram algorithm, but in some of our tests other algorithms performed better than the 2-gram algorithm. Table 2.7 presents the confusion matrix of the 2-gram algorithm. The confusion matrices of the rest of the algorithms can be found in Appendix C.

As can be seen the two centroids representing encrypted information are better at categorising each other than themselves. This phenomena can be traced back to the smaller data sets used to create the centroids for the detection rate evaluation (see Section 2.5). A smaller data sub-set (of a larger data set) gives a standard deviation less or equal to the standard deviation of the full data set. Therefore the distance metric used (see Equation (2.2)) gives slightly higher distance values for the sub-sets than for the full data set.

The variations in the data between the encrypted file types are smaller than the difference in the mean standard deviation between the full data and the sub-set data centroids. Consequently the full data set centroids are better at cate-



## 2.6. RESULTS

**Table 2.6:** The following short names are used for the confusion matrices.

Short name	Centroid
exe	Windows PE files
cast5	CAST5 encrypted GPG file
aes	AES encrypted GPG file
no rst	JPEG images without RST markers
rst	JPEG images with RST markers
mp3	MP3 audio files without an ID3 tag
zip	Files compressed using Zip

**Table 2.7:** The confusion matrix of the 2-gram algorithm.

	exe	aes	cast5	no rst	rst	mp3	zip
exe	15332	8	10	12	15	7247	56
aes	100	402	16450	3	0	0	5725
cast5	107	16614	339	0	0	0	5620
no rst	10	0	0	22667	0	0	3
rst	0	0	0	9677	13003	0	0
mp3	60	371	276	3	6	21882	82
zip	113	5699	5650	4	0	0	11214

gorising almost-similar data types than the small centroids are at categorising their own data type. The mean standard deviations of the four sub-set centroids are approximately 1.5% lower than the mean standard deviations of the full data set centroids, thus increasing the distance measurements by the same amount for the sub-set centroids.

The effect of the smaller amount of data on the detection rate is only noticeable for the encrypted file types, and then only between different encryption algorithms. The small extra amount of 0xED in the CAST5 encrypted file is hidden by the difference in the mean standard deviation values between the full data set and the sub-set centroids.

To verify that a centroid created from a larger data set would improve the detection rate of the centroid's file type, we performed a limited test where we used a 604 MiB compressed file to create two GPG centroids, one using the AES encryption algorithm and one using CAST5. The testing files were the same as we used in the evaluation (see Section 2.5.2). The resulting confusion matrix can be seen in Table 2.8

The centroids created using the 604 MiB file have a mean standard deviation approximately 1.2% higher than the centroids created from the 94 MiB `zips_new_new.uni` file. As can be seen when comparing Table 2.7 and Table 2.8 the detection ability of the AES and CAST5 centroids has increased significantly. The AES centroid now actually detects its own file type better than

**Table 2.8:** The confusion matrix of the 2-gram algorithm using a 604 MiB compressed file to create the GPG AES and GPG CAST5 centroids.

	exe	aes	cast5	no rst	rst	mp3	zip
aes	56	9720	9609	0	0	0	3295
cast5	55	9682	9641	0	0	0	3302

any other, although the difference to the CAST5 file type is only 111 fragments, or 1.2%. The amount of false positives for all file types included in the test have decreased, with a 45% decrease for exe files and a 42% decrease for the zip files.

## Chapter 3

# Putting Fragments Together

When a number of data fragments of a specific file type have successfully been recovered using the file type categorisation method, the next step is to try to reconnect them in the right order to recreate the original file. This chapter will present a method to do that for JPEG images having RST markers. The basic idea behind the JPEG image fragment reconnection method is outlined in Section 1.5.

The term *fragment joint validity* will be used to represent the validity of two fragments being properly joined together into a correct pair, and as such being part of a correct JPEG image.

The parameters we judge as being important when decoding a JPEG scan have been tested during the development of the image fragment reconnection method. The parameters represent different logical layers of a JPEG image, starting at the bottom with the raw byte stream and ending at the DC frequency pixel level. The evaluation and the results of that method are presented in Section 3.4 and Section 3.5.

For the first round of tests we used image for all tests. We extracted the picture data and divided it into 381 4 KiB large fragments, which were then used for the reconnection experiments. A brute-force attempt to put together the original image from these fragment pairs would give 381! sequences to test for the correct combination. For the next round of experiments we used images from several different cameras. This is explained in more detail in Section 3.4.

### 3.1 Background

A JPEG file is made up of a *header* section and a data section, which is called *scan*. A header consists of some markers containing tables and other information, and compressed pixel data in the scan. The raw pixel data is divided into *data units* of  $8 \times 8$  pixels. The pixels are often coded as Red Green Blue (RGB) colours.

The majority of the markers can be found in the header section, but there are markers which may appear in the scan too. All markers start with the byte `0xFF` and then one byte identifying the marker type. Most of the markers are followed

by a two byte long value indicating the length of the marker, excluding the two first bytes. A JPEG file always starts with the Start-Of-Image (SOI) marker, `0xFFD8`, and ends with an End Of Image (EOI) marker, `0xFFD9`. The scan is compressed in several ways, one of them being an entropy encoding, usually Huffman.

The Start of Image (SOI) marker should always be the first two bytes of a JPEG image. Its hexadecimal value is `0xFFD8` and it stands alone, i.e. is not followed by a two-byte length value. This marker is often used by file carving tools to identify JPEG files. Any JPEG coded image is preceded by this marker. This means that there can be several markers in one image file, because if there is a thumbnail version of the image embedded in the header section the thumbnail will also start with a SOI.

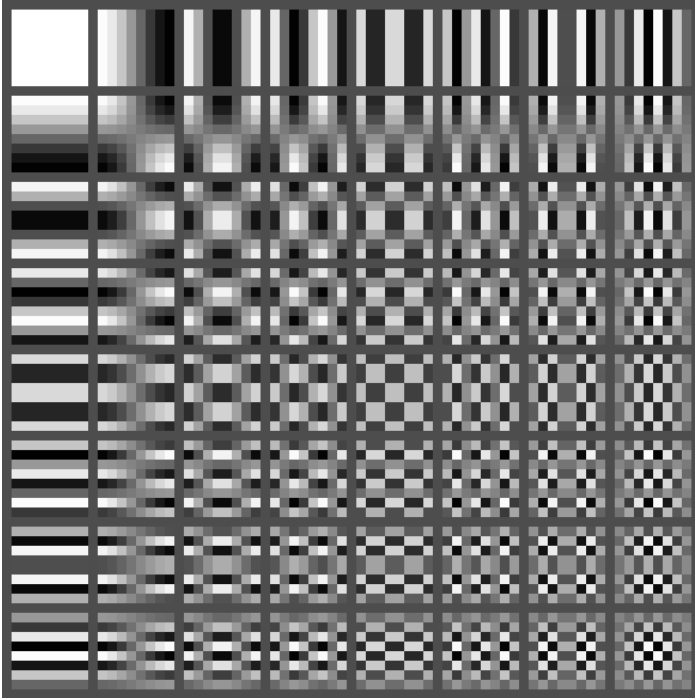
The EOI marker is denoted by `0xFFD9` and marks the end of a scan. Therefore it is not followed by any length value. Since embedded thumbnails are small images in themselves there can be several EOIs in one image file. This marker is used by file carving tools to stop the carving when dealing with JPEG images. Since an image file can be fragmented this is an inexact way of extracting JPEG image files; there is a high risk of non-JPEG data being included in the extracted data. There is also a risk of missing the real image and only carve a thumbnail version of it if the carving algorithm is not set to require a large enough amount of data to be included in an image before allowing the appearance of an EOI.

Colour baseline sequential JPEG images do not use the RGB scheme, instead they use a *luminance and chrominance* colour coding scheme. The luminance represents the black and white nuances of the image and the chrominance the colours, often taken from a two-dimensional palette. A common colour coding scheme used for JPEG images is the YCbCr scheme, where Y represents the luminance, Cb represents the blue hues and Cr represents the red hues of an image. Each of the Y, Cb, and Cr are called *components* of the image. There are other colour coding schemes possible to use, e.g. YIQ, YUV, and CMYK, but they are all variants of the luminance and chrominance coding scheme. The reason for not using the RGB colour coding scheme is the fact that it is easier to compress images using a luminance and chrominance coding scheme.

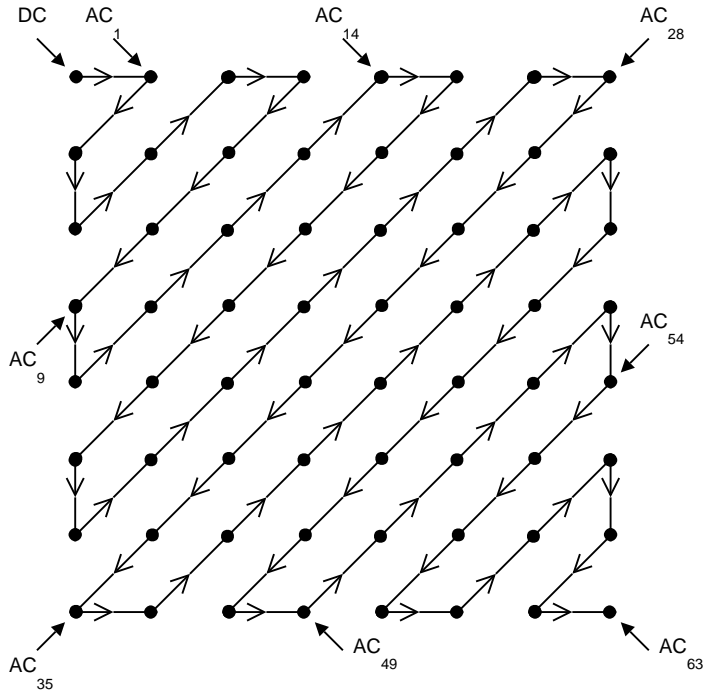
One of the compression steps utilised in JPEG images uses the Discrete Cosine Transform (DCT) to convert from the amplitude domain to the frequency domain. Performing calculations in the frequency domain allows lossy compression to be applied to an image by zeroing out higher frequencies. The higher frequencies hold the tiny details of an image and can be omitted without much loss of quality. This can be seen in Figure 3.1, which shows the frequency coefficients of a JPEG image data unit.

The JPEG coding algorithms are applied to blocks of one to four data units at a time, starting at the upper-left corner of an image and traversing it horizontally. Since the luminance is more important for the perceived quality of an image further compression is sometimes achieved by extracting the chrominance for larger areas of an image. This is done by using pixel pairs for the chrominance, either horizontally, vertically, or both. Hence sometimes two or four data units

### 3.1. BACKGROUND



**Figure 3.1:** The 64 frequencies resulting from a DCT of a JPEG data unit. The DC coefficient is in the upper-left corner. The image is copied from Wikipedia [27].



**Figure 3.2:** The zig-zag order in which a data unit in the frequency domain is traversed to form a 64 items long vector. The DC coefficient is in the upper-left corner. The AC coefficients are traversed in increasing frequency order.

are used at the same time, the standard stipulates that this can be done for any component, as long as the total sum of data units used is less than 11, counting the data units for each component separately.

The block of frequencies resulting from the DCT is traversed in zig-zag order (see Figure 3.2) to form a 64 item long vector. The first item is the DC coefficient, having zero frequency both horizontally and vertically. The rest of the coefficients are called AC coefficients.

The smallest unit to be put into the scan part of a JPEG image is a Minimum Coded Unit (MCU). It contains at least one of each components of a data unit. If the scan part of an image does not contain a discrete number of MCUs it is corrupt. Hence the quality and structure of the MCUs in the scan part of an image file is a good thing to check when working with JPEG fragments.

## 3.2 Requirements

To be able to reconnect fragments we need a method that does not require any, or very little, post-processing. To be usable the method has to be able to correctly point out a significant amount of the correct fragment pairs at once by ordering

them based on their fragment joint validity. The validity measurements should clearly differentiate between a correct pair and a false pair.

The method should scale well and be able to handle fragments from different images at the same time. This also requires the method to be able to separate fragments belonging to images with different restart intervals<sup>1</sup> and MCU sizes. The method should also be able to handle black-and-white and colour image fragments at the same time, which means fragments with different amounts of image components in a frame.

It is important to be able to find the starting fragment of a scan, since we then can base the rest of the processing on that information. Having access to the starting fragments in a collection we get a lower bound on the number of possible images to recreate.

When looking at the connection between two fragments the most probable connection point is in the middle of a MCU. Either the joint cuts through a Huffman code, a value code, or through the boundary between them. These types of joints are trivial to handle and can be used for calculating the fragment joint validity. There are however situations where the joint cuts through, or occurs at the beginning or end of, a marker segment. These types of joints are harder to handle and depending on the abstraction level we are working at, such a joint may give very little or no information regarding the fragment joint validity.

The method used for reconnection of image fragments should be able to correctly handle all types of joints.

## 3.3 Parameters Used

This section presents the parameters that have been tested for their ability to help in reconnecting JPEG image fragments. Results are included even for the parameters that do not meet the requirements specified in Section 3.2. For each experiment involving a specific parameter we tried to falsify the hypothesis that a specific parameter was good enough to fulfil the specified requirements.

### 3.3.1 Background

The coding of a JPEG image is more complex than what will be described in this section. We have chosen to exclude details that are not relevant to the understanding of our work and focus on the overall process, to be able to clearly explain our ideas. Our algorithms, tools and experiments are all compliant to the original JPEG standard from 1992 [13], with extensions. A full description of the coding process can be found in Annex F of the JPEG standard [13, pp. 87–118].

The scan contains the compressed pixel data from the light sensors of the camera. As mentioned earlier the data are first converted to a luminance and chrominance colour scheme, often having three components. The components are transformed into the frequency domain with the help of the Discrete Cosine

---

<sup>1</sup>The distance in MCUs between two RST markers. More information is available in Section 4.4.

Transform, giving a vector with amplitude values for 64 frequencies in increasing order.

The frequency vectors are compressed using a quantization table, which gives a reduction factor for each frequency. The higher frequencies are usually reduced to a higher degree since they hold less important information for the overall quality of the image. The integer part of the resulting quantized values are used, hence values less than 1 are set to 0, giving a lossy compression.

In Figure 3.3 we can see how the coding of the DC coefficients is based on the difference of two consecutive DC values for components of the same type. The length of the signed difference value is Huffman coded and the value itself follows immediately after the Huffman code. The longest representation, i.e. the largest difference to be coded, is 11 bytes long. The first DC value of a scan, or in the case of RST markers being used, the first DC of a new restart interval, is set to the actual luminance value<sup>2</sup> of that data unit. Since both the length of the Huffman code and the following value can vary in size there is no guarantee that Huffman codes or value representations start at byte boundaries.

Figure 3.3 also shows the AC value coding. The AC values of the frequency vector of a component often holds sequences of consecutive zeros. Therefore the AC values are zero run length coded to further compress the scan data. This is done by letting the first nibble<sup>3</sup> in a byte represent the number of consecutive zeroes since the last non-zero value in the AC part of the vector, and the second nibble<sup>4</sup> hold the bit length of the next non-zero value.

The length value for the number of consecutive zeroes is restricted to 15. The value for the number of bits used to represent a non-zero value is restricted to 10 for baseline sequential JPEGs. The resulting byte is Huffman coded, put into the scan and immediately followed by the signed non-zero value represented by the number of bits given in the Huffman coded byte. As for the DC component coding, this means that there is no guarantee for the AC component Huffman codes or value representations to be byte aligned.

The scan part might contain markers too. If RST markers are used the scan is interrupted at regular<sup>5</sup> intervals by a marker  $0xFFDx$ , where  $x_{i+1} = (x_i + 1) \bmod 8$ ;  $x_0 = 0$ . The marker is not followed by any length value bytes. If RST markers are used in the scan the JPEG file header will contain a DRI marker,  $0xFFDD$ , containing a value larger than zero, indicating the number of MCUs between each RST marker (see also Section 4.4).

Sometimes the encoding of the image may lead to situations where a byte value of  $0xFF$  will be used in the scan without signifying a header segment. In those cases the JPEG standard stipulates that the  $0xFF$  value should be converted to  $0xFF00$ . The extra zeros should be ignored when decoding the scan.

Any value other than  $0x00$  or  $0xD0$  to  $0xD7$  following a  $0xFF$  value is forbidden and should generate a decoding error. This requirement can be utilised

<sup>2</sup>The algorithm uses 0 as a starting value, giving  $DC_0 - 0$  as the first difference value.

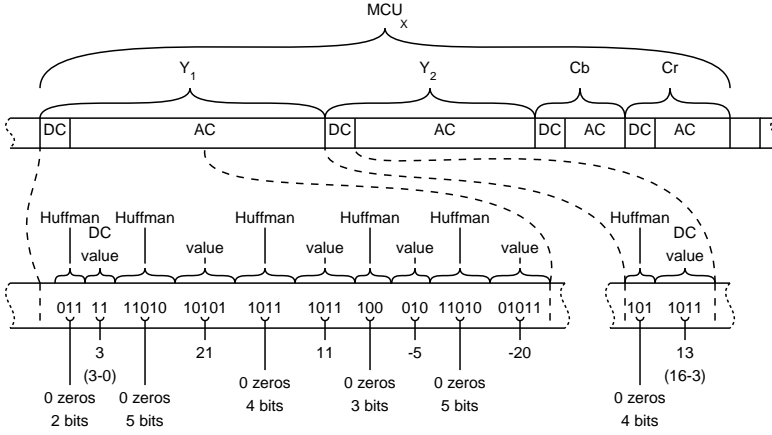
<sup>3</sup>The first four bits of a byte.

<sup>4</sup>The last four bits of a byte.

<sup>5</sup>The intervals are regular relative to the number of MCUs, the number of bytes between two RST markers may vary.



### 3.3. PARAMETERS USED



**Figure 3.3:** The scan part in binary format showing the DC and AC component encodings for a MCU. The DC values are given as the difference between two consecutive DC components. This is the first MCU after a restart marker, hence the DC value 3 – 0. The Huffman codes contain the number of consecutive zeros and the number of bits needed to encode the non-zero value.

to increase the precision when creating file type models.

#### 3.3.2 Correct decoding

An obvious way of checking if two fragments can be joined together into a larger JPEG fragment is to look at the sequence of RST markers. We then use the requirement that for two consecutive fragments,  $i$  and  $i + 1$ , containing RST markers, the expression

$$\text{RST}_{i+1}^{\text{First}} = (\text{RST}_i^{\text{Last}} + 1) \bmod 8$$

should be true.

There is a problem if the joint cuts through the  $\text{FFDx}$  pair. The  $0\text{xFF}$  might belong to a  $0\text{xFF00}$  marker, or the  $\text{Dx}$  might be just an ordinary part of the data stream, not the second half of a marker. We therefore have to include fragments ending in  $0\text{xFF}$  and fragments starting with  $0\text{xD0}$  to  $0\text{xD7}$ , and  $0\text{x00}$ . The amount of extra fragment pairs to check after such an inclusion is manageable.

Using a requirement of a correct RST marker sequence our test image gives us 18012 possible fragment pairs. 190 of the pairs are correct and can be joined together into a full image. The set of fragment pairs includes pairs where the joint cuts through a marker. The method of using correct RST markers sequences cannot give the fragment joint validity, thus we have to test every possible combination to find the correct one. If we assume there are an equal amount of possible fragment pairs sharing the same start fragment, we still need to test more than  $(18012/381)^{380} > 47^{380}$  combinations to recreate the original image.

This is much better than the 381! fragment combinations to be tested using a brute force method, but not good enough.

Decoding the actual data stream, starting at the last known RST marker in the potential first fragment of a pair, will help us tell whether the 0xFF belong to a 0x00 or is part of a RST marker. If the decoding ends in a proper way, it is a RST part, if the decoding fails at the 0xFF it belongs to a 0x00. We need not use the embedded amplitude values nor take care of more than the length information<sup>6</sup> in the Huffman code. The decoding has to check for the correct number of MCUs per DRI, and that each MCU is correctly built. This assumes we know these features in advance, but the possible number of allowed values are low, hence we can test for each of them.

The full decoding strategy can also be used to check if any fragment pair is a proper one. If a pre-selection of possible fragments has been done using the RST markers, the full decoding strategy will decrease the amount of possible fragment pairs to 3118 for our test image. This amounts to more than  $(3118/381)^{380} > 8^{381}$  combinations to test, which is too many. Since the method can only tell whether a certain fragment pair is possible or not, we cannot get any measurements of the fragment joint validity and consequently have to test every combination.

### 3.3.3 *Non-zero frequency values*

The embedded amplitude values in the stream can be used to measure the fragment joint validity. We define the length of a luminance or chrominance amplitude value vector as the index of the highest frequency non-zero amplitude, starting at the DC frequency. To get a value of the fragment joint validity we measure the luminance and chrominance vectors' average lengths for a MCU and compare them to average values calculated over several hundred megabytes of data.

The length of the luminance vector(s) of a MCU is related to the length of the chrominance vectors, because they all represent the same collection of pixels. The same quantization tables are used for all vectors of the same type throughout the image, thus the relation of the lengths of the luminance and chrominance vectors should be fairly constant, at least compared to the actual differences. This is true even when the amount of details encoded in each MCU varies. We therefore used the relation between the luminance and chrominance vectors as an alternative measurement of the fragment joint validity.

We also tested a variant of the average difference value luminance and chrominance relation measurement. Instead of using a single average value, we measured the probability of a certain vector length. The measurements were made over the same amount of image data as for the average measurements. The probabilities for each MCU, as well as the sum of the probabilities over a restart interval was used to calculate the fragment joint validity.

A third method that we tested was to code the luminance and chrominance vectors as 8 byte sequences where each bit position indicated if the corresponding

---

<sup>6</sup>The second nibble of the decoded Huffman value.

### 3.3. PARAMETERS USED

vector value was non-zero. We then calculated the probability for each vector pattern. To lower the amount of required data storage we truncated the 8 byte sequences and kept only the upper 4 bytes (the 32 lowest frequencies in each vector). The byte sequences were used to calculate the sum of the probabilities of each vector for every MCU, and all MCUs in a restart interval. We also calculated the combined probability of the luminance and chrominance vectors to be able to use only one value for the measurements.

Finally we created a matrix containing the frequencies of each vector position and amplitude value of the luminance and chrominance vectors. We then calculated the mean and standard deviation of the frequency values. This method became too slow and complicated to be useful, and we therefore did not proceed with any further tests.

None of the methods described in this section did fulfilled the requirements specified in Section 3.2.

#### 3.3.4 *Luminance DC value chains*

The DC luminance value is the most important factor for the quality of a JPEG image. It can be used to show a low-resolution, grey-scale version of the original image. Consequently it contains the outlines of the objects in the image. Assuming there is at least one vertically oriented line in the image, the luminance DC values can be used to correctly rebuild the image. The vertically oriented line need not be perpendicular to the horizontal edge of the image, but the closer, the better. Since most of the objects in an image have smooth outlines, this method will work for a large set of images. The use of the DC values puts this method at the pixel level in the coding model of a JPEG image.

The method builds on the fact that the scan of a JPEG image is started at the top-left corner and proceeds one horizontal line at a time downwards. Therefore the vertically oriented lines will be reflected as a high rate of change of the DC values in a limited area. If we use two sliding windows that move through the DC value chain and calculate the difference between the DC values in the windows we get a measurement of similarity of the DC sub-chains.

The method consists of several steps:

1. Process all available image fragments, check for correct RST marker sequences and fully decode all restart intervals.
2. Identify the area with the highest rate of change and the distance to its closest match using a sliding window approach.
3. Use the distance values to calculate the most probable width (distance between two high rate of changes) of the image.
4. Rerun the algorithm with the calculated value for most probable width as a fixed distance value for all fragments and calculate the area between the DC curves of the high rate of change windows.

5. Use the area value as the fragment joint validity, the smaller the area the higher the validity.

The size of the sliding window can change automatically and will be shown together with the fragment joint validity value, because the window size affects the confidence of the validity value. Larger window sizes gives higher confidence, but requires larger fragments. We also give a weighted fragment joint validity value where the original validity is divided by the window size.

The rate of change of the similarity values calculated when the two sliding windows move in parallel indicate whether the fragment pair is correctly connected or not. A low rate of change shows that the distance between the sliding windows is close, or equal, to the width of the image, and hence the pair is correct. However, if the DC values are static the rate of change of the similarity values will also be low. We therefore also measure the rate of change of the DC sub-chain in a sliding window. To get a picture of the rate of change for the whole fragment pair we measure the rate of change of the rate of change values. A higher overall rate of change indicates that the fragment pair holds both smoother areas and more rapidly changing areas. This is typical for parts of proper images.

Depending on the compression rate of the image and the fragment size used the method can handle images of different sizes with different rates of success. Generally the use of 4 KiB fragments give a maximum image width of approximately 2000 pixels. The compression rates in our data set gives a range between 1700 and 4200 pixels for a 4 KiB fragment. If the level of detail is high, the compression level is low, and the maximum image width the image fragment reconnection method can handle is decreased. Of course the opposite situation is also true.

The DC value chain algorithm will give the fragment joint validity of all possibly correct fragment pairs. Each fragment can be part of two pairs, one as the first member of the pair and one as the second member. Thus it is possible to piece the pairs resulting from an iteration of the algorithm into a complete image, if all fragments of the image are present. Identifying the end fragment is trivial, the possible start fragments are detected by the algorithm.

### 3.4 Evaluation

This section presents how the evaluations of the image fragment reconnection method was set up. We evaluated the method using both a large amount of single images, as well as using fragments from six images at the same time, due to an unfortunate duplication of seven images.

Please observe that the images used in the evaluation are named and referenced as being 240, but only 233 are used in reality.

### 3.4.1 Single image reconnection

The DC value chain algorithm was evaluated using 233 images from 3 cameras using RST markers in our database. We included 40 of the largest image files from each camera to account for low compression rates and high levels of detail. We also included 40 of the smallest image files from each camera to incorporate technically bad images, featuring blur, significant low key or high key, and other extreme conditions. Due to a mistake 7 of the Fuji FinePix 240Z images in the small image group was duplicated from the large image group, consequently in reality we only used 113 small images. The tests are however still numbered as being 240, not 233. The reason for the selection of images<sup>7</sup> was to test the limits of the method.

The evaluation was performed by first splitting each image into 4 KiB fragments. The largest image file gave 446 fragments and the smallest file was split into 57 fragments. The compression rates varied between 3.32 and 8.16.

The fragments were combined into pairs with the help of their first and last RST markers, as described in Section 3.3.2. Then a first run of the DC value chain algorithm was made and the image width in DC values was calculated. The next run of the algorithm used the width and measured the fragment joint validity of each fragment pair. Finally the fragment joint validity values were sorted and the number of correct pairs at first to fifteenth place were counted. We also noted the number of pairs that were missing, i.e. not properly categorised, and the reasons why.

The reasons for the missing fragment pairs are:

**Missing:** the algorithm can not correctly decode the fragment pair, for miscellaneous reasons (excluding the following reasons),

**Short loop:** the pair is too short to fit even the first algorithm run,

**Short 2nd:** the second fragment is too short,

**DC 0 seq:** all DC values are zero, and

**Short total:** too few DC values in the pair to fill the image width.

All missing fragments have been manually checked to find out the reason for why the algorithm could not properly categorise them. The reasons are presented in Section 3.5.

### 3.4.2 Multiple image reconnection

We also tested the algorithm's ability to handle fragments from several images at the same time. To do this we fragmented six images into 4 KiB fragments, giving a total of 1331 fragments. The images came from the set of 233 images used for the other reconnection evaluation and can be found in Table 3.1.

---

<sup>7</sup>The raw data files and source code can be found at <http://www.ida.liu.se/~iislab/security/forensics/material/> or by contacting the author at [g-makar@ida.liu.se](mailto:g-makar@ida.liu.se)

**Table 3.1:** The images used for the multiple image reconnection evaluation. The files are taken from the set of 233 images used for the single-file reconnection evaluation.

Test	File name	Size	Pixels	Comp.	Corr. width	Est. width
116	jocke_fuji_257	296003	1280x960	4.15	160	320
83	jocke_fuji_283	314698	1280x960	3.90	160	320
43	lars_olympus_p9020199	1347595	2816x2112	4.41	352	352
123	lars_olympus_p9090233	308676	1600x1200	6.22	200	400
163	micke_fuji_0730	1451003	2848x2136	4.19	356	356
23	micke_fuji_1535	1719515	2848x2136	3.53	356	356

The 1331 fragments were processed in the same way as was done for the single-file reconnection evaluation. The images were selected to represent one high compression rate image and one low compression rate image from the three cameras using RST markers in our data set.

## 3.5 Result

In this section the results of the two evaluations are presented, together with comments on the results. The images used in the evaluation are referenced as being 240, although there are only 233 unique images used in reality. Because of a mistake 7 images were duplicated. The duplicates are excluded from the results and tables.

### 3.5.1 Single image reconnection

The results<sup>8</sup> of the reconnection evaluation using 233 images are given in Table 3.2. We have calculated the percentages of fragments in each category relative to the total number of fragments, and also relative to the sum of fragments that the algorithm should have categorised. To avoid incorrect pairing of fragments due to an insufficient amount of data the algorithm is set to exclude fragments categorised as too short, i.e. not a full image width in length, or where all luminance DC values are 0.

The results show that the fragment joint validity value is highest for the proper image fragment pair in 96.8% of the cases. If we also include the second highest fragment joint validity we cover 98.8% of the proper pairs. From third to fifteenth place the number of proper pairs at each level decreases by a factor of approximately 2.

For 64 of the 233 images the algorithm correctly connected all fragment pairs. Thus an image can theoretically be fully restored in 27.5% of all cases. For 47

<sup>8</sup>The raw data files and source code can be found at <http://www.ida.liu.se/~iislab/security/forensics/material/> or by contacting the author at [g-makar@ida.liu.se](mailto:g-makar@ida.liu.se)

### 3.5. RESULT

**Table 3.2:** The results of the reconnection evaluation using 233 images. The first percentage column is calculated relative to the total, and the second relative to the sum of fragments excluding the short and DC 0 sequences.

Connection validity rank	Number of fragments	Fragments including short [%]	Fragments excluding short [%]
1:	42884	85.130	96.821
2:	864	1.715	1.951
3:	235	0.466	0.531
4:	106	0.210	0.239
5:	59	0.117	0.133
6:	42	0.083	0.095
7:	26	0.052	0.059
8:	11	0.022	0.025
9:	12	0.024	0.027
10:	5	0.010	0.011
11:	0	0	0
12:	2	0.004	0.004
13:	0	0	0
14:	1	0.002	0.002
15:	1	0.002	0.002
Missing:	44	0.087	0.099
Short loop:	2	0.004	-
Short 2nd:	0	0	-
DC 0 seq:	7	0.014	-
Short total:	6074	12.058	-
Total:	50375	100.000	100.000

of the remaining 169 images in the evaluation the algorithm gave the correct pairs the highest and second highest fragment joint validity. Consequently a brute force reconnection using the two highest fragment joint validities of each starting fragment in a possible pair would give a fully correct image scan part in more than 52.4% of the cases.

There was a total of 44 missing fragments, spread over 10 images. Only one image gave rise to more than one missed fragment. The reasons for the missed fragments were:

**Test 43, 63, 91, 105, 162, 169, 209, 210, 231:** The last fragment is too short to contain any RST marker. Hence the first check of the RST marker sequence is too strict and omits the fragment.

**Test 153:** The width of the image was wrongly detected to be 600 MCUs. The correct value was 200. Consequently the fragment was not included in the final set of probable fragment pairs.

There is really only one reason for all the missed fragments, and that is that we have put constraints that are too strict on the algorithm. We have done so because we want to keep the amount of false positives down. If we happen to miss one correct pair it matters less than having many incorrect fragment pairs to waste time on. Knowing we can trust the algorithm to give a low degree of false positives lets us use the algorithm repeatedly, having only a small amount of possible fragment pairs to deal with in each iteration.

The problem with the short last fragment, lacking a RST marker, can be helped by a more intelligent pre-screening of the fragments. Since we have used the file type categorisation method to extract the fragments we know that they are most probably JPEG fragments. We can therefore look for all fragments with the EOI marker and include them in the process even though they do not contain any RST markers. Should they not belong to the image we want to recreate, we will probably get an error. Missing the last fragment of an image containing RST markers does not do any serious harm, the image will be correct apart from the last restart interval, i.e. the lower right corner of the image.

The image width problem originates in the fact that some highly compressed fragments can actually contain more than a full image width of DC values. Since the algorithm is set to compare the two fragments, it starts the second sliding window at the border of the fragments, in the second fragment. Therefore the distance between the sliding windows has to be a multiple of the image width. As long as there are good vertical lines without any disruptions in the image there is no real problem, the algorithm can still correlate the two sequences.

The problem with the incorrect image width is manageable. We can manually see that a size of 600 MCUs is not probable<sup>9</sup>. The fact is that the image fragment reconnection method identified the width to be 400 or 600, not 200 as it should be, for all but one of the images for that camera and resolution. One

---

<sup>9</sup>Currently the standard cameras have a maximum resolution of about 12 M pixels. A width of 600 MCUs for such a camera would be equal to at least 15 M pixels.



### 3.5. RESULT

**Table 3.3:** The results of the reconnection evaluation using 6 images at once. The first percentage column is calculated relative to the total, and the second relative to the sum of fragments excluding the short and DC 0 sequences.

Connection validity rank	Number of fragments	Fragments including short [%]	Fragments excluding short [%]
1:	748	56.241	70.433
2:	77	5.790	7.250
3:	39	2.932	3.672
4:	25	1.880	2.354
5:	16	1.203	1.507
6:	23	1.729	2.166
7:	15	1.128	1.412
8:	13	0.977	1.224
9:	14	1.053	1.318
10:	10	0.752	0.942
11:	8	0.602	0.753
12:	6	0.451	0.565
13:	13	0.977	1.224
14:	6	0.451	0.565
15:	3	0.226	0.282
> 15:	40	3.008	3.766
Missing:	6	0.451	0.565
Short loop:	0	0	-
Short 2nd:	0	0	-
DC 0 seq:	0	0	-
Short total:	268	20.150	-
Total:	50375	100.000	100.000

of the images was identified as being 599 MCUs wide. Still only one of the 40 images was affected by the bad width estimation. We can therefore conclude that the image fragment reconnection method is robust and handles noise well.

#### 3.5.2 Multiple image reconnection

The multiple image reconnection evaluation used 1331 fragments from 6 images, giving a total number of 219200 possible fragment pairs. These were reduced to a total of 32515 possible fragment pairs after setting the image width to 356 DC values in the DC value chain algorithm. The final result is given in Table 3.3.

The 40 fragment pairs that were given fragment joint validity values placing them at 16<sup>th</sup> place or worse comes from the use of the wrong image width. The width that was used originates from the majority of the fragments. When these are removed another size would be estimated, and related to the new most

probable image width, possibly fitting the 40 fragment pairs better. In this way the set of fragments can be iterated through and eventually all fragment pairs are correctly connected.

Five of the six missing fragment pairs are related to the last and first fragments of different images. Hence they cannot be connected. By manually examining them we can easily find the start and end fragments of the involved images, which helps us identify the number of images in our data set. This will in the end allow us to recreate any image where we have access to all fragments.

The remaining missing fragment pair, which does not contain an image start and end fragment, is missing because its second fragment is too short to contain a RST marker. It is the end fragment of an image and is only 11 bytes long, hence it does not contain a full restart interval, or even a complete MCU.

Although the multiple image reconnection ability of the image fragment reconnection method is lower than for single images, the method is still able to correctly identify the majority of the fragment pairs. By using it repeatedly and removing fragment pairs that have been correctly joined, any number of fragment pairs and images should be possible to handle. The execution time increases linearly with the number of potential fragment pairs. Therefore, it is important to use pre-selection using the RST markers to decrease the number of fragment pairs to be checked.

## Chapter 4

# Viewing Damaged JPEG Images

This chapter will explore the effects of missing or erroneous JPEG image header tables. If the image fragment reconnection method does not have access to all fragments of an image, or for some other reason cannot connect all fragments into the original image, the same problems arise as when we have a missing or damaged JPEG header. But when the header is missing completely, it is in some situations still possible to view the image.

The most important header marker segments and tables will be presented in separate sections. The presentation is not meant to be exhaustive, only to show some of the more significant errors and their effects on the image. The image used to show the effects of the manipulations we have made is shown in its original undamaged state in Figure 4.1.

### 4.1 Start of Frame

There are 13 different SOF markers, but our work only concerns the first one (SOF<sub>0</sub>), which is used for non-differential baseline sequential DCT Huffman coded images.

The SOF marker contains the following information, which can be seen in Figure 4.2:

**Start Of Frame (SOF)** Marks the beginning of the Start of Frame marker segment, indicated by the hexadecimal value of 0xFFC0.

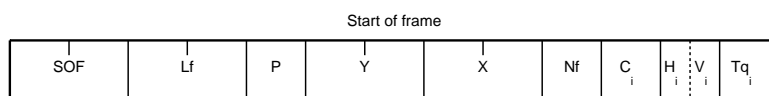
**Frame header length (Lf)** The length of the marker segment, excluding the first two marker bytes. The formula used to calculate the length is  $8 + 3 \times Nf$ .

**Sample precision (P)** The number of bits used for the samples of the components in the image. Should always be 8 for baseline sequential DCT JPEGs.

**Number of lines (Y)** The height of the image, can be 0 to 65535 lines.



**Figure 4.1:** This is the original undamaged image used to show the effects of different manipulations of the JPEG header.



**Figure 4.2:** The Start Of Frame (SOF) marker segment

**Number of samples per line (X)** The width of the image, can be 1 to 65535 samples per line.

**Number of image components in frame (Nf)** Number of components used to encode a pixel in the image. A grey-scaled image has one component, a colour image has three or four components depending on the colour coding system used. The valid values lay in the range of 1 to 255 for baseline sequential DCT JPEGs.

**Component identifier ( $C_i$ )** A unique label used to identify component  $i$  in an image. The label can be 0 to 255.

**Horizontal sampling factor ( $H_i$ )** Specifies the number of horizontal data units of component  $i$  in each MCU, ranging between 1 and 4.

**Vertical sampling factor ( $V_i$ )** Specifies the number of vertical data units of component  $i$  in each MCU, ranging between 1 and 4.

**Quantization table destination selector ( $Tq_i$ )** Defines which quantization table to use for component  $i$  of up to four possible, for baseline sequential DCT JPEGs.

The SOF header section contains several parameters affecting the quality of the image. The number of data units in each MCU can be changed in many ways, leading to a number of effects appearing in the image. If any of the chrominance quantization tables get a high (or erroneous) sample factor ( $H_i$ ), the result will be a discolouring of the image, together with single MCUs creating specks. This is not very probable, though, since the common case is to repeat the luminance quantization table several times to compensate for the lower resolution specified for the chrominance, especially horizontally. If we retain the sum of the sample factors in the image, i.e. move the factor from the luminance quantization table  $H_1$  to one of the chrominance tables, we get the effect that can be seen in Figure 4.3. In this specific case it is the Cr ( $H_2$ ) table that is affected.

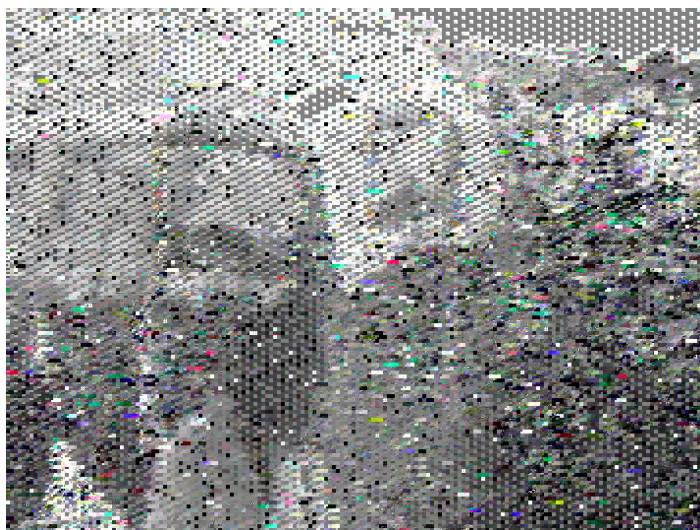
The luminance quantization table is more important than the chrominance dittos for the overall quality of the image. This becomes apparent when the luminance table gets the wrong sample factor. In our case the original image has two-to-one pixels horizontally and one-to-one vertically, i.e.  $H_1 = 2$  and  $V_1 = 1$ . If we increase that to two-to-one vertically ( $V_1 = 2$ ) we get the effect that can be seen in Figure 4.4. The image is stretched out vertically to twice its height and the colours are almost gone. Areas that have thin colour in the original image are nearly perfectly decoded into a grey scale, but coloured areas are heavily affected.

If we instead decrease the sampling factor for the luminance quantization table into one-to-one both horizontally and vertically, i.e.  $H_1 = 1$  and  $V_1 = 1$ , the effect is that the image gets rendered as one fourth of its original size. Since the original image has a two-to-one horizontal sample factor the image is repeated twice horizontally. The effect is shown in Figure 4.5.

When the component identifiers  $C_i$  of an image are swapped, the effect is clearly visible. Swapping the two chrominance components results in heavy



**Figure 4.3:** The effect on a JPEG image having the horizontal sample rate  $H_1$  of the luminance quantization tables exchanged with the Cr chrominance table ( $H_2$ ), retaining the sum of the sample factors for the image.



**Figure 4.4:** The effect on a JPEG image having the sample rate  $V_1$  of the luminance table set too high.

discolouring of the image. The effect comes from swapping the red and blue hues in the image, and typically human skin turns significantly blue. An example of that can be seen in Figure 4.6.

If the component identifier of the luminance is swapped with one of the chrominance components, the result is an almost unrecognisable image. This clearly shows the importance of the luminance for the overall quality of the image. Since the luminance is often reduced less by the quantization step of the JPEG process, the effect of the swap is a decreased contrast and a more intense red or blue colour, compared to the original image.

An image having the luminance component identifier swapped with the blue, Cb, chrominance component can be seen in Figure 4.7. The disproportionate red colour comes from the non-affected red Cr chrominance table and values in the scan. The reason for the discolouring is that the values in the luminance quantization table are lower than the chrominance table values, therefore the affected chrominance table will not properly restore the full colouring of its corresponding hue.

When the width of the image is wrong, the image will be symmetrically distorted in either left or right direction. If the image width is very wrong the pattern will be almost horizontal. An example of a moderately distorted image, with the width increased by 16 pixels, can be seen in Figure 4.8. The diagonals from the lower left corner to the upper right corner are typical for images where the image width, the number of samples per line  $X$ , is  $n \times \text{width} \leq X < 1.5n \times \text{width}$ , where  $n = 0, 1, 2, \dots$ . If the width instead is  $1.5n \times \text{width} \leq X < n \times \text{width}$  the diagonals are oriented from the upper left corner to the lower right corner.

An increase of the image width in intervals of the original correct image width will generate a fully viewable image, which is repeated horizontally a number of times. Even changes of the image's width to one half the original width keeps enough of the image to make it easily recognisable. If we convert all images in our database to landscape format, the most common relation between the width  $w$  and the height  $h$  of the images is 1.333, i.e.  $w = \frac{4}{3} \times h$ . The sizes, width and height relations, and number of images of each size can be found in Table 4.1. The image sizes were extracted using the Linux *exif* application.

Since the relation between the width and the height of an image in reality only take on a few values, it is easy to guess the correct width or height of an image. The light sensors of digital cameras often come in a few known sizes, which makes the guessing even easier. To further simplify the matter, there are only a few manufacturers of light sensors for cameras in the world, hence there is a high probability of the same sensor make in different cameras.

Changes of the image height will only affect the number of lines of the image shown, incremented in steps of 8 or 16 pixels per line depending on the vertical sampling factor. If the height is set too high, the bottom of the image will be extended by a grey area.





**Figure 4.5:** The effect on a JPEG image having the sample rate of the luminance table set too low, i.e.  $H_1 = 1$  and  $V_1 = 1$ .



**Figure 4.6:** The effect on a JPEG image having the identifiers of the two chrominance components swapped.



#### 4.1. START OF FRAME



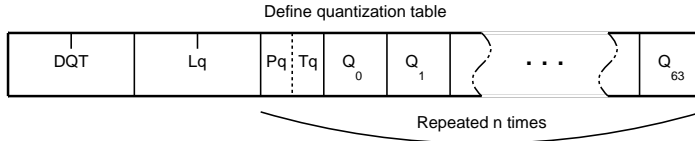
**Figure 4.7:** The effect on a JPEG image having the component identifiers of the luminance swapped with one of the chrominance identifiers, in this case the blue Cb component.



**Figure 4.8:** The effect on a JPEG image with the image width, the number of samples per line, being set too high (increased 16 pixels in this case).

**Table 4.1:** The relation between the image width and height of the images in our data base. All images sizes have been converted to landscape format.

Width	Height	Relation	# of images
640	480	1.333	110
1280	960	1.333	73
1397	931	1.500	1
1536	1024	1.500	685
1600	1200	1.333	1528
1888	1259	1.499	1
2048	1365	1.500	2
2048	1536	1.333	3594
2272	1704	1.333	298
2304	1728	1.333	216
2816	2112	1.333	1257
2848	2136	1.333	1211
2883	1922	1.500	1
3008	2000	1.504	178
3456	2304	1.500	262
3658	2439	1.499	1
3888	2592	1.500	402



**Figure 4.9:** The Define Quantization Table (DQT) marker segment. The last 65 bytes in the marker segment are repeated  $n$  times, where  $n$  is the number of quantization tables used.

## 4.2 Define Quantization Table

The Define Quantization Table (DQT) marker segment has the hexadecimal value `0xFFDB` as its first two bytes. The tables are used to compress the image by putting weights on the 64 frequencies from the discrete cosine transformation. The amplitude of each frequency is divided by a value in the table. Higher frequencies are less important for the perceived quality of the image and thus get a higher value in the table.

The outline of the segment can be seen in Figure 4.9 and contains the following information:

**Define Quantization Table (DQT)** Marks the beginning of the Define Quantization Table marker segment. The marker segment is indicated by the hexadecimal value `0xFFDB`.

**Quantization table definition length (Lq)** The length of the marker segment, excluding the first two marker bytes. The length value for baseline sequential DCT JPEGs is calculated as  $2 + n \times 65$ , where  $n$  is the number of quantization tables used.

**Quantization table element precision (Pq)** This value is always 0 for baseline sequential DCT JPEGs, meaning 8 bit precision.

**Quantization table destination identifier (Tq)** Specifies one of four destinations for the quantization table when decoding an image. Valid values are 0 to 3.

**Quantization table element ( $Q_k$ )** Gives the  $k^{th}$  value, in order of increasing frequency, to use for the quantization step when coding and decoding an image. For 8 bit precision the values should be in the range of 1 to 255. To achieve the specific ordering of the elements an  $8 \times 8$  block of pixels in the frequency domain is traversed in zig-zag, beginning at the top-left corner and the next step being horizontal (see Figure 3.2).

Errors in the quantization tables, defined in the Define Quantization Table header segment, affects the contrast and the colours of an image. The biggest effect is achieved when the DC component in the luminance quantization table is wrong. Usually the quantization factor for the DC component is low. We have found empirically that 4 is a common value for the luminance DC component, as well as the chrominance DC components. If the luminance DC component is high, the contrast of the image becomes higher. In Figure 4.10 the luminance DC component is set to  $0xFF$ , i.e. 255.

If instead the luminance quantization table DC component is set to  $0x00$  the result is a flat, low contrast image. The quality is higher than for the value  $0xFF$ , but this is not surprising since the original value is  $0x04$ .

The DC component of the chrominance quantization table greatly affects the colouring of the image. The Cb and Cr quantization tables affect different colours, but otherwise the effect of setting any of their DC values to  $0xFF$  is the same for both tables. Figure 4.11 shows the effect of the red, Cr, chrominance DC component being set to  $0xFF$ .

Figures 4.10 and 4.11 show the maximum effect on an image an error in a quantization table will have. If the highest frequency is affected the effect on the image is not visible. Even if we set the second lowest frequency to  $0xFF$  the effect on the image is much lower than for the DC component. We can therefore conclude that as long as the quantization tables have the correct length and format, their effect on the quality of the image is limited, although on the border of severe in the case of the luminance DC component.

### 4.3 Define Huffman Table

The DHT marker segment contain the Huffman tables used for compression of the image. The JPEG standard allows the use of arithmetic entropy coding too,

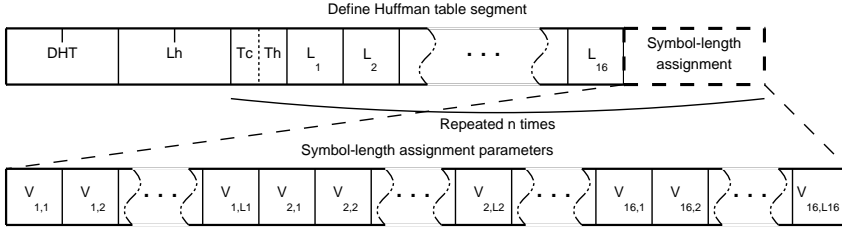


**Figure 4.10:** The effect on a JPEG image of setting the luminance quantization table DC component to  $0xFF$ .



**Figure 4.11:** The effect on a JPEG image of setting the chrominance quantization table DC component to  $0xFF$ .

### 4.3. DEFINE HUFFMAN TABLE



**Figure 4.12:** The DHT marker segment. The last bytes in the marker segment are repeated  $n$  times, where  $n$  is the number of Huffman tables used.

but our work only concerns non-differential Huffman entropy coded images. When coding an image the standard does not specify any particular Huffman tables to use, but in reality we have found that all cameras included in our work use the same Huffman tables, which are the example tables [13, pp. 149–157] from the standard.

The outline of the segment can be seen in Figure 4.12 and contains the following information:

**Define Huffman Table (DHT)** Marks the beginning of the Define Huffman Table marker segment, indicated by the hexadecimal value of  $0xFFC4$ .

**Huffman table definition length (Lh)** The length of the DHT marker segment, excluding the first two bytes. The formula  $2 + \sum_{t=1}^n (17 + m_t)$  is used to calculate the length parameter, where  $m_t = \sum_{i=1}^{16} L_i$  is the number of parameters following the 16  $L_i$  values for table  $t$ .

**Table class (Tc)** Specifies whether this is a DC table, indicated by value 0, or an AC table, indicated by 1.

**Huffman table destination identifier (Th)** Indicates which table this is, out of two possible for baseline sequential DCT JPEGs.

**Number of Huffman codes of length  $i$  ( $L_i$ )** Specifies the number of Huffman codes for each of the 16 possible lengths allowed by the standard [13]. The allowed values lie in the range of 0 to 255.

**Value associated with each Huffman code ( $V_{i,j}$ )** Gives the value associated with each Huffman code of length  $i$ . The Huffman coding model determines the meaning of each value. The values range between 0 to 255.

The Huffman tables definition is sensitive to errors, but if the format is correct and the standard is followed, it is still possible to change the definition without corrupting the image. If the maximum number of possible Huffman codes of a specific length is not exceeded, it is possible to redistribute the number of Huffman codes of length  $i$  specifications. It is also possible to change the values associated with each Huffman code, as long as they allow the scan to be decoded.

To illustrate the effect of an error in a Huffman table the image in Figure 4.13 has been given new Huffman table definitions taken from a different image with a completely different Huffman tables definition. The two images are not related, one is captured by a Fuji digital camera and the other is professionally scanned from a film negative. In our database all cameras used the same Huffman table definition, which is taken from the JPEG standard [13, pp. 149–157].

The two Huffman tables definitions<sup>1</sup> differ in many positions, but the image is still not corrupted beyond decoding. Due to the change of the tables the decoder reports 17 extraneous bytes before the EOI marker, but still shows the image. This can be done because of the sequential decoding of the scan used in baseline sequential DCT JPEG images.

## 4.4 Define Restart Interval

The Define Restart Interval (DRI) marker segment should always be 2 + 4 bytes long. Figure 4.14 specifies the DRI marker segment and the information held in the marker segment has the following meaning:

**Define Restart Interval (DRI)** Marks the beginning of the Define Restart Interval marker segment, indicated by the hexadecimal value of 0xFFDD.

**Define restart interval segment length (Lr)** The length of the marker segment, excluding the first two marker bytes. The value should always be 4.

**Restart interval (Ri)** Gives the number of MCUs in each restart interval. There can be 0 to 65535 MCUs in a restart interval, but there should always be the same number of MCUs in each interval, except in the interval ending the image. The ending interval can have any number of full MCUs, which in reality means up to Ri MCUs.

The existence of this header marker segment, having a non-zero restart interval value, indicates the use of RST markers in the scan. For every  $Ri^{\text{th}}$  MCU in the scan there should be a marker 0xFFD $x$ , where  $x_{i+1} = (x_i + 1) \bmod 8$ , starting at 0xFFD0.

The definition of the restart interval also affects the quality of an image. This is a short restart marker segment, which only contains one value. If we set the restart interval too low, the decoder uses that number of MCUs after each restart marker and skips the rest until the next restart marker in the scan. This gives the effect shown in Figure 4.15 where the original 4 MCU long restart interval has been reset to 3.

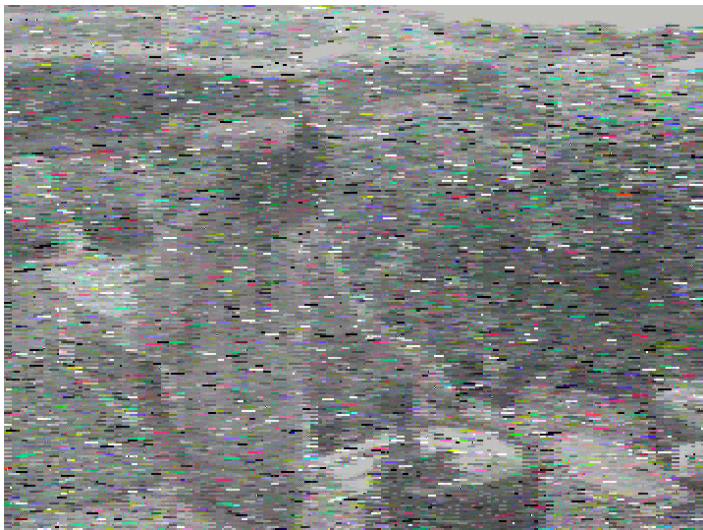
Since not all the data in the scan is used, the image is shortened by the amount of skipped MCUs. This can be seen in Figure 4.15, which is 75% of its original height.

---

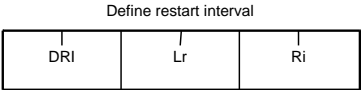
<sup>1</sup>The raw data files and source code can be found at <http://www.ida.liu.se/~iislab/security/forensics/material/> or by contacting the author at [g-makar@ida.liu.se](mailto:g-makar@ida.liu.se)



4.4. DEFINE RESTART INTERVAL



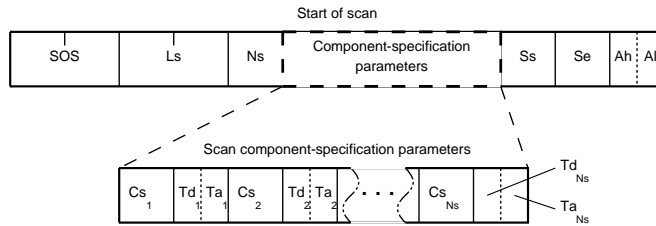
**Figure 4.13:** The effect on a JPEG image having its original Huffman tables definition exchanged by the definition of a different image.



**Figure 4.14:** The DRI marker segment



**Figure 4.15:** The effect on a JPEG image having its original restart interval definition shortened from 4 MCUs to 3.



**Figure 4.16:** The Start Of Scan (SOS) marker segment. There can be 1, 3, or 4 scan component specifications in an image.

If instead the restart interval definition value is set too high the image gets specks at repeated intervals, coming from the missing extra MCUs the decoder tries to decode. These two types of errors are easy to spot, but not very likely, since it is trivial to find the correct restart interval definition from the scan.

Even though the multi-layered coding of the scan, using Huffman codes and values interleaved, is easily corrupted, the use of restart markers improves the robustness of the scan. The level of robustness is connected to the restart marker interval; the shorter the interval, the more robust the decoding of the scan. Depending on the decoder the extra spurious bytes at the end of the scan, before the EOI, resulting from the wrong restart interval definition, will be reported. It is still possible to decode the image, possibly lacking the pixels of some of the last MCUs.

## 4.5 Start of Scan

The SOS marker segment precedes the scan of an image and connects the components of the image to the quantization and Huffman tables specified earlier in the header section. The components are interleaved if there is more than one component in the image, i.e. it is a colour image.

The outline of the segment can be seen in Figure 4.16 and contains the following information:

**Start Of Scan (SOS)** Marks the beginning of the Start of Scan marker segment, indicated by the hexadecimal value of `0xFFDA`.

**Scan header length (Ls)** The length of the SOS marker segment, excluding the first two bytes. The formula  $6 + 2 \times Ns$  is used to calculate the length of the segment.

**Number of image components in scan (Ns)** Defines how many source image components there are in the scan. The number of components given by Ns should match the number of  $Cs_j$ ,  $Td_j$ , and  $Ta_j$  there are. Valid values are in the range 1 to 4.



**Scan component selector ( $Cs_j$ )** Uniquely identifies and orders the  $N_f$  components in the image. Each  $Cs_j$  shall match a  $C_i$  in the SOF header, and follow the same order. If there is more than one image component, the components should be interleaved in the MCUs, in increasing order of  $j$ . The following restriction applies if  $N_f > 1$

$$\sum_{j=1}^{N_s} H_j \times V_j \leq 10,$$

where  $H_j$  and  $V_j$  are the horizontal and vertical scan rates, specified in the SOF header. The marker parameter can have a value between 0 and 255, but should be part of the set of  $C_i$  specified in the SOF header.

**DC entropy coding table destination selector ( $Td_j$ )** Specifies which of the two possible DC Huffman tables to use when decoding component  $Cs_j$ . Valid values are 0 and 1.

**AC entropy coding table destination selector ( $Ta_j$ )** Specifies which of the two possible AC Huffman tables to use when decoding component  $Cs_j$ . Valid values are 0 and 1.

**Start of spectral or predictor selection ( $Ss$ )** This parameter holds the index of the first DCT coefficient in the zig-zag order to be used when coding an image. For baseline sequential DCT JPEGs it should be set to 0.

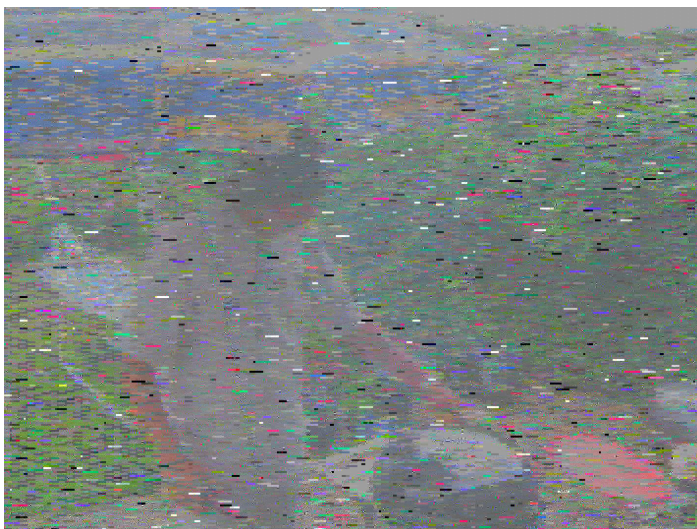
**End of spectral selection ( $Se$ )** This parameter holds the index of the last DCT coefficient in the zig-zag order to be used when coding an image. For baseline sequential DCT JPEGs it should be set to 63.

**Successive approximation bit position high ( $Ah$ )** This parameter should be set to 0.

**Successive approximation bit position low ( $Al$ )** The parameter has no meaning for baseline sequential DCT JPEGs and should be set to 0.

The information that can be changed in the start of scan header segment is the Huffman tables' connections to the luminance and chrominance components. It is only possible to exchange a luminance DC or AC table with the corresponding table for the chrominance. Since the DC table for the luminance has the greatest effect on the image quality, Figure 4.17 shows what happens when the luminance DC Huffman table is set to point to the chrominance DC table.

If we set one of the chrominance components' DC table to point to the luminance DC Huffman table, the colours are affected, with a pattern of corrupted MCUs. The quality of the image is still good and it is possible to recognise the subjects in the image. A complete exchange of all Huffman tables gives the result shown in Figure 4.18.



**Figure 4.17:** The effect on a JPEG image where the luminance DC Huffman table is set to use the chrominance DC Huffman table.



**Figure 4.18:** The effect on a JPEG image where all Huffman table pointers in the Start of Scan header segment are wrong.

The start of scan header segment is static for all 9958 valid colour baseline sequential DCT JPEG images in our database. The formal specification of the start of scan header marker only allows a few different alternatives for baseline sequential DCT JPEG images. Consequently the marker's position as the last header marker segment before the scan, together with its static appearance, makes it suitable to use as an indicator of the first fragment of an image. Even if the start of scan segment is not complete, it should be possible to use as a signature of the start of a JPEG image scan. For a colour baseline sequential DCT JPEG image the header segment's typical hexadecimal byte sequence is

```
FF DA 00 0C 03 01 00 02 11 03 11 00 3F 00.
```

### 4.6 Combined Errors

The errors described in the previous sections can be combined in many ways. Some combinations will even cancel out each other, as for example a change of the Huffman table indices in the start of scan header segment can be reversed by changes to the Huffman table definition segment. In other situations the effects will enforce each other and render the image almost unrecognisable.

There are however only a limited number of unique effects, although their level of severity may vary. It is therefore possible to recognise and separate their effects even in cases where several errors are combined. In severe cases it might be possible to perform brute force searches for the factors in the combined effect.

### 4.7 Using an Artificial JPEG Header

It is possible to use an artificial JPEG header, preferably taken from another image captured by the same digital camera or created by the same application. An important factor is the Huffman code definitions, but as can be seen in Section 4.3, it is possible to exchange the Huffman tables of an image with the tables from a completely different image, as long as the new tables are valid Huffman tables.

However, all cameras in our database use the same Huffman tables definition, which happens to be the example tables [13, pp. 149–157] given in the JPEG standard. The scanner uses its own Huffman table definitions, but they are the same for all images. The thumbnail images embedded in the scanner images' headers use the standard tables, though.

A possible reason for the cameras using the standard Huffman tables is execution speed and low power consumption. It is costly both in time and power consumption to let the camera calculate unique Huffman tables for every image. The downside is an increased file size, but since the consumer has to pay separately for the storage media, the camera manufacturers do not have any incentive to keep the file sizes down. It might therefore be safe to assume that all (consumer grade) cameras use the same standard Huffman tables.

The effect of using the wrong quantization tables is not that severe, especially as the values in the tables are often found in the hexadecimal range of  $0 \times 00$  to  $0 \times 20$ . The lower values often appear at the top of the table and then slowly increase towards the end. It is therefore possible to create completely artificial quantization tables giving acceptable results, even though using tables from other images is probably safer.

Some cameras create two quantization table definitions, one for the luminance table and one for the chrominance tables. Their lengths are the same, though. Consequently it does not matter if one single definition header segment is used in the artificial header, even for images having split definitions from the beginning.

The restart interval definition header segment is not always used. All images using restart markers in our database use the interval 4. But our database only contains images from three cameras using RST markers and the value may therefore very well be different for other cameras.

The horizontal and vertical sampling factors can vary. Each parameter can either have the value 1 or 2. For all images in our database only the luminance settings vary, not the chrominance ditto, which keeps the number of possible settings down. The sum of the sampling factors should always be between 2 and 4. It is therefore possible to use a brute force model, simply trying three different combinations. Most of the images in our database have a horizontal sampling factor of 2, and a vertical factor of 1.

The settings for the number of lines and the number of samples per line of an image can vary and finding the correct setting for them may require some work. As shown in Table 4.1 the width and height of the images in our database either relate to each other as  $\frac{3}{2}$  or  $\frac{4}{3}$ , or the inverses thereof. This fact, combined with knowledge of the common number of pixels in cameras, can be a help in finding the proper image width. The height is less important; as long as there is enough data (pixels) in the scan it is enough to test a few settings, the image will not be distorted. The decoder will give an error reporting too few scan lines if the height is too large.

The conclusion to be made is that artificial headers will give acceptable results in most cases, although perhaps not give the original image back. From our experiments we have found that there are only a few crucial parameters that change between images, regardless of the camera make.

## 4.8 Viewing Fragments

If there are restart markers embedded in the scan part of an image it is possible to use an artificial header with properly adjusted height to view the fragment. The hardest part is to collect a large enough amount of consecutive fragments, since a 8 KiB fragment covers approximately  $3000 \times 8$  pixels, i.e. one line in the scan. To be able to see a large enough part of an image we need around 10 scan lines.

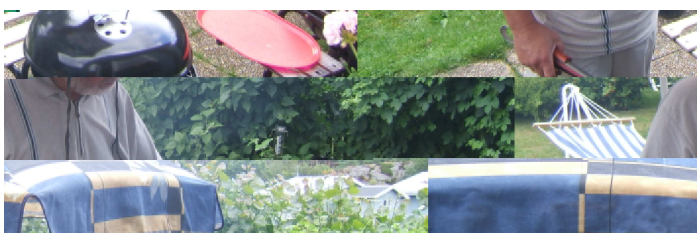
Figure 4.19 shows an image made from fragments 146 to 301 of our test image, resulting in a 624 KiB file. We have used an artificial (but correct) header and found the height to be 880 pixels by trial and error. Otherwise there are no adjustments. The first fragment gives a few corrupted MCUs in the upper left corner of the image. They come from the last part of the restart interval that was cut in half when the fragment was created. Since the first restart marker in fragment 146 is `0xFFD1` the image is wrapping around, i.e. horizontally displaced.

If the sequence of fragments joined together is not correct the result is an image made up of patches of the original image, or the images included in the set of fragments. It is easily recognisable from the sharp horizontal borders between the different fragment sequences. A typical example can be seen in Figure 4.20.

The fragments used in Figure 4.20 come from three sequences of fragments from the test image. The sequences are from top to bottom; fragments 310–360, fragments 146–201, and fragments 15–66. As can be seen they all give different horizontal displacements of the image.



**Figure 4.19:** A correct sequence of fragments with an artificial header adjusted to fit the height of the resulting image.



**Figure 4.20:** An incorrect sequence of fragments with an artificial header adjusted to fit the height of the resulting image. The typical sharp horizontal borders between the different fragment sequences are clearly visible.

## Chapter 5

# Discussion

In this chapter we discuss the research and the results presented in the previous chapters. We end the chapter with some conclusions drawn from our research.

### 5.1 File Type Categorisation

The results indicate that the file type categorisation method and its algorithms are well suited for high information entropy file types. One reason might be the fact that some lower information entropy files contain sub-parts of different data types. A typical example is the Windows PE files, which often contain machine code, tables and possibly text strings in different parts of a file. By using a complete Windows PE file for training we include all of these sorts of data into the centroid, making the centroid more general. It might therefore generate false positives for any of the sub-part data types.

The partitioning of files is clearly a problem, but might possibly be fixed by treating the sub-parts of a file as separate entities. This approach will however require the parts of a specific file type to be found and isolated in advance when a centroid is to be created. To be able to make the partitioning of the file type data we have to find the exact borders of the sub-parts before we can proceed.

If all sub-parts of a file can be found and isolated, we then have to decide how the sub-part centroids should be combined into a metacentroid of the file type in question. A solution might be to give each sub-part centroid a weight based on its share of the total amount of bytes used for centroid creation. Another possible solution is to modify the file type categorisation method to use a sliding window approach, where the size of the window equals the smallest possible unique sub-part of a file.

The 2-gram algorithm is the only algorithm able to detect executable files with a higher detection rate than simply guessing the correct file type. The results for the Windows PE files are better than the results presented in [12], because we have cleaned the files used to create the centroid. The previous executable file centroid contained both JPEG, PNG and GIF images, these were

removed this time.

By cleaning the executable files, i.e. using (partially) artificial data, we get a more exact model of pure executable code, but at the same time we dissociate the centroid from a real world situation. This duality burdens all file types where there might be other file types embedded in the binary data. For example Zip files may contain JPEG images, because the algorithm does not try to compress already compressed files. If we remove this embedded data, we get more homogeneous file type data to create our centroids from, but once again, what we might then face in a real life situation may not be what we have trained our detectors to handle. A large set of centroids to use for categorisation will to some extent compensate for this.

The main difference in the performance of the 2-gram algorithm for different file types may possibly be explained by its ability to capture regularities in files. For example, JPEG images without RST markers should not contain any 2-grams starting with `0xFF`, except for `0xFF00` and `0xFFD9`. This is a strong regularity and consequently the results from running the 2-gram algorithm on that file type are very good indeed. For the JPEG with RST marker file type the results are not that good. This file type allows, but does not require, some occurrences of 2-grams between `0xFFD0` and `0xFFD7`. This relaxes the 2-gram algorithm, and its performance degrades.

Another feature that affects the performance of the 2-gram algorithm is the necessary scaling of the size of the centroid creation blocks relative to the size of the detection fragments. Since we use 1 MiB blocks to create the centroid, but 4 KiB fragments for categorisation, we introduce instability into the process. Every deviation from the centroid by a sample produces a disproportional effect on the distance metric. As long as the standard deviation for a specific 2-gram is low, its effect on the distance measurement is decreased, but for 2-grams with a high standard deviation the effect can be considerable.

The use of a file type specific rule set in combination with 1-grams only improves the results for the stand-alone algorithms, i.e. when the BFD or RoC are not combined. The signed difference value extension of the RoC algorithm affects the results more than the rule set extension does. The file type of the centroid has a remarkable effect on the results. The 2-gram algorithm is good at detecting fragments of JPEG images lacking RST markers and MP3 file fragments. When it comes to other file types the results are worse, it has the highest level of false positives at the 50% detection level for AES encrypted GPG files and is not even included on the ROC curve plot for Zip files. The 1-gram algorithms and their corresponding centroids seem to be better at describing a file type without any specific regularities, while the 2-gram algorithm is overly sensitive to a lack of regularities. However, the 2-gram algorithm is much better than all the 1-gram algorithms when dealing with files containing strict regularities, such as JPEG images lacking RST markers.

Consequently, if we have specific and known features related to a certain file type, we should preferably use the 2-gram algorithm. If we need to work with less regular and featureless file types, we should go for a 1-gram algorithm.



## 5.2 Fragment Reconnection

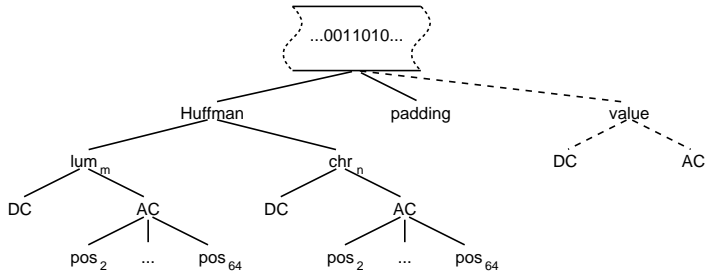
The image fragment reconnection method depends heavily on the existence of RST markers in the scan part of an image. Without the RST markers it is hard to synchronise the decoder with the stream, especially as the Huffman codes and embedded amplitude values often can be decoded incorrectly for quite long sequences. However, the probability of incorrectly decoding a full 4 KiB fragment is very low, based on our empirical evaluations. Important features to be used for decreasing the false positives rate of the decoding are the sampling factor, the relation between the length of the luminance and chrominance non-zero value vectors, and the probability of a certain non-zero vector pattern. Also the actual values of the vectors can be used; small variations of the low frequency amplitudes are more probable than high variations, in normal, every-day images at least.

By varying the sampling factors and the starting bit of a fragment a brute force model of synchronising the decoder with the scan might be achievable even for scan parts of images without RST markers. To do this the maximum decoding length for each iteration has to be recorded together with the non-zero amplitude value vector parameters. These parameters then can be used to find the most probable starting point of the first MCU in the fragment. If we know where the first MCU starts we can use a slightly modified image fragment reconnection method; the checking of the length of the RST intervals should be abolished. There is no need to calculate the differences between consecutive DC values, since they are already given as delta values.

Since the image fragment reconnection method relies on finding the start of a MCU, i.e. the first DC luminance Huffman code, to work, every bit position starting from the beginning of a fragment needs to be processed to see whether it is the first in a MCU or not. The first bits of a fragment can represent different attributes in the JPEG scan. Figure 5.1 illustrates this. For JPEGs without RST markers each bit may belong to a Huffman code, its accompanying amplitude value, or a zero byte padding after a  $0xFF$  byte value. The Huffman codes can either be part of a luminance table, there can be one, two, or four tables in a row, or it can be coming from one of two chrominance tables. The codes in the tables can in turn belong to a DC or an AC table, where AC codes are up to 63 times more frequent than DC values.

To find a bound on the performance penalty introduced by the iterative search for the start of the first MCU, we use the longest possible Huffman codes that represent no consecutive zeros and the longest possible amplitude value representation. In this way we find the maximum theoretical length of a MCU to be 1820 bytes, or 14560 bits. Each bit position needs to be tested with one, two or four luminance tables, thus the maximum number of iterations before a modified image fragment reconnection method has found the start of an MCU is  $14560 \times 3 = 43680$ . Consequently it should be possible to use the proposed method on any standard JPEG image fragment, without any severe performance degradation.

The real iteration limit is lower than 43680, because it is not possible to use



**Figure 5.1:** The possible functions of the starting bits of a non-RST marker JPEG fragment. There are one, two, or four luminance tables and two chrominance tables.

only the maximum positive value for the DC frequency. The DC tables give the delta value from the previous DC value of the same component<sup>1</sup>. By using the maximum possible positive value once, the next DC value will be out of range if we once again use a maximum positive delta value for the same component.

The image fragment reconnection method can handle fragments as single entities without any order known in advance. In reality, file systems will store parts of files in consecutive order, although possibly wrapping over the end of the hard disk. This simplifies matters a lot, since we then only need to find the first fragment and then can build from there. The situation is similar for network TCP packets, where we have packet header fields giving the ordering of the packets. The image fragment reconnection method might appear as unnecessarily advanced for such situations, but if the ordering is lost for some reason, it can still reconnect the fragments.

The description of the modified image fragment reconnection method to be used for image fragments lacking RST markers is a theoretical idea of one way to generalise the method. We have not performed any tests or other empirical evaluations of the modifications, hence what we suggest is only a proposal for a method.

### 5.3 Viewing Fragments

The possibility to view (parts of) the original image is heavily dependent on the amount of data it is possible to collect. Even though it is enough with as little as one MCU worth of data, the resulting image is too small to give much viewable information. Still, the smaller the image, the larger the amount of image information in each MCU. Since the size of images is a limiting factor on the Internet, because of the relatively small size of the current computer displays, a single network packet payload can still contain enough image data to render it viewable. Thus the difference between a network packet and a hard disk fragment is not as big as the difference in size might indicate.

<sup>1</sup>Luminance or one of the two chrominance tables, that is.

The use of an artificial JPEG header makes it possible to view one version of the original image. The question is if the loss of quality disqualifies the image from being used as evidence in a court of law. What we can recreate is an image where the main objects in the image are visible and correct, with a varying degree of noise. There is however a risk that the noise gives errors in the details of the image, and therefore decreases the evidence value of the image.

An alternative to trying to recreate as much of an image as possible, with the negative effects of increased noise and potential errors in the image details, is to recreate the image using only the DC luminance values. This method also avoids the use of a full artificial header, the only thing needed is a Huffman table for the decoding of the values. We need only to know the length of the AC values interleaved in the data stream, we do not need to decode them. Such an image will show the outlines of the objects in the image. The requirements on the viewer application is not high, it is enough that it can display a surface plot of a matrix, thus any 3D-plotting software is enough.

## 5.4 Conclusion

This work can be divided into three sub-parts logically describing the work flow of finding and reconnecting data fragments, especially of baseline sequential DCT JPEG images. It is possible to use the sub-parts individually, but they are meant to work together.

The first part describes how the file type can be found from fragments, without access to any metadata. To do this our method for categorisation of the file type of data fragments is presented. The fragments can be taken from a crashed hard disk, a network packet, a RAM dump, or any other situation where there is a lot of unknown data to be processed. The file type categorisation method works very well for JPEG images, but also satisfactory for handling other file types.

The file type categorisation method comprises three algorithms that should be chosen differently depending on the aim of the application. When robust function and fast execution is important, and the detection rate can be sacrificed, the BFD and RoC algorithms are suitable. When JPEG images are to be found the 2-gram algorithm is the best. The BFD and RoC algorithms are better at categorising file types without any clear structure; the 2-gram algorithm may be too sensitive for file types having byte frequency distributions with high standard deviation rates.

The second part describes how to reassemble fragmented baseline sequential JPEG images having RST markers in the scan. The image fragment reconnection method uses the internal structure of fragments at the DC luminance amplitude value level to form fragment pairs, measuring their fragment joint validity. By sorting the pairs in fragment joint validity order, approximately 97% of an image can be restored automatically, if all fragments are available. Repeated use of the image fragment reconnection method on the remaining fragments should quickly restore the full image.

When there are fragments from several images to be reconnected the first iteration of the image fragment reconnection method correctly combines approximately 70% of the pairs. Even in this case the method is to be used iteratively.

The third part presents techniques to enable viewing of reassembled JPEG images, the important JPEG header marker segments and how errors in them affect an image. There are in reality only two parameters that are crucial for the decoding of an individual image. These are the size of the image and its sampling rate. For all other parameters artificial data can be used, preferably a header from another image, but even fully artificial constructions are possible. This gives us the ability to view fractions of images, where not all fragments are found, or where a full reconnection of the fragments for some reason is not possible.

An important factor for the ability to view image fragments is the use of RST markers in the scan. The first few MCUs in the reconnected fragments may be corrupt, but when the decoder encounters the first RST marker it can correctly decode the following MCUs. If the height setting in the header is too large the decoder will fail, but by starting at a low setting and iteratively increasing the image height the setting can be optimised for the partial image.

Consequently it is possible to identify, reconnect, and view a fragmented JPEG image, as long as it uses RST markers. This ability is crucial to the police and will help them in their search for child pornography. The methods and tools make it possible to scan even heavily fragmented hard disks and network traffic for remnants of illegal images. The remnants then can be used to recreate parts of an image, which can be equipped with an artificial header to allow the image parts to be displayed.

## Chapter 6

### Related Work

The PAYLoad-based anomaly detector for intrusion detection (PAYL) [28, 29] is related to our file type categorisation method through the use of a byte frequency distribution algorithm. The system uses a distance metric called the *simplified Mahalanobis distance* to measure the similarity between a sample and a centroid. The metric is based on two vectors, called  $x$  and  $y$ , representing the sample and the centroid respectively, and the standard deviation,  $\sigma_i$ , of each byte  $i$  in the centroid. A smoothing factor  $\alpha$  is used to avoid division by zero when  $\sigma_i = 0$ . As can be seen in Equation (6.1) this is similar to a linear-based distance metric between the sample and a model, weighted by the standard deviation of the byte frequency distribution.

$$d(\vec{x}, \vec{y}) = \sum_{i=0}^{n-1} (|x_i - y_i| / (\sigma_i + \alpha)) \quad . \quad (6.1)$$

PAYL has been implemented using different approaches for how to handle the centroid creation. The later implementations use  $k$ -means clustering in two steps to reduce the memory requirements and increase the modelling accuracy.

The main differences between the file type categorisation method and PAYL are the modelling of the data, the distance metric, and the centroid creation process. PAYL is based on the byte frequency distribution of the data stream, which does not take the ordering of the bytes into consideration. The file type categorisation RoC algorithm measures the rate of change in the data stream and can therefore distinguish between data blocks having similar byte frequency distributions, but different ordering of the bytes.

The simplified Mahalanobis distance metric used by the PAYL intrusion detector suffers from an inability to separate samples having a few large deviations measured against the centroid from samples having several smaller deviations, in cases where the mean and standard deviation values of the centroid are almost uniform. This is the case when dealing with compressed or encrypted data, for example JPEG or .zip files. The file type categorisation method uses a quadratic

distance metric which is better at handling almost uniform byte frequency distributions. Due to the weaknesses of the distance metric the PAYL method has to use a more complex and computationally heavy centroid creation process than the file type categorisation method.

The anomaly based intrusion detector Anagram [30] has evolved from PAYL and uses higher order ( $n > 1$ )  $n$ -grams to detect malicious content in network traffic. The creators of Anagram use both normal and malicious traffic to detect intrusion attempts. To minimise the memory requirements two Bloom filters are used to store the  $n$ -grams, one for each traffic type. The  $n$ -grams appearing in the malicious Bloom filter are weighted by a factor 5 when the anomaly score is calculated. To increase the resistance to mimicry attacks Anagram is suited with a randomised testing function, which chooses several, possibly interleaved, sub-parts of a packet payload for anomaly testing. The reported results are very good with a detection rate of 100% at a false positives rate of less than 0.03%. The metric used to calculate the anomaly score of a packet counts the number of new  $n$ -grams,  $N_{new}$ , and the total number of  $n$ -grams,  $T$ , giving:  $Score = N_{new}/T$ .

The main difference between our file categorisation method and Anagram is the latter method's use of high-order  $n$ -grams. In our experiments with the 2-gram algorithm we noticed that it had problems categorising encrypted file types. Anagram is currently only applied to text and executable code, which have a lower entropy than compressed and encrypted files. It would therefore be interesting to see how well Anagram can handle high entropy file types. Since we have not seen any such evaluations we cannot judge its efficiency compared to our file type categorisation method.

A system called Statistical PARSEr (SPARSE) [31, 32], created by the research group which invented PAYL and Anagram, is used to detect malware hidden in Microsoft Word documents. SPARSE incorporates Anagram as one of its detectors. Documents to be scanned are parsed and data objects are extracted. These are then fed to a static detector (Anagram). If the static detector labels a object as negative the object is given to a dynamic detector executing the object and looking for malicious system events. If any of the detectors report a positive finding a "malcode locator" searches each parsed section for the malicious code.

SPARSE is focused on Word documents and use of multi- $n$ -gram algorithms through Anagram. The incorporation of Anagram gives SPARSE the same detection characteristics as Anagram, thus also the same differences compared to our file categorisation method.

The group that created PAYL and Anagram has also used the byte frequency distribution algorithm for creating *fileprints* [33, 34, 35], which are used to determine the file type of unknown files, in the same manner as the first version [9] of the file type categorisation method. The fileprint method has borrowed its base from the byte frequency distribution of PAYL, together with the simplified Mahalanobis distance metric. The method is further improved by more advanced centroid creation methods, of which the two-layer method of PAYL is one. Furthermore the authors experiment with truncating the files to be categorised by the fileprint method. They find that by using only a few hundred bytes from

the beginning of the files they get higher detection rates than when applying the method to full-length files.

Since the fileprint method and PAYL were developed in parallel these two methods also share their main differences relative to our file type categorisation method. In addition to the differences in the modelling of the data, the distance metric, and the centroid creation process, the fileprint method also differs from the file type categorisation method in the previous method's implicit dependence on file headers through the truncation of files to be categorised.

Our file type categorisation method is meant to do what the current tools cannot do, namely categorise data fragments without having to rely on header data and file blocks being stored consecutively. In other words our file type categorisation method can work regardless of the state of the file system or degree of fragmentation. By using only the first few hundred bytes of a file the fileprint method in reality becomes a reinvention of the Unix *file* command.

McDaniel and Heydari [36] use the concept of n-grams in a file type categorisation method. They use different statistical metrics applied to single bytes in files to make fingerprints of files. McDaniel and Heydari's main source of data is the header and footer parts of files. They try to consider the ordering of the bytes in files through the use of what they call *byte frequency cross-correlation*. One example of bytes with high byte frequency cross-correlation is the Hyper-Text Markup Language (HTML) tag markers "<" and ">". The McDaniel and Heydari method differs from the 2-gram method through the use of single bytes, as well as their method's dependence on file header and footer data.

Yet another n-gram based method is presented in a paper by Shanmugasundaram and Memon [37]. They use a sliding window algorithm for fragmented file reassembly using n-gram frequency statistics. They let the window pass over the edges of the fragments and then combine those fragments having the highest probability of being consecutive based on the frequency of the n-grams found in the window. Their use of n-grams frequency statistics is similar to the 2-gram method, but they apply the method in another way and require all fragments of a file to be known in advance.

Shamir and Someren [38] use the entropy of hard disk data fragments to locate the cryptographic keys stored on disk. They use a sliding window of 64 bytes and count the number of unique byte values in the window. A cryptographic key will generate a higher entropy than other binary data and in that way it is possible to differentiate keys from other data.

The Shamir and Someren method differs from the file type categorisation method in that their method uses smaller data blocks and does not need any centroid to measure new, unknown samples against. Also the area of usage differs, although the file type categorisation method could be used for similar tasks. On the other hand, the Shamir and Someren method cannot be used for identifying the file type of an unknown data fragment and is specialised at detecting cryptographic keys.

Garfinkel [39] presents a method called Bifragment Gap Carving (BGC), which can reconnect a file fragmented into two fragments separated by one

or more junk hard disk sectors. The algorithm requires that the fragments are stored in order and the start and end of the fragmented file can be found. The algorithm runs in quadratic time for single object carving and  $O(n^4)$  time if all bi-fragmented files of a specific type is to be carved.

To verify that a file is appropriately carved Garfinkel has developed a pluggable validator framework, implementing each validator as a C++ class. The framework has several return values, apart from the standard *accept* or *reject*.

The BGC method relates to the file type categorisation and image fragment reconnection methods in combination. A drawback of the BGC method is that it, as the name implies, can only handle files fragmented into two parts. The main idea behind the algorithm gives it a high complexity, which makes it scale badly. The file type categorisation method can find all fragments of a specific type in linear time, and then the image fragment reconnection method can be used to reconnect them in much less than  $O(n^2)$  time. If we assume ordered fragments, the reconnection can be done in linear time. The file type categorisation and image fragment reconnection methods do not need access to any headers or footers to work, hence they have a more generic application area than the BGC method.

Cohen [40] describes two semantic carvers for Portable Document Format (PDF) and Zip files. The carvers utilise the internal structure of the two file types to identify and reassemble fragments. For text based file types, such as PDF, this is straightforward, but for binary file types there might be large sequences of uniformly distributed byte values. Cohen does not explain how fragments not containing any header-related information can be identified. The article implies that the fragments need to be consecutively stored and possibly the number of other fragments need to be low. A file type specific discriminator should be used to find the correct sequence of fragments for each file, taken from a set of possible sequences of fragments.

The method Cohen proposes uses only information related to the headers and internal structure. Therefore the method in reality is limited to ordered structures and low fragmentation environments. The file type categorisation and image fragment reconnection methods are designed to work in any circumstances and environment. They therefore do not make any assumptions about the surrounding fragments, and only need information that can be retrieved from the fragments themselves to work.

Erbacher and Mulholland [41] have studied the problem of basic steganography, where complete and unencrypted files are hidden within other files. To do this they use 13 different statistical measures calculated over a sliding window to detect deviations in files of various formats. They also experiment with varying sizes of the sliding window. The best results are given by the byte value average, kurtosis, distribution of averages, standard deviation, and distribution of standard deviations, using sliding window sizes between 256 and 1024 bytes. The results Erbacher and Mulholland get show that files are not homogeneous streams of data, but instead vary significantly throughout and have clearly visible subsections.



The work by Erbacher and Mulholland have a slightly different approach than our work and do not aim at recognising file types, but to see deviations within files. The statistical metrics used by Erbacher and Mulholland are related to the BFD and RoC algorithms of the file type categorisation method, but the smaller sliding window size used by Erbacher and Mulholland, and the fact that they do not really model the file types, separate the two methods from each other.

Memon and his research group [42, 37, 43, 44] have looked at how to re-assemble fragmented bitmap images. To do this they have assigned weights to each possible fragment pair and used different path optimising algorithms to find the correct fragment pair sequence. They assume each fragment contains at least an image width of data, and that all fragments are available. The header fragments provide their weight-assigning algorithms with needed information on image width and resolution. The weights are calculated for each possible image width. Memon and Pal say that with some additional steps of decompression, denormalization, and inverse DCT their method can handle JPEG images, but they do not prove it.

The image fragment reconnection method is similar to Memon's and Pal's method at a high level, both of the methods assign weights to fragment pairs and optimise the sum of the fragment chains. There are differences between the methods too; the image fragment reconnection method does not need to have a header fragment to be able to determine the width of an image. It is also successfully applied to JPEG images, and is more efficient, since it uses pre-processing to decrease the set of possible fragment pairs needed to be assigned weights and optimised.

Memon and his group have developed a network abuse detection system, called Nabs [45], that can identify whether the payload of a network packet is uncompressed, compressed, or encrypted. They used statistical measures from the time domain, the frequency domain, and some higher order statistics and applied a Sequential Forward Feature Selection (SFFS) algorithm to find the best parameters. The parameters are, in order of decreasing importance, entropy, power in the  $0-\frac{\pi}{8}$  frequency band, mean, variance, and mean and variance in the  $\frac{\pi}{2}-\pi$  frequency band. These parameters give an accuracy of approximately 84%, where the accuracy is calculated as the number of correctly categorised samples divided by the total number of samples.

The Nabs detection system shows that the important features are not the most complex, hence the aim of the file type categorisation method to use simple, fast and robust algorithms for detection is in line with the research by Memon and his group. The important parameters presented by Memon and colleagues would be interesting to test with the file type categorisation method. The difference between the two methods is that Nabs only tries to identify whether a fragment is compressed or not, or encrypted, while the file type categorisation method is meant to identify the exact file type of a fragment.

Shannon [46] uses entropy and the proportion of human readable ASCII characters to identify the type of files. The calculations are performed on com-

plete files, not fragments. The work is interesting, even though it is not directed at categorisation of data fragments. The main difference between this method and the file type categorisation method is the latter method's focus on a lower abstraction level through data fragments.

Hall and Davis [47] use the compressibility and entropy calculated over a 90 byte large sliding window and collect samples at 100 points in a file, to identify the file type of fragments. Their text implies that the sliding window is moved one window size at a time, in reality making it a fixed window used to create samples. The standard deviation of the average measurement of each sample point is calculated and used to indicate the level of confidence of each point. They use all parts of a file, including the header, which is deductible from the plots of their results. Hall and Davis also claim their method can be used to link unencrypted files to their resulting encrypted files by their entropy measurements. The plots of the preliminary results show a remarkable similarity between the encrypted files and their source files. According to Davis [48] the reason is that they used Electronic Code Book (ECB) mode, which does not properly hide the underlying structure of the source file [49]. Due to this fact ECB mode is not used in modern encryption applications, hence decreasing the importance of their preliminary results.

The work by Hall and Davis clearly shows how files are often possible to divide into subsections with completely different properties. This is interesting and we will use their ideas in our future work. The main difference between the file type categorisation method and the work by Hall and Davis is the latter method's use of complete files, although divided into 90 byte large windows, while the file type categorisation method is applied to data fragments, although averaged over complete files. Hall and Davis also use more complex metrics to calculate their statistics, by their use of entropy and compressibility. The file type categorisation method is meant to be simple to lower the execution time of the algorithms.

Veenman [50] presents an alternative way to identify the file type of a data fragment. The method uses histograms, similar to the BFD algorithm, entropy measurements, and algorithmic complexity to find the file type of an unknown fragment. Unlike the file type categorisation method Veenman does not use any threshold to do the categorisation. For the applicable file types the results are all worse than for the 2-gram algorithm of the file type categorisation method. To improve the recognition performance Veenman incorporates preceding and consecutive fragments in the calculations. This may be useful in a non-fragmented environment, but for unordered fragment environments the recognition performance may very well decrease. Veenman reports that for text based file types the recognition performance is improved, but for binary file types, such as JPEG and bitmap (BMP), the performance is degenerated, even in a non-fragmented environment.

The main differences between Veenman's work and the file type categorisation method are the metrics used and especially the way Veenman incorporates surrounding fragments in the calculations. The file type categorisation method

is mainly intended to be used for binary, high entropy file types, and is therefore not tested on text based file types. Hence some of the results of Veenman's work are not possible to compare to the results of the file type categorisation method reported in this thesis.

Richard et al [51] have studied what they call in-place file carving, which is a method to reduce the requirements for hard disk space during file carving. They utilise a virtual file system separated from the computer to be analysed, and use it to map interesting blocks of a corrupted hard disk together with a file carver. In this way it is possible to view the prospective files found during carving as if they existed on the hard disk. Since prospective files are not real files they can easily be deleted without actually touching the corrupt hard disk.

The work by Richard et al is related to the work in this thesis because the file type categorisation method can be used for in-line file carving too. By repeatedly using the file type categorisation method with different centroids the contents of a hard disk can be mapped and stored as for example lists of disk clusters belonging to a specific file type.

Scalpel [52] is a file carving tool developed by Richard and colleagues. It shares some code with the Foremost 0.69 tool [53], and is aimed at fast execution. The main idea is to perform two passes through a disk image. The first pass identifies all file headers and stores their positions. The next pass identifies all file footers found within a pre-set distance from the headers and stores the footers' positions. The tool then uses the positions to carve all possible files of different types.

Since the Scalpel tool uses file header and footer information to find files, it cannot perform file carving in a unordered disk image, which the file type categorisation method can. Richard and colleagues plan to extend the header and footer identification capability, but the difference in file categorisation capability from our file type categorisation method will still remain.

Haggerty and Taylor [54] have taken on the problem of matching unknown samples of a file to the original file. Their solution is called *forsigs* and is meant to be faster and harder to defeat than using hash checksums. Forsigs uses one or more signature blocks from the original file and searches a suspect's hard disk for any hits. 16 byte positions from one or more randomly chosen blocks from the original file are used as a signature. If needed the 16 byte positions can also be randomly chosen. The randomness is used to make it harder for a suspect to defeat forsigs by simply changing a few bytes in the file. Forsigs enables specific files to be identified with high certainty without the need for a file system.

Forsigs is used to identify specific and known files, whilst the file type categorisation method is used to identify the file type of fragments. This is definitely a difference between the two methods, but they complement each other. Both are independent of any file system and by including the image fragment reconnection method in the process we get an efficient way of searching for malicious material. We first run forsigs on a hard disk to look for known malicious files. Then we let the file type categorisation method identify all fragments of the same type(s) as the malicious files. Finally we use the image fragment reconnection

method to reconnect the found fragments. Depending on the circumstances we may stop at only checking the reconnected files containing fragments matched by forsigs.

## Chapter 7

# Future Work

This chapter presents areas and questions left as future work. Some of the material has already been discussed in Chapter 5, but this chapter more specifically points out our intended future research path.

The overall direction of our research will move from classical storage media file carving to realtime applications such as network monitoring and live system forensics. The focus will be on parameters that govern the level of success in dire circumstances, where no metadata is available, but applied to different areas where binary data is stored, transferred, or modified. We will continue to work our way through the abstraction layers of a computer system eventually covering all abstraction layers and main application areas for binary data in a computer system.

Practical use of our future work can be found in many computer security areas covering, for example:

- computer forensics,
- data rescue,
- intrusion detection systems,
- rootkit protection,
- anti-virus scanners,
- intrusion analysis, and
- network management.

An interesting area for possible use of our research is efficient network communication in asymmetric channels. This falls outside of the computer security scope of our work, but may offer the possibility of synergy effects where both research fields will benefit. We will conduct a pilot study to further evaluate the applicability of the idea and therefore have left it as future work at the moment.

## 7.1 The File Type Categorisation Method

An outstanding issue that has to be dealt with in the near future is the fact that the results of the experiments using the file type categorisation method for the Windows PE file type can still be improved. Since the ability to detect executable code is a key feature to many computer security services, for example intrusion detection systems or anti-virus scanners, any further improvements of the detection ability of the file type categorisation method would be of high value to the computer security field.

We have already improved the file type categorisation method's detection rate regarding executable code from below 50% to 70% by erasing all GIF, JPEG, and PNG images from the files used to create the centroid. The simplicity of the solution in combination with the drastic improvement of the detection rate has inspired us and we will therefore test an approach where we divide the executable files into smaller parts and sort them into sub-groups. Each group would then get its own centroid, which hopefully will have a high detection rate for its specific part. The sub-division of the centroid creation files may also result in a lower need for cleaning the files before a centroid is created.

We will explore a new algorithm related to the RoC algorithm, where we check the distance between similar byte values. This new algorithm would take regularities in a byte stream into account, such as tables in an executable file, without adding the instability of the 2-gram algorithm. In this way we would get the fast execution and small footprint of the 1-gram algorithms and the higher detection ability of the 2-gram algorithm. The new algorithm is currently being implemented and we hope to be able to report any results in the near future.

The important parameters presented in the Nabs [45] paper are partially similar to the parameters used in the file type categorisation method. The parameters mentioned in the Nabs paper that are currently not used by the file type categorisation method will be evaluated for possible inclusion in the method. We will therefore study the Nabs approach more closely and follow up on its current status.

The Anagram [30] intrusion detector uses interesting techniques to achieve very good results. The key features of the detector will be tested in our file categorisation method. We are especially interested in the use of Bloom filters and the benefits of longer n-grams. We will evaluate the Anagram's ability to handle high entropy file types. We will also investigate the stability of the Anagram approach when subject to data fragments of different sizes.

The quadratic distance metric used in the file type categorisation method will possibly be exchanged. Our idea for a new metric is to calculate the angle between the centroid vectors and the vectors of an unknown sample. The advantages of such an approach are that it is well-founded in mathematics and can be efficiently implemented in certain computer architectures. There are also other similarity metrics that might be interesting to test in the future.

It is possible to decide whether the camera was used to capture a specific image or not [55, 56, 57, 58]. This fact has inspired us to briefly study [11, 12] the possibility of identifying the camera make used to capture an image from a

fragment of the image file. Our preliminary results indicate that it is not possible. In light of our new ideas for algorithms to improve the file type categorisation method, we will re-initiate the camera recognition study. Since there are only a few manufacturers of light sensors on the market, minimally we may be able to distinguish between cameras using different light sensors. Any positive result within this field would extend the capabilities of the police and help them capture child molesters. We will therefore run the camera recognition experiments in parallel with our current file detection research.

Depending on the results from the re-initiated camera recognition experiments, we might also look at identification of image manipulation software. The problem is harder to solve, since computer software is not subject to the same physical limitations as camera light sensors. Hence software can be adjusted and patched to compensate for any positive results from our research.

## 7.2 The Image Fragment Reconnection Method

An important issue we will address in the near future is the fact that the image fragment reconnection method currently only handles JPEG images containing RST markers. By adding the ability to also reconnect fragments lacking RST markers the method will be usable even in live situations, not only in testing environments. We already have a theoretical idea of how to extend the capabilities of the image fragment reconnection method and will implement and test it in the near future. In a more distant future we plan to try to explore the possibility to generalise the image fragment reconnection method, giving the ability to re-assemble any type of file. This is partly dependent on the progress of the file type categorisation algorithm development, hence the timing of the generalisation of the image fragment reconnection method is not yet clear.

The near future work on the image fragment reconnection method includes continued experiments using fragments from several images, to confirm the results we present in this thesis. The experiments should also include a complete reassembly of a file to find the number of iterations needed to correctly reconnect all fragments in our test set. The results from these experiments can be used to improve the image fragment reconnection method and help in finding any yet unknown weaknesses in the algorithms. This work is planned to be done in parallel with the implementation of the fully JPEG capable image fragment reconnection method.

## 7.3 Artificial JPEG Header

Our research on the use of artificial JPEG headers has been performed using an image containing RST markers. This has biased the results in a positive direction, possibly leading to a higher level of success than if we had used an image without RST markers. All example images containing small discoloured horizontal bars are subject to the bias. The bars end at the border of a restart interval; if there had been no RST markers the first bar to appear would have continued to the end

of the image. We therefore have to re-evaluate the effects of changes to the sample rates, the component identifiers, and Huffman table definitions and pointers in a non-RST marker JPEG image.

An open issue to address regarding the use of artificial JPEG headers is whether it is possible to use a fully generic Huffman table definition. Our current research indicates that almost identical Huffman table definitions are exchangeable, but what happens when the definitions differ to a higher degree? Will the image be possible to decode at all?

All cameras in our database use the same Huffman table definitions, but we need to know whether the same is true for any digital camera. If there are only a few different definitions used, a brute force model where all possible Huffman table definitions are tried on an image would be feasible. This would possibly also lead to the identification of the camera make used to capture the image.

The principle of Huffman coding is that frequent symbols get shorter codes than less frequent symbols, and shorter codes mean fewer possible combinations. Thus the set of short Huffman codes is small and should theoretically be frequently used, hence are there any theoretical or practical structures governing how the symbols get coded in a JPEG Huffman table definition? If any structures exist, will they cause similarities between two independent Huffman table definitions and if that is the case, in what parts of the definitions?



# Bibliography

- [1] R. Wortley and S. Smallbone, "Child pornography on the internet," in *Problem-Oriented Guides for Police*, ser. Problem-Specific Guides Series. US Department of Justice, May 2006, no. 41, accessed 2008-03-15.
- [2] J. Ruscitti, "Child porn fight merits police resources," <http://www.crime-research.org/news/16.02.2008/3201/>, Feb. 2008, accessed 2008-03-14.
- [3] A. McCue, "Online child porn investigation costs police £15m," <http://management.silicon.com/government/0,39024677,39128445,00.htm>, Mar. 2005, accessed 2008-03-14.
- [4] BBC News, "Child porn fight 'lacks funding'," <http://news.bbc.co.uk/go/pr/fr/-/1/hi/technology/3972763.stm>, Nov. 2004, accessed 2008-03-14.
- [5] —, "BT acts against child porn sites," <http://news.bbc.co.uk/1/hi/technology/3786527.stm>, June 2004, accessed 2008-03-17.
- [6] Yahoo News Canada, "Police arrest 22 in Ontario as part of massive child porn bust," [http://ca.news.yahoo.com/s/capress/080212/national/child\\_porn\\_raids\\_7](http://ca.news.yahoo.com/s/capress/080212/national/child_porn_raids_7), Feb. 2008, accessed 2008-04-01.
- [7] "DFRWS 2007 forensics challenge results," <http://www.dfrws.org/2007/challenge/results.shtml>, 2007, accessed 2008-03-15.
- [8] B. Carrier, *File System Forensic Analysis*. Addison-Wesley, 2005.
- [9] M. Karresand and N. Shahmehri, "Oscar – file type identification of binary data in disk clusters and RAM pages," in *Proceedings of IFIP International Information Security Conference: Security and Privacy in Dynamic Environments (SEC2006)*, ser. Lecture Notes in Computer Science, 2006, pp. 413–424.
- [10] —, "File type identification of data fragments by their binary structure," in *Proceedings from the Seventh Annual IEEE Systems, Man and Cybernetics (SMC) Information Assurance Workshop, 2006*. Piscataway, NJ, USA: IEEE, 2006, pp. 140–147.

- [11] —, “Oscar – file type and camera identification using the structure of binary data fragments,” in *Proceedings of the 1st Conference on Advances in Computer Security and Forensics, ACSF*, J. Haggerty and M. Merabti, Eds. Liverpool, UK: The School of Computing and Mathematical Sciences, John Moores University, July 2006, pp. 11–20.
- [12] —, “Oscar – using byte pairs to find file type and camera make of data fragments,” in *Proceedings of the 2nd European Conference on Computer Network Defence, in conjunction with the First Workshop on Digital Forensics and Incident Analysis (EC2ND 2006)*, A. Blyth and I. Sutherland, Eds. Springer Verlag, 2007, pp. 85–94.
- [13] The International Telegraph and Telephone Consultative Committee (CCITT), “Recommendation T.81 — Information Technology – Digital Compression and Coding of Continuous-Tone Still Images – Requirements and Guidelines,” Sept. 1992, also published as ISO/IEC International Standard 10918-1.
- [14] E. Hamilton, “JPEG File Interchange Format – version 1.02,” Sept. 1992.
- [15] International Telecommunication Union (ITU-T), “Annex F of ITU-T Recommendation T.84 | ISO/IEC IS 10918-3, Digital Compression and Coding of Continuous-Tone Still Images – Extensions,” 1996, ITU-T was previously known as CCITT.
- [16] Japan Electronics and Information Technology Industries Association (JEITA), “Exchangeable image file format for digital still cameras: Exif Version 2.2,” Apr. 2002, JEITA CP-3451.
- [17] Japan Electronic Industry Development Association (JEIDA), “Digital Still Camera Image File Format Standard (Exchangeable image file format for Digital Still Cameras: Exif – Version 2.1),” June 1998, earlier versions published 1996 (v. 1.0), and 1997 (v. 1.1).
- [18] M. Damashek, “Gauging similarity with n-grams: Language-independent categorization of text,” *Science*, vol. 267, no. 5199, pp. 843–848, Feb. 1995.
- [19] P. Deutsch, “RFC1951: DEFLATE compressed data format specification version 1.3,” <http://www.ietf.org/rfc/rfc1951.txt>, May 1996, last accessed 2007-11-20.
- [20] J. Callas, L. Donnerhake, H. Finney, D. Shaw, and R. Thayer, “RFC 4880: OpenPGP message format,” <http://tools.ietf.org/rfc/rfc4880.txt>, Nov. 2007, last accessed 2007-11-26.
- [21] D. O’Neill, “ID3v2 — the audience is informed,” <http://www.id3.org/Home>, July 2007, last accessed 2007-11-27.
- [22] M. Nilsson, D. Mahoney, and J. Sundström, “ID3 tag version 2.3.0,” <http://www.id3.org/id3v2.3.0>, Feb. 1999, last accessed 2007-11-27.

- [23] The Gutenberg Project, “Project Gutenberg,” [http://www.gutenberg.org/wiki/Main\\_Page](http://www.gutenberg.org/wiki/Main_Page), July 2007, last accessed 2007-11-26.
- [24] “Debian Etch r2 XFce CD iso file from the SUNET FTP site,” <ftp://ftp.sunet.se/pub/Linux/distributions/debian-cd/current/i386/iso-cd/debian-40r2-i386-xfce-CD-1.iso>, Feb. 2008, accessed 2008-02-05.
- [25] T. Fawcett, “An introduction to ROC analysis,” *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, June 2006.
- [26] L. Råde and B. Westergren, *Mathematics Handbook for Science and Engineering*, 4th ed. Studentlitteratur, 1998.
- [27] FelixH and Stephantom, “Image:Dctjpeg.png,” <http://upload.wikimedia.org/wikipedia/commons/2/23/Dctjpeg.png>, Jan. 2008, accessed 2008-03-29.
- [28] K. Wang and S. Stolfo, “Anomalous payload-based network intrusion detection,” in *Recent Advances in Intrusion Detection (RAID) 2004*, ser. Lecture Notes in Computer Science, E. Jonsson et al., Eds., vol. 3224. Springer-Verlag, July 2004, pp. 203–222.
- [29] K. Wang, G. Cretu, and S. Stolfo, “Anomalous payload-based worm detection and signature generation,” in *8th International Symposium on Recent Advances in Intrusion Detection, RAID 2005*, ser. Lecture Notes in Computer Science, A. Valdes and D. Zamboni, Eds., vol. 3858. Springer-Verlag, 2006, pp. 227–246.
- [30] K. Wang, J. Parekh, and S. Stolfo, “Anagram: A content anomaly detector resistant to mimicry attacks,” in *9th International Symposium Recent Advances in Intrusion Detection (RAID)*, ser. Lecture Notes in Computer Science, D. Zamboni and C. Kruegel, Eds., vol. 4219. Springer Berlin / Heidelberg, 2006, pp. 226–248, DOI: 10.1007/11856214\_12.
- [31] W.-J. Li, S. Stolfo, A. Stavrou, E. Androulaki, and A. Keromytis, “A study of malware-bearing documents,” in *Proceedings of the 4th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA 2007)*, ser. Lecture Notes in Computer Science (LNCS), B. Hämmerli and R. Sommer, Eds., vol. 4579. Springer Verlag, 2007, pp. 231–250.
- [32] W.-J. Li and S. Stolfo, “SPARSE: a hybrid system to detect malware-bearing documents,” <http://www1.cs.columbia.edu/ids/publications/cucs-006-08.pdf>, Dept. of Computer Science, Columbia University, USA, Tech. Rep. cuacs-006-08, Jan. 2008, accessed 2008-04-01.
- [33] W.-J. Li, K. Wang, S. Stolfo, and B. Herzog, “Fileprints: Identifying file types by n-gram analysis,” in *Proceedings from the sixth IEEE Systems, Man and Cybernetics Information Assurance Workshop*, June 2005, pp. 64–71.

- [34] S. Stolfo, K. Wang, and W.-J. Li, "Fileprint analysis for malware detection," Computer Science Department, Columbia University, New York, NY, USA, Tech. Rep., 2005, review draft.
- [35] —, *Malware Detection*, ser. Advances in Information Security. Springer US, 2007, vol. 27, no. IV, ch. Towards Stealthy Malware Detection, pp. 231–249.
- [36] M. McDaniel and H. Heydari, "Content based file type detection algorithms," in *HICSS '03: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 9*. Washington, DC, USA: IEEE Computer Society, 2003, p. 332.1.
- [37] K. Shanmugasundaram and N. Memon, "Automatic reassembly of document fragments via context based statistical models," in *Proceedings of the 19th Annual Computer Security Applications Conference*, 2003, <http://www.acsac.org/2003/papers/97.pdf>, accessed at 2006-04-30.
- [38] A. Shamir and N. van Someren, "Playing 'hide and seek' with stored keys," in *Financial Cryptography: Third International Conference, FC'99*, ser. Lecture Notes in Computer Science, M. Franklin, Ed., vol. 1648. Springer-Verlag, 1999, pp. 118–124.
- [39] S. Garfinkel, "Carving contiguous and fragmented files with fast object validation," *Digital Investigation*, vol. 4, no. Supplement 1, pp. 2–12, Sept. 2007, DOI:<http://dx.doi.org/10.1016/j.diin.2007.06.017>.
- [40] M. Cohen, "Advanced carving techniques," *Digital Investigation*, 2007, in press, DOI:<http://dx.doi.org/10.1016/j.diin.2007.10.001>.
- [41] R. Erbacher and J. Mulholland, "Identification and localization of data types within large-scale file systems," in *SADFE '07: Proceedings of the Second International Workshop on Systematic Approaches to Digital Forensic Engineering*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 55–70, DOI:<http://dx.doi.org/10.1109/SADFE.2007.12>.
- [42] K. Shanmugasundaram and N. Memon, "Automatic reassembly of document fragments via data compression," in *Digital Forensic Research Workshop*, 2002, [http://isis.poly.edu/memon/publications/pdf/2002\\_Automatic\\_Reassembly\\_of\\_Document\\_Fragments\\_via\\_Data\\_Compression.pdf](http://isis.poly.edu/memon/publications/pdf/2002_Automatic_Reassembly_of_Document_Fragments_via_Data_Compression.pdf), last accessed 2008-01-14.
- [43] A. Pal, K. Shanmugasundaram, and N. Memon, "Automated reassembly of fragmented images," in *ICME '03: Proceedings of the 2003 International Conference on Multimedia and Expo*. Washington, DC, USA: IEEE Computer Society, 2003, pp. 625–628.
- [44] N. Memon and A. Pal, "Automated reassembly of file fragmented images using greedy algorithms," *IEEE Transactions on Image Processing*, vol. 15, no. 2, pp. 385–393, Feb. 2006.

- [45] K. Shanmugasundaram, M. Kharrazi, and N. Memon, "Nabs: A system for detecting resource abuses via characterization of flow content type," in *Proceedings of 20th Annual Computer Security Applications Conference*, 2004.
- [46] M. Shannon, "Forensic relative strength scoring: ASCII and entropy scoring," *International Journal of Digital Evidence*, vol. 2, no. 4, 2004, <http://www.utica.edu/academic/institutes/ecii/publications/articles/A0B3DC9E-F145-4A89-36F7462B629759FE.pdf>, last accessed 2007-12-21.
- [47] G. Hall and W. Davis, "Sliding window measurement for file type identification," <http://www.mantechcfia.com/SlidingWindowMeasurementforFileTypeIdentification.pdf>, 2007, last accessed 2007-12-10.
- [48] W. Davis, "Re: Fw: Questions on "sliding window measurement for file type identification" paper," Mail conversation, Jan. 2008.
- [49] A. Menezes, P. Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, 1997.
- [50] C. Veenman, "Statistical disk cluster classification for file carving," in *Proceedings of the Third International Symposium on Information Assurance and Security, 2007 (IAS 2007)*, N. Zhang, A. Abraham, Q. Shi, and J. Thomas, Eds. IEEE Computer Society, 2007, pp. 393–398, DOI:<http://dx.doi.org/10.1109/ISIAS.2007.4299805>.
- [51] G. Richard III, V. Roussev, and L. Marziale, "In-place file carving," in *Advances in Digital Forensics III*, ser. IFIP International Federation for Information Processing, P. Craiger and S. Sheno, Eds., vol. 242/2007. Springer Boston, 2007, pp. 217–230.
- [52] G. Richard III and V. Roussev, "Scalpel: A frugal, high performance file carver," in *Proceedings of the 5th Annual Digital Forensics Research Workshop (DFRWS 2005)*, 2005.
- [53] K. Kendall, J. Kornblum, and N. Mikus, "Foremost - latest version 1.5.3," <http://foremost.sourceforge.net/>, 2007, accessed 2008-04-14.
- [54] J. Haggerty and M. Taylor, "FORSIGS: forensic signature analysis of the hard drive for multimedia file fingerprints," in *New Approaches for Security, Privacy and Trust in Complex Environments*, ser. IFIP International Federation for Information Processing, vol. 232. Springer Boston, 2007, pp. 1–12, DOI: 10.1007/978-0-387-72367-9\_1.
- [55] J. Lukáš, J. Fridrich, and M. Goljan, "Determining digital image origin using sensor imperfections," in *Proceedings of SPIE Electronic Imaging, Image and Video Communication and Processing*, 2005, pp. 249–260.
- [56] —, "Digital bullet scratches for images," in *Proceedings of ICIP 2005*, 2005.

## BIBLIOGRAPHY

- [57] G. Glover, “Binghamton University research links digital images and cameras,” [http://www.eurekalert.org/pub\\_releases/2006-04/bu-bur041806.php](http://www.eurekalert.org/pub_releases/2006-04/bu-bur041806.php), Apr. 2006, accessed 2006-04-28.
- [58] J. Lukáš, J. Fridrich, and M. Goljan, “Digital camera identification from sensor pattern noise,” *IEEE Transactions on Information Forensics and Security*, vol. 1, no. Summer 2, pp. 205–214, 2006.

# Appendix A

## Acronyms

<b>ADIT</b>	Division for Database and Information Techniques
<b>AES</b>	Advanced Encryption Standard
<b>ASCII</b>	American Standard Code for Information Interchange
<b>BFD</b>	Byte Frequency Distribution
<b>BGC</b>	Bifragment Gap Carving
<b>BMP</b>	bitmap
<b>CCITT</b>	International Telegraph and Telephone Consultative Committee
<b>DCT</b>	Discrete Cosine Transform
<b>DFRWS</b>	Digital Forensic Research Workshop
<b>DHT</b>	Define Huffman Table
<b>DRI</b>	Define Restart Interval
<b>ECB</b>	Electronic Code Book
<b>EOI</b>	End Of Image
<b>FAT</b>	File Allocation Table
<b>FOI</b>	Swedish Defence Research Agency
<b>GIF</b>	Graphics Interchange Format
<b>GPG</b>	GNU Privacy Guard
<b>HTML</b>	HyperText Markup Language
<b>ID3</b>	IDentify an MP3

## APPENDIX A. ACRONYMS

<b>IEC</b>	International Electrotechnical Commission
<b>IISLAB</b>	Laboratory for Intelligent Information Systems
<b>ISO</b>	International Organization for Standardization
<b>JPEG</b>	Joint Photographic Experts Group
<b>MCU</b>	Minimum Coded Unit
<b>MP3</b>	MPEG 1, Audio Layer-3
<b>NTFS</b>	New Technologies File System
<b>PAYL</b>	PAYLoad-based anomaly detector for intrusion detection
<b>PDF</b>	Portable Document Format
<b>PE</b>	Portable Executable
<b>PNG</b>	Portable Network Graphics
<b>PXE</b>	Preboot Execution Environment
<b>RAM</b>	Random Access Memory
<b>RGB</b>	Red Green Blue
<b>RIP</b>	Recovery Is Possible
<b>RKP</b>	National Criminal Investigation Department
<b>RoC</b>	Rate of Change
<b>ROC</b>	Receiver Operating Characteristic
<b>RST</b>	Restart
<b>SFFS</b>	Sequential Forward Feature Selection
<b>SKL</b>	National Laboratory of Forensic Science
<b>SOF</b>	Start Of Frame
<b>SOS</b>	Start Of Scan
<b>SPARSE</b>	Statistical PARSEr
<b>TCP</b>	Transmission Control Protocol



## Appendix B

# Hard Disk Allocation Strategies

When an operating system allocates hard disk data units to the file system, several different strategies can be used. A simple strategy is to start from the beginning of the hard disk and allocate the first available data unit. This strategy is logically called *first available*. The result is often that files are fragmented and that data units close to the end of the hard disk are more seldomly allocated. The Linux second and third extended (ext2/ext3) file systems use a first available allocation strategy. [8, p. 179]

Another allocation strategy similar to the first available strategy is the *next available* strategy. When this is used the operating system starts looking for free data units from the position of the last allocated data unit. In this way the hard disk surface is more evenly utilised, but the risk of fragmentation is still high. The possibility to recover deleted files is increased relative to the first available strategy, since the new allocation point is moved in a round-robin fashion. The File Allocation Table (FAT) file system typically uses this allocation strategy. [8, p. 180].

A third allocation strategy is the *best fit* strategy, where the operating system searches for a set of consecutive unallocated data units large enough to hold the new file in its entirety. This strategy cannot handle all situations; if there is not any large enough available unallocated area the operating system will have to resort to a first available or next available strategy. If a file grows in size the best fit allocation strategy might still lead to fragmentation, if an application opens the original file and updates it. If the application instead makes a copy of the file, updates it and writes it to a new location on disk the risk of fragmentation is decreased. The behaviour is application specific, so any file system can become fragmented. An example of a file system using the best fit allocation strategy is the New Technologies File System (NTFS) file system. [8, p. 180]

The actual allocation strategy used by an operating system may vary, but the most probable strategy has been empirically examined by Carrier [8]. The results can be seen in Table B.1.

The Linux file systems ext2 and ext3 divide the hard disk into a number of

**Table B.1:** Data unit allocation strategies for different operating systems and file systems, as reported by Carrier [8].

Operating System	File System	Strategy
MS Windows 9x	FAT	Next available
MS Windows XP	FAT	Next available
MS Windows XP	NTFS	Best fit
Linux	ext2/3	First and next available

groups containing blocks of hard disk sectors. The strategy is to keep the seek time to a minimum by allocating metadata and data units of a file to the same group, if possible. Therefore the file systems will use a first available strategy when allocating data units to a new file, and a next available strategy when a file is enlarged. [8, p. 410]

Regardless of which of the three allocation strategies an operating system uses, the data units will be allocated in a sequential order, although possibly in a round-robin fashion. If the allocation address of the first data unit of a file is not known, it is possible to perform a brute force search to locate it. This will simplify an attempt to reconnect the fragments (data units) of a file found on a hard disk. By locating every fragment of a certain file type it is possible to reconnect them into the original files simply by using their relative positions and a method to calculate the connection validity of each fragment. The final step is then to test each of them as the starting fragment. Since we know the ordering of the fragments we only have to separate fragments belonging to different files to be able to reconnect them.

## Appendix C

### Confusion Matrices

This appendix contains the confusion matrices of the different algorithms used in the testing. The rows in the tables represent the centroids and the columns represent the testing files. This should be seen as the centroid representing the correct categorisation, the actual class, and the testing files representing the predicted classes.

The short form of the centroid names used in the confusion matrices can be found in Table C.1. The matrices can be found in Tables C.2–C.16.

To increase the readability the table for the 2-gram algorithm is repeated in the appendix. It can also be found in Section 2.6.6, together with a short analysis of the results shown in the table.

**Table C.1:** The following short names are used for the confusion matrices.

Short name	Centroid
exe	Windows PE files
cast5	CAST5 encrypted GPG file
aes	AES encrypted GPG file
no rst	JPEG images without RST markers
rst	JPEG images with RST markers
mp3	MP3 audio files without an ID3 tag
zip	Files compressed using Zip

**Table C.2:** The confusion matrix of the 2-gram algorithm.

	exe	aes	cast5	no rst	rst	mp3	zip
exe	15332	8	10	12	15	7247	56
aes	100	402	16450	3	0	0	5725
cast5	107	16614	339	0	0	0	5620
no rst	10	0	0	22667	0	0	3
rst	0	0	0	9677	13003	0	0
mp3	60	371	276	3	6	21882	82
zip	113	5699	5650	4	0	0	11214

**Table C.3:** The confusion matrix of the Byte Frequency Distribution algorithm having the JPEG rules set extension.

	exe	aes	cast5	no rst	rst	mp3	zip
no rst	96	4332	4425	2716	8260	1	2850
rst	33	956	1034	1884	18454	18	301

**Table C.4:** The confusion matrix of the combination of the Byte Frequency Distribution algorithm with the JPEG rule set extension and the Rate of Change (RoC) algorithm.

	exe	aes	cast5	no rst	rst	mp3	zip
no rst	145	5155	5253	2100	7123	0	2904
rst	49	1480	1518	1487	17713	1	432

**Table C.5:** The confusion matrix of the Byte Frequency Distribution algorithm combined with the Rate of Change algorithm using a Manhattan distance metric.

	exe	aes	cast5	no rst	rst	mp3	zip
exe	5480	209	181	26	0	16635	149
aes	52	10213	10116	1	0	0	2298
cast5	51	10261	10077	1	0	0	2290
no rst	86	4576	4660	3023	7513	2	2820
rst	26	890	878	1715	18850	14	307
mp3	18	832	858	1	3	20659	309
zip	87	8987	8963	0	1	0	4642

**Table C.6:** The confusion matrix of the Byte Frequency Distribution algorithm combined with the Rate of Change algorithm.

	exe	aes	cast5	no rst	rst	mp3	zip
exe	3547	252	251	20	0	18380	230
aes	64	10027	10003	0	0	0	2586
cast5	65	10057	9966	0	0	0	2592
no rst	87	4752	4782	3180	7004	1	2874
rst	31	941	933	1688	18727	4	356
mp3	16	785	804	0	2	20740	333
zip	101	8727	8750	0	0	0	5102

**Table C.7:** The confusion matrix of the Byte Frequency Distribution algorithm.

	exe	aes	cast5	no rst	rst	mp3	zip
exe	5717	312	299	38	0	16147	167
aes	35	10211	10325	0	1	0	2108
cast5	36	10204	10321	0	1	0	2118
no rst	96	4332	4425	2716	8260	1	2850
rst	33	956	1034	1884	18454	18	301
mp3	19	1762	1785	0	0	18704	410
zip	76	8781	8749	0	2	0	5072

**Table C.8:** The confusion matrix of the combination of the Byte Frequency Distribution algorithm having a JPEG rule set and the Rate of Change algorithm using signed values.

	exe	aes	cast5	no rst	rst	mp3	zip
no rst	63	4430	4455	3377	7590	1	2764
rst	19	809	817	1673	19078	5	279

**Table C.9:** The confusion matrix of the Byte Frequency Distribution algorithm with a JPEG rule set and the Rate of Change algorithm using signed values and a Manhattan distance metric.

	exe	aes	cast5	no rst	rst	mp3	zip
exe	4891	214	191	22	0	17206	156
aes	40	10172	10175	1	0	0	2292
cast5	39	10183	10152	1	0	0	2305
no rst	79	4342	4387	3177	7925	0	2770
rst	18	816	851	1730	18999	8	258
mp3	20	976	1026	1	0	20336	321
zip	70	8949	8846	0	1	0	4814

## APPENDIX C. CONFUSION MATRICES

**Table C.10:** The confusion matrix of the Byte Frequency Distribution algorithm combined with the Rate of Change algorithm using signed values.

	exe	aes	cast5	no rst	rst	mp3	zip
exe	2671	310	274	23	1	19167	234
aes	52	10080	10104	0	0	0	2444
cast5	51	10112	10076	0	0	0	2441
no rst	63	4430	4455	3377	7590	1	2764
rst	19	809	817	1673	19078	5	279
mp3	15	861	914	0	1	20558	331
zip	77	8749	8579	0	0	0	5275

**Table C.11:** The confusion matrix of the Rate of Change algorithm using a JPEG rule set and signed values.

	exe	aes	cast5	no rst	rst	mp3	zip
no rst	120	4609	4513	6355	3694	87	3302
rst	55	1915	1906	3398	13962	33	1411

**Table C.12:** The confusion matrix of the Rate of Change algorithm with Manhattan distance metric and signed values.

	exe	aes	cast5	no rst	rst	mp3	zip
exe	3473	678	635	1338	261	14979	1316
aes	182	7189	7156	2242	1056	147	4708
cast5	182	7323	7180	2148	1011	119	4717
no rst	133	4605	4580	6280	3659	151	3272
rst	68	2164	2175	3724	13016	55	1478
mp3	65	799	771	554	234	19240	1017
zip	179	7055	6999	2461	1083	190	4713

**Table C.13:** The confusion matrix of the Rate of Change algorithm using signed values.

	exe	aes	cast5	no rst	rst	mp3	zip
exe	1002	1054	979	2232	314	15242	1857
aes	200	8411	8270	133	99	66	5501
cast5	198	8427	8274	120	94	66	5501
no rst	120	4609	4513	6355	3694	87	3302
rst	55	1915	1906	3398	13962	33	1411
mp3	67	1071	1024	53	47	19178	1240
zip	205	8089	7958	120	100	153	6055

**Table C.14:** The confusion matrix of the Rate of Change algorithm using a JPEG rule set.

	exe	aes	cast5	no rst	rst	mp3	zip
no rst	150	5056	4924	5125	3763	99	3563
rst	84	2727	2716	3370	11777	52	1954

**Table C.15:** The confusion matrix of the Rate of Change algorithm using a Manhattan distance metric.


	exe	aes	cast5	no rst	rst	mp3	zip
exe	5268	431	421	1651	192	13689	1028
aes	211	7134	6932	2470	1318	97	4518
cast5	209	7119	6952	2430	1358	87	4525
no rst	164	5086	5004	5230	3521	161	3514
rst	91	2869	2838	3486	11327	100	1969
mp3	60	561	608	635	431	19469	916
zip	209	7040	6748	2674	1337	135	4537

**Table C.16:** The confusion matrix of the Rate of Change algorithm.

	exe	aes	cast5	no rst	rst	mp3	zip
exe	2213	570	542	1491	107	16534	1223
aes	205	8366	8226	211	245	67	5360
cast5	204	8382	8210	206	241	66	5371
no rst	150	5056	4924	5125	3763	99	3563
rst	84	2727	2716	3370	11777	52	1954
mp3	64	937	935	173	277	18942	1352
zip	229	8055	7911	257	298	137	5793





 <p><b>Linköpings universitet</b></p>		<p><b>Avdelning, Institution</b> Division, Department</p> <p>Institutionen för datavetenskap Department of Computer and Information Science</p>	<p><b>Datum</b> Date</p> <p>2008-05-20</p>
<p><b>Språk</b> Language</p> <p><input type="checkbox"/> Svenska/Swedish <input checked="" type="checkbox"/> Engelska/English</p> <p><input type="checkbox"/> _____</p>	<p><b>Rapporttyp</b> Report category</p> <p><input checked="" type="checkbox"/> Licentiatavhandling <input type="checkbox"/> Examensarbete <input type="checkbox"/> C-uppsats <input type="checkbox"/> D-uppsats <input type="checkbox"/> Övrig rapport <input type="checkbox"/> _____</p>	<p><b>ISBN</b> 978-91-7393-915-7</p> <hr/> <p><b>ISRN</b> LiU-Tek-Lic-2008:19</p> <hr/> <p><b>Serietitel och serienummer</b> Title of series, numbering</p> <p><b>ISSN</b> 0280-7971</p> <hr/> <p>Linköping Studies in Science and Technology Thesis No. 1361</p>	
<p><b>URL för elektronisk version</b></p>			
<p><b>Titel</b> Title</p> <p>Completing the Picture — Fragments and Back Again</p> <p><b>Författare</b> Author</p> <p>Martin Karresand</p>			
<p><b>Sammanfattning</b> Abstract</p> <p>Better methods and tools are needed in the fight against child pornography. This thesis presents a method for file type categorisation of unknown data fragments, a method for reassembly of JPEG fragments, and the requirements put on an artificial JPEG header for viewing reassembled images. To enable empirical evaluation of the methods a number of tools based on the methods have been implemented.</p> <p>The file type categorisation method identifies JPEG fragments with a detection rate of 100% and a false positives rate of 0.1%. The method uses three algorithms, Byte Frequency Distribution (BFD), Rate of Change (RoC), and 2-grams. The algorithms are designed for different situations, depending on the requirements at hand.</p> <p>The reconnection method correctly reconnects 97% of a Restart (RST) marker enabled JPEG image, fragmented into 4 KiB large pieces. When dealing with fragments from several images at once, the method is able to correctly connect 70% of the fragments at the first iteration.</p> <p>Two parameters in a JPEG header are crucial to the quality of the image; the size of the image and the sampling factor (actually factors) of the image. The size can be found using brute force and the sampling factors only take on three different values. Hence it is possible to use an artificial JPEG header to view full of parts of an image. The only requirement is that the fragments contain RST markers.</p> <p>The results of the evaluations of the methods show that it is possible to find, reassemble, and view JPEG image fragments with high certainty.</p>			
<p><b>Nyckelord</b> Keywords</p> <p>computer security, computer forensics, file carving</p>			



**Linköping Studies in Science and Technology**  
**Faculty of Arts and Sciences - Licentiate Theses**

- No 17 **Vojin Plavsic:** Interleaved Processing of Non-Numerical Data Stored on a Cyclic Memory. (Available at: FOA, Box 1165, S-581 11 Linköping, Sweden. FOA Report B30062E)
- No 28 **Arne Jönsson, Mikael Patel:** An Interactive Flowcharting Technique for Communicating and Realizing Algorithms, 1984.
- No 29 **Johnny Eckerland:** Retargeting of an Incremental Code Generator, 1984.
- No 48 **Henrik Nordin:** On the Use of Typical Cases for Knowledge-Based Consultation and Teaching, 1985.
- No 52 **Zebo Peng:** Steps Towards the Formalization of Designing VLSI Systems, 1985.
- No 60 **Johan Fagerström:** Simulation and Evaluation of Architecture based on Asynchronous Processes, 1985.
- No 71 **Jalal Maleki:** ICONStraint, A Dependency Directed Constraint Maintenance System, 1987.
- No 72 **Tony Larsson:** On the Specification and Verification of VLSI Systems, 1986.
- No 73 **Ola Strömfors:** A Structure Editor for Documents and Programs, 1986.
- No 74 **Christos Levcopoulos:** New Results about the Approximation Behavior of the Greedy Triangulation, 1986.
- No 104 **Shamsul I. Chowdhury:** Statistical Expert Systems - a Special Application Area for Knowledge-Based Computer Methodology, 1987.
- No 108 **Rober Bilos:** Incremental Scanning and Token-Based Editing, 1987.
- No 111 **Hans Block:** SPORT-SORT Sorting Algorithms and Sport Tournaments, 1987.
- No 113 **Ralph Rönquist:** Network and Lattice Based Approaches to the Representation of Knowledge, 1987.
- No 118 **Mariam Kamkar, Nahid Shahmehri:** Affect-Chaining in Program Flow Analysis Applied to Queries of Programs, 1987.
- No 126 **Dan Strömberg:** Transfer and Distribution of Application Programs, 1987.
- No 127 **Kristian Sandahl:** Case Studies in Knowledge Acquisition, Migration and User Acceptance of Expert Systems, 1987.
- No 139 **Christer Bäckström:** Reasoning about Interdependent Actions, 1988.
- No 140 **Mats Wirén:** On Control Strategies and Incrementality in Unification-Based Chart Parsing, 1988.
- No 146 **Johan Hultman:** A Software System for Defining and Controlling Actions in a Mechanical System, 1988.
- No 150 **Tim Hansen:** Diagnosing Faults using Knowledge about Malfunctioning Behavior, 1988.
- No 165 **Jonas Löwgren:** Supporting Design and Management of Expert System User Interfaces, 1989.
- No 166 **Ola Petersson:** On Adaptive Sorting in Sequential and Parallel Models, 1989.
- No 174 **Ngve Larsson:** Dynamic Configuration in a Distributed Environment, 1989.
- No 177 **Peter Åberg:** Design of a Multiple View Presentation and Interaction Manager, 1989.
- No 181 **Henrik Eriksson:** A Study in Domain-Oriented Tool Support for Knowledge Acquisition, 1989.
- No 184 **Ivan Rankin:** The Deep Generation of Text in Expert Critiquing Systems, 1989.
- No 187 **Simin Nadjim-Tehrani:** Contributions to the Declarative Approach to Debugging Prolog Programs, 1989.
- No 189 **Magnus Merkel:** Temporal Information in Natural Language, 1989.
- No 196 **Ulf Nilsson:** A Systematic Approach to Abstract Interpretation of Logic Programs, 1989.
- No 197 **Staffan Bonnier:** Horn Clause Logic with External Procedures: Towards a Theoretical Framework, 1989.
- No 203 **Christer Hansson:** A Prototype System for Logical Reasoning about Time and Action, 1990.
- No 212 **Björn Fjellborg:** An Approach to Extraction of Pipeline Structures for VLSI High-Level Synthesis, 1990.
- No 230 **Patrick Doherty:** A Three-Valued Approach to Non-Monotonic Reasoning, 1990.
- No 237 **Tomas Sokolnicki:** Coaching Partial Plans: An Approach to Knowledge-Based Tutoring, 1990.
- No 250 **Lars Strömberg:** Postmortem Debugging of Distributed Systems, 1990.
- No 253 **Torbjörn Näslund:** SLDFA-Resolution - Computing Answers for Negative Queries, 1990.
- No 260 **Peter D. Holmes:** Using Connectivity Graphs to Support Map-Related Reasoning, 1991.
- No 283 **Olof Johansson:** Improving Implementation of Graphical User Interfaces for Object-Oriented Knowledge-Bases, 1991.
- No 298 **Rolf G Larsson:** Aktivitetsbaserad kalkylering i ett nytt ekonomisystem, 1991.
- No 318 **Lena Srömbäck:** Studies in Extended Unification-Based Formalism for Linguistic Description: An Algorithm for Feature Structures with Disjunction and a Proposal for Flexible Systems, 1992.
- No 319 **Mikael Pettersson:** DML-A Language and System for the Generation of Efficient Compilers from Denotational Specification, 1992.
- No 326 **Andreas Kågedal:** Logic Programming with External Procedures: an Implementation, 1992.
- No 328 **Patrick Lambrix:** Aspects of Version Management of Composite Objects, 1992.
- No 333 **Xinli Gu:** Testability Analysis and Improvement in High-Level Synthesis Systems, 1992.
- No 335 **Torbjörn Näslund:** On the Role of Evaluations in Iterative Development of Managerial Support Systems, 1992.
- No 348 **Ulf Cederling:** Industrial Software Development - a Case Study, 1992.
- No 352 **Magnus Morin:** Predictable Cyclic Computations in Autonomous Systems: A Computational Model and Implementation, 1992.
- No 371 **Mehran Noghabai:** Evaluation of Strategic Investments in Information Technology, 1993.
- No 378 **Mats Larsson:** A Transformational Approach to Formal Digital System Design, 1993.
- No 380 **Johan Ringström:** Compiler Generation for Parallel Languages from Denotational Specifications, 1993.
- No 381 **Michael Jansson:** Propagation of Change in an Intelligent Information System, 1993.
- No 383 **Jonni Harrius:** An Architecture and a Knowledge Representation Model for Expert Critiquing Systems, 1993.
- No 386 **Per Österling:** Symbolic Modelling of the Dynamic Environments of Autonomous Agents, 1993.
- No 398 **Johan Boye:** Dependency-based Groudness Analysis of Functional Logic Programs, 1993.

- No 402 **Lars Degerstedt:** Tabulated Resolution for Well Founded Semantics, 1993.
- No 406 **Anna Moberg:** Satellitkontor - en studie av kommunikationsmönster vid arbete på distans, 1993.
- No 414 **Peter Carlsson:** Separation av företagsledning och finansiering - fallstudier av företagsledarutköp ur ett agent-teoretiskt perspektiv, 1994.
- No 417 **Camilla Sjöström:** Revision och lagreglering - ett historiskt perspektiv, 1994.
- No 436 **Cecilia Sjöberg:** Voices in Design: Argumentation in Participatory Development, 1994.
- No 437 **Lars Viklund:** Contributions to a High-level Programming Environment for a Scientific Computing, 1994.
- No 440 **Peter Loborg:** Error Recovery Support in Manufacturing Control Systems, 1994.
- FHS 3/94 **Owen Eriksson:** Informationssystem med verksamhetskvalitet - utvärdering baserat på ett verksamhetsinriktat och samskapande perspektiv, 1994.
- FHS 4/94 **Karin Pettersson:** Informationssystemstrukturer, ansvarsfördelning och användarinflytande - En komparativ studie med utgångspunkt i två informationssystemstrategier, 1994.
- No 441 **Lars Poignant:** Informationsteknologi och företagsetablering - Effekter på produktivitet och region, 1994.
- No 446 **Gustav Fahl:** Object Views of Relational Data in Multidatabase Systems, 1994.
- No 450 **Henrik Nilsson:** A Declarative Approach to Debugging for Lazy Functional Languages, 1994.
- No 451 **Jonas Lind:** Creditor - Firm Relations: an Interdisciplinary Analysis, 1994.
- No 452 **Martin Sköld:** Active Rules based on Object Relational Queries - Efficient Change Monitoring Techniques, 1994.
- No 455 **Pär Carlshamre:** A Collaborative Approach to Usability Engineering: Technical Communicators and System Developers in Usability-Oriented Systems Development, 1994.
- FHS 5/94 **Stefan Cronholm:** Varför CASE-verktyg i systemutveckling? - En motiv- och konsekvensstudie avseende arbetssätt och arbetsformer, 1994.
- No 462 **Mikael Lindvall:** A Study of Traceability in Object-Oriented Systems Development, 1994.
- No 463 **Fredrik Nilsson:** Strategi och ekonomisk styrning - En studie av Sandviks förvärv av Bahco Verktyg, 1994.
- No 464 **Hans Olsén:** Collage Induction: Proving Properties of Logic Programs by Program Synthesis, 1994.
- No 469 **Lars Karlsson:** Specification and Synthesis of Plans Using the Features and Fluents Framework, 1995.
- No 473 **Ulf Söderman:** On Conceptual Modelling of Mode Switching Systems, 1995.
- No 475 **Choong-ho Yi:** Reasoning about Concurrent Actions in the Trajectory Semantics, 1995.
- No 476 **Bo Lagerström:** Successiv resultatavräkning av pågående arbeten. - Fallstudier i tre byggföretag, 1995.
- No 478 **Peter Jonsson:** Complexity of State-Variable Planning under Structural Restrictions, 1995.
- FHS 7/95 **Anders Avdic:** Arbetsintegrerad systemutveckling med kalkylprogram, 1995.
- No 482 **Eva L Ragnemalm:** Towards Student Modelling through Collaborative Dialogue with a Learning Companion, 1995.
- No 488 **Eva Toller:** Contributions to Parallel Multiparadigm Languages: Combining Object-Oriented and Rule-Based Programming, 1995.
- No 489 **Erik Stoy:** A Petri Net Based Unified Representation for Hardware/Software Co-Design, 1995.
- No 497 **Johan Herber:** Environment Support for Building Structured Mathematical Models, 1995.
- No 498 **Stefan Svenberg:** Structure-Driven Derivation of Inter-Lingual Functor-Argument Trees for Multi-Lingual Generation, 1995.
- No 503 **Hee-Cheol Kim:** Prediction and Postdiction under Uncertainty, 1995.
- FHS 8/95 **Dan Fristedt:** Metoder i användning - mot förbättring av systemutveckling genom situationell metodkunskap och metodanalys, 1995.
- FHS 9/95 **Malin Bergvall:** Systemförvaltning i praktiken - en kvalitativ studie avseende centrala begrepp, aktiviteter och ansvarsroller, 1995.
- No 513 **Joachim Karlsson:** Towards a Strategy for Software Requirements Selection, 1995.
- No 517 **Jakob Axelsson:** Schedulability-Driven Partitioning of Heterogeneous Real-Time Systems, 1995.
- No 518 **Göran Forslund:** Toward Cooperative Advice-Giving Systems: The Expert Systems Experience, 1995.
- No 522 **Jörgen Andersson:** Bilder av småföretagares ekonomistyrning, 1995.
- No 538 **Staffan Flodin:** Efficient Management of Object-Oriented Queries with Late Binding, 1996.
- No 545 **Vadim Engelson:** An Approach to Automatic Construction of Graphical User Interfaces for Applications in Scientific Computing, 1996.
- No 546 **Magnus Werner :** Multidatabase Integration using Polymorphic Queries and Views, 1996.
- FIF-a 1/96 **Mikael Lind:** Affärsprocessinriktad förändringsanalys - utveckling och tillämpning av synsätt och metod, 1996.
- No 549 **Jonas Hallberg:** High-Level Synthesis under Local Timing Constraints, 1996.
- No 550 **Kristina Larsen:** Förutsättningar och begränsningar för arbete på distans - erfarenheter från fyra svenska företag, 1996.
- No 557 **Mikael Johansson:** Quality Functions for Requirements Engineering Methods, 1996.
- No 558 **Patrik Nordling:** The Simulation of Rolling Bearing Dynamics on Parallel Computers, 1996.
- No 561 **Anders Ekman:** Exploration of Polygonal Environments, 1996.
- No 563 **Niclas Andersson:** Compilation of Mathematical Models to Parallel Code, 1996.
- No 567 **Johan Jenvall:** Simulation and Data Collection in Battle Training, 1996.
- No 575 **Niclas Ohlsson:** Software Quality Engineering by Early Identification of Fault-Prone Modules, 1996.
- No 576 **Mikael Ericsson:** Commenting Systems as Design Support—A Wizard-of-Oz Study, 1996.
- No 587 **Jörgen Lindström:** Chefers användning av kommunikationsteknik, 1996.
- No 589 **Esa Falkenroth:** Data Management in Control Applications - A Proposal Based on Active Database Systems, 1996.
- No 591 **Niclas Wahllöf:** A Default Extension to Description Logics and its Applications, 1996.
- No 595 **Annika Larsson:** Ekonomisk Styrning och Organisatorisk Passion - ett interaktivt perspektiv, 1997.
- No 597 **Ling Lin:** A Value-based Indexing Technique for Time Sequences, 1997.

- No 598 **Rego Granlund:** C<sup>3</sup>Fire - A Microworld Supporting Emergency Management Training, 1997.
- No 599 **Peter Ingels:** A Robust Text Processing Technique Applied to Lexical Error Recovery, 1997.
- No 607 **Per-Arne Persson:** Toward a Grounded Theory for Support of Command and Control in Military Coalitions, 1997.
- No 609 **Jonas S Karlsson:** A Scalable Data Structure for a Parallel Data Server, 1997.
- FiF-a 4 **Carita Åbom:** Videomötesteknik i olika affärssituationer - möjligheter och hinder, 1997.
- FiF-a 6 **Tommy Wedlund:** Att skapa en företagsanpassad systemutvecklingsmodell - genom rekonstruktion, värdering och vidareutveckling i T50-bolag inom ABB, 1997.
- No 615 **Silvia Coradeschi:** A Decision-Mechanism for Reactive and Coordinated Agents, 1997.
- No 623 **Jan Ollinen:** Det flexibla kontorets utveckling på Digital - Ett stöd för multifix? 1997.
- No 626 **David Byers:** Towards Estimating Software Testability Using Static Analysis, 1997.
- No 627 **Fredrik Eklund:** Declarative Error Diagnosis of GAPLog Programs, 1997.
- No 629 **Gunilla Iwefors:** Krigsspel och Informationsteknik inför en oförutsägbart framtid, 1997.
- No 631 **Jens-Olof Lindh:** Analysing Traffic Safety from a Case-Based Reasoning Perspective, 1997
- No 639 **Jukka Mäki-Turja:** Smalltalk - a suitable Real-Time Language, 1997.
- No 640 **Juha Takkinen:** CAFE: Towards a Conceptual Model for Information Management in Electronic Mail, 1997.
- No 643 **Man Lin:** Formal Analysis of Reactive Rule-based Programs, 1997.
- No 653 **Mats Gustafsson:** Bringing Role-Based Access Control to Distributed Systems, 1997.
- FiF-a 13 **Boris Karlsson:** Metodanalys för förståelse och utveckling av systemutvecklingsverksamhet. Analys och värdering av systemutvecklingsmodeller och dess användning, 1997.
- No 674 **Marcus Bjärelund:** Two Aspects of Automating Logics of Action and Change - Regression and Tractability, 1998.
- No 676 **Jan Håkegård:** Hierarchical Test Architecture and Board-Level Test Controller Synthesis, 1998.
- No 668 **Per-Ove Zetterlund:** Normering av svensk redovisning - En studie av tillkomsten av Redovisningsrådets rekommendation om koncernredovisning (RR01:91), 1998.
- No 675 **Jimmy Tjäder:** Projektledaren & planen - en studie av projektledning i tre installations- och systemutvecklingsprojekt, 1998.
- FiF-a 14 **Ulf Melin:** Informationssystem vid ökad affärs- och processorientering - egenskaper, strategier och utveckling, 1998.
- No 695 **Tim Heyer:** COMPASS: Introduction of Formal Methods in Code Development and Inspection, 1998.
- No 700 **Patrik Hägglund:** Programming Languages for Computer Algebra, 1998.
- FiF-a 16 **Marie-Therese Christiansson:** Inter-organisatorisk verksamhetsutveckling - metoder som stöd vid utveckling av partnerskap och informationssystem, 1998.
- No 712 **Christina Wennestam:** Information om immateriella resurser. Investeringar i forskning och utveckling samt i personal inom skogsindustrin, 1998.
- No 719 **Joakim Gustafsson:** Extending Temporal Action Logic for Ramification and Concurrency, 1998.
- No 723 **Henrik André-Jönsson:** Indexing time-series data using text indexing methods, 1999.
- No 725 **Erik Larsson:** High-Level Testability Analysis and Enhancement Techniques, 1998.
- No 730 **Carl-Johan Westin:** Informationsförsörjning: en fråga om ansvar - aktiviteter och uppdrag i fem stora svenska organisationers operativa informationsförsörjning, 1998.
- No 731 **Åse Jansson:** Miljöhänsyn - en del i företags styrning, 1998.
- No 733 **Thomas Padron-McCarthy:** Performance-Polymorphic Declarative Queries, 1998.
- No 734 **Anders Bäckström:** Värdeskapande kreditgivning - Kreditriskhantering ur ett agentteoretiskt perspektiv, 1998.
- FiF-a 21 **Ulf Seigerroth:** Integration av förändringsmetoder - en modell för välgrundad metodintegration, 1999.
- FiF-a 22 **Fredrik Öberg:** Object-Oriented Frameworks - A New Strategy for Case Tool Development, 1998.
- No 737 **Jonas Mellin:** Predictable Event Monitoring, 1998.
- No 738 **Joakim Eriksson:** Specifying and Managing Rules in an Active Real-Time Database System, 1998.
- FiF-a 25 **Bengt E W Andersson:** Samverkande informationssystem mellan aktörer i offentliga åtaganden - En teori om aktörsarenor i samverkan om utbyte av information, 1998.
- No 742 **Pawel Pietrzak:** Static Incorrectness Diagnosis of CLP (FD), 1999.
- No 748 **Tobias Ritzau:** Real-Time Reference Counting in RT-Java, 1999.
- No 751 **Anders Ferntoft:** Elektronisk affärskommunikation - kontaktkostnader och kontaktprocesser mellan kunder och leverantörer på producentmarknader, 1999.
- No 752 **Jo Skåmedal:** Arbete på distans och arbetsformens påverkan på resor och resmönster, 1999.
- No 753 **Johan Alvehus:** Mötets metaforer. En studie av berättelser om möten, 1999.
- No 754 **Magnus Lindahl:** Bankens villkor i låneavtal vid kreditgivning till högt belånade företagsförvärv: En studie ur ett agentteoretiskt perspektiv, 2000.
- No 766 **Martin V. Howard:** Designing dynamic visualizations of temporal data, 1999.
- No 769 **Jesper Andersson:** Towards Reactive Software Architectures, 1999.
- No 775 **Anders Henriksson:** Unique kernel diagnosis, 1999.
- FiF-a 30 **Pär J. Ågerfalk:** Pragmatization of Information Systems - A Theoretical and Methodological Outline, 1999.
- No 787 **Charlotte Björkegren:** Learning for the next project - Bearers and barriers in knowledge transfer within an organisation, 1999.
- No 788 **Håkan Nilsson:** Informationsteknik som drivkraft i granskningsprocessen - En studie av fyra revisionsbyråer, 2000.
- No 790 **Erik Berglund:** Use-Oriented Documentation in Software Development, 1999.
- No 791 **Klas Gäre:** Verksamhetsförändringar i samband med IS-införande, 1999.
- No 800 **Anders Subotic:** Software Quality Inspection, 1999.
- No 807 **Svein Bergum:** Managerial communication in telework, 2000.

- No 809 **Flavius Gruian:** Energy-Aware Design of Digital Systems, 2000.  
FiF-a 32 **Karin Hedström:** Kunskapsanvändning och kunskapsutveckling hos verksamhetskonsulter - Erfarenheter från ett FOU-samarbete, 2000.
- No 808 **Linda Askenäs:** Affärssystemet - En studie om teknikens aktiva och passiva roll i en organisation, 2000.  
No 820 **Jean Paul Meynard:** Control of industrial robots through high-level task programming, 2000.  
No 823 **Lars Hult:** Publika Gränssytor - ett designexempel, 2000.  
No 832 **Paul Pop:** Scheduling and Communication Synthesis for Distributed Real-Time Systems, 2000.  
FiF-a 34 **Göran Hultgren:** Nätverksinriktad Förändringsanalys - perspektiv och metoder som stöd för förståelse och utveckling av affärsrelationer och informationssystem, 2000.
- No 842 **Magnus Kald:** The role of management control systems in strategic business units, 2000.  
No 844 **Mikael Cäker:** Vad kostar kunden? Modeller för intern redovisning, 2000.  
FiF-a 37 **Ewa Bräf:** Organisationers kunskapsverksamheter - en kritisk studie av "knowledge management", 2000.  
FiF-a 40 **Henrik Lindberg:** Webbaserade affärsprocesser - Möjligheter och begränsningar, 2000.  
FiF-a 41 **Benneth Christiansson:** Att komponentbasera informationssystem - Vad säger teori och praktik?, 2000.  
No. 854 **Ola Pettersson:** Deliberation in a Mobile Robot, 2000.  
No 863 **Dan Lawesson:** Towards Behavioral Model Fault Isolation for Object Oriented Control Systems, 2000.  
No 881 **Johan Moe:** Execution Tracing of Large Distributed Systems, 2001.  
No 882 **Yuxiao Zhao:** XML-based Frameworks for Internet Commerce and an Implementation of B2B e-procurement, 2001.
- No 890 **Annika Flycht-Eriksson:** Domain Knowledge Management in Information-providing Dialogue systems, 2001.  
FiF-a 47 **Per-Arne Segerkvist:** Webbaserade imaginära organisationers samverkansformer: Informationssystemmarkitektur och aktörssamverkan som förutsättningar för affärsprocesser, 2001.
- No 894 **Stefan Svarén:** Styrning av investeringar i divisionaliserade företag - Ett koncernperspektiv, 2001.  
No 906 **Lin Han:** Secure and Scalable E-Service Software Delivery, 2001.  
No 917 **Emma Hansson:** Optionsprogram för anställda - en studie av svenska börsföretag, 2001.  
No 916 **Susanne Odar:** IT som stöd för strategiska beslut, en studie av datorimplementerade modeller av verksamhet som stöd för beslut om anskaffning av JAS 1982, 2002.
- FiF-a-49 **Stefan Holgersson:** IT-system och filtrering av verksamhetskunskap - kvalitetsproblem vid analyser och beslutsfattande som bygger på uppgifter hämtade från polisens IT-system, 2001.  
FiF-a-51 **Per Oscarsson:** Informationssäkerhet i verksamheter - begrepp och modeller som stöd för förståelse av informationssäkerhet och dess hantering, 2001.
- No 919 **Luis Alejandro Cortes:** A Petri Net Based Modeling and Verification Technique for Real-Time Embedded Systems, 2001.  
No 915 **Niklas Sandell:** Redovisning i skuggan av en bankkris - Värdering av fastigheter, 2001.  
No 931 **Fredrik Elg:** Ett dynamiskt perspektiv på individuella skillnader av heuristisk kompetens, intelligens, mentala modeller, mål och konfidens i kontroll av mikrovärlden Moro, 2002.  
No 933 **Peter Aronsson:** Automatic Parallelization of Simulation Code from Equation Based Simulation Languages, 2002.
- No 938 **Bourhane Kadmiry:** Fuzzy Control of Unmanned Helicopter, 2002.  
No 942 **Patrik Haslum:** Prediction as a Knowledge Representation Problem: A Case Study in Model Design, 2002.  
No 956 **Robert Sevenius:** On the instruments of governance - A law & economics study of capital instruments in limited liability companies, 2002.
- FiF-a 58 **Johan Petersson:** Lokala elektroniska marknadsplatser - informationssystem för platsbundna affärer, 2002.  
No 964 **Peter Bunus:** Debugging and Structural Analysis of Declarative Equation-Based Languages, 2002.  
No 973 **Gert Jerwan:** High-Level Test Generation and Built-In Self-Test Techniques for Digital Systems, 2002.  
No 958 **Fredrika Berglund:** Management Control and Strategy - a Case Study of Pharmaceutical Drug Development, 2002.
- FiF-a 61 **Fredrik Karlsson:** Meta-Method for Method Configuration - A Rational Unified Process Case, 2002.  
No 985 **Sorin Manolache:** Schedulability Analysis of Real-Time Systems with Stochastic Task Execution Times, 2002.
- No 982 **Diana Szentiványi:** Performance and Availability Trade-offs in Fault-Tolerant Middleware, 2002.  
No 989 **Iakov Nakhimovski:** Modeling and Simulation of Contacting Flexible Bodies in Multibody Systems, 2002.  
No 990 **Levon Saldamli:** PDEModelica - Towards a High-Level Language for Modeling with Partial Differential Equations, 2002.
- No 991 **Almut Herzog:** Secure Execution Environment for Java Electronic Services, 2002.  
No 999 **Jon Edvardsson:** Contributions to Program- and Specification-based Test Data Generation, 2002.  
No 1000 **Anders Arpteg:** Adaptive Semi-structured Information Extraction, 2002.  
No 1001 **Andrzej Bednarski:** A Dynamic Programming Approach to Optimal Retargetable Code Generation for Irregular Architectures, 2002.
- No 988 **Mattias Arvola:** Good to use! : Use quality of multi-user applications in the home, 2003.  
FiF-a 62 **Lennart Ljung:** Utveckling av en projektivtetsmodell - om organisationers förmåga att tillämpa projektarbetsformen, 2003.
- No 1003 **Pernilla Qvarfordt:** User experience of spoken feedback in multimodal interaction, 2003.  
No 1005 **Alexander Siemers:** Visualization of Dynamic Multibody Simulation With Special Reference to Contacts, 2003.
- No 1008 **Jens Gustavsson:** Towards Unanticipated Runtime Software Evolution, 2003.  
No 1010 **Calin Curescu:** Adaptive QoS-aware Resource Allocation for Wireless Networks, 2003.  
No 1015 **Anna Andersson:** Management Information Systems in Process-oriented Healthcare Organisations, 2003.  
No 1018 **Björn Johansson:** Feedforward Control in Dynamic Situations, 2003.  
No 1022 **Traian Pop:** Scheduling and Optimisation of Heterogeneous Time/Event-Triggered Distributed Embedded Systems, 2003.
- FiF-a 65 **Britt-Marie Johansson:** Kundkommunikation på distans - en studie om kommunikationsmediets betydelse i affärstransaktioner, 2003.

- No 1024 **Aleksandra Tešanovic:** Towards Aspectual Component-Based Real-Time System Development, 2003.  
No 1034 **Arja Vainio-Larsson:** Designing for Use in a Future Context - Five Case Studies in Retrospect, 2003.  
No 1033 **Peter Nilsson:** Svenska bankers redovisningsval vid reservering för befarade kreditförluster - En studie vid införandet av nya redovisningsregler, 2003.
- FiF-a 69 **Fredrik Ericsson:** Information Technology for Learning and Acquiring of Work Knowledge, 2003.  
No 1049 **Marcus Comstedt:** Towards Fine-Grained Binary Composition through Link Time Weaving, 2003.  
No 1052 **Åsa Hedenskog:** Increasing the Automation of Radio Network Control, 2003.  
No 1054 **Claudiu Duma:** Security and Efficiency Tradeoffs in Multicast Group Key Management, 2003.  
FiF-a 71 **Emma Eliason:** Effekttanalys av IT-systems handlingsutrymme, 2003.  
No 1055 **Carl Cederberg:** Experiments in Indirect Fault Injection with Open Source and Industrial Software, 2003.  
No 1058 **Daniel Karlsson:** Towards Formal Verification in a Component-based Reuse Methodology, 2003.  
FiF-a 73 **Anders Hjalmarsson:** Att etablera och vidmakthålla förbättringsverksamhet - behovet av koordination och interaktion vid förändring av systemutvecklingsverksamheter, 2004.  
No 1079 **Pontus Johansson:** Design and Development of Recommender Dialogue Systems, 2004.  
No 1084 **Charlotte Stoltz:** Calling for Call Centres - A Study of Call Centre Locations in a Swedish Rural Region, 2004.
- FiF-a 74 **Björn Johansson:** Deciding on Using Application Service Provision in SMEs, 2004.  
No 1094 **Genevieve Gorrell:** Language Modelling and Error Handling in Spoken Dialogue Systems, 2004.  
No 1095 **Ulf Johansson:** Rule Extraction - the Key to Accurate and Comprehensible Data Mining Models, 2004.  
No 1099 **Sonia Sangari:** Computational Models of Some Communicative Head Movements, 2004.  
No 1110 **Hans Nässla:** Intra-Family Information Flow and Prospects for Communication Systems, 2004.  
No 1116 **Henrik Sällberg:** On the value of customer loyalty programs - A study of point programs and switching costs, 2004.
- FiF-a 77 **Ulf Larsson:** Designarbete i dialog - karaktärisering av interaktionen mellan användare och utvecklare i en systemutvecklingsprocess, 2004.  
No 1126 **Andreas Borg:** Contribution to Management and Validation of Non-Functional Requirements, 2004.  
No 1127 **Per-Ola Kristensson:** Large Vocabulary Shorthand Writing on Stylus Keyboard, 2004.  
No 1132 **Pär-Anders Albinsson:** Interacting with Command and Control Systems: Tools for Operators and Designers, 2004.
- No 1130 **Ioan Chisalita:** Safety-Oriented Communication in Mobile Networks for Vehicles, 2004.  
No 1138 **Thomas Gustafsson:** Maintaining Data Consistency in Embedded Databases for Vehicular Systems, 2004.  
No 1149 **Vaida Jakonienė:** A Study in Integrating Multiple Biological Data Sources, 2005.  
No 1156 **Abdil Rashid Mohamed:** High-Level Techniques for Built-In Self-Test Resources Optimization, 2005.  
No 1162 **Adrian Pop:** Contributions to Meta-Modeling Tools and Methods, 2005.  
No 1165 **Fidel Vascós Palacios:** On the information exchange between physicians and social insurance officers in the sick leave process: an Activity Theoretical perspective, 2005.
- FiF-a 84 **Jenny Lagsten:** Verksamhetsutvecklande utvärdering i informationssystemprojekt, 2005.  
No 1166 **Emma Larsdotter Nilsson:** Modeling, Simulation, and Visualization of Metabolic Pathways Using Modelica, 2005.
- No 1167 **Christina Keller:** Virtual Learning Environments in higher education. A study of students' acceptance of educational technology, 2005.  
No 1168 **Cécile Åberg:** Integration of organizational workflows and the Semantic Web, 2005.  
FiF-a 85 **Anders Forsman:** Standardisering som grund för informationssamverkan och IT-tjänster - En fallstudie baserad på trafikinformationstjänsten RDS-TMC, 2005.
- No 1171 **Yu-Hsing Huang:** A systemic traffic accident model, 2005.  
FiF-a 86 **Jan Olausson:** Att modellera uppdrag - grunder för förståelse av processinriktade informationssystem i transaktionsintensiva verksamheter, 2005.
- No 1172 **Petter Ahlström:** Affärsstrategier för seniorbostadsmarknaden, 2005.  
No 1183 **Mathias Cöster:** Beyond IT and Productivity - How Digitization Transformed the Graphic Industry, 2005.  
No 1184 **Åsa Horzella:** Beyond IT and Productivity - Effects of Digitized Information Flows in Grocery Distribution, 2005.
- No 1185 **Maria Kollberg:** Beyond IT and Productivity - Effects of Digitized Information Flows in the Logging Industry, 2005.  
No 1190 **David Dinka:** Role and Identity - Experience of technology in professional settings, 2005.  
No 1191 **Andreas Hansson:** Increasing the Storage Capacity of Recursive Auto-associative Memory by Segmenting Data, 2005.
- No 1192 **Nicklas Bergfeldt:** Towards Detached Communication for Robot Cooperation, 2005.  
No 1194 **Dennis Maciuszek:** Towards Dependable Virtual Companions for Later Life, 2005.  
No 1204 **Beatrice Alenljung:** Decision-making in the Requirements Engineering Process: A Human-centered Approach, 2005
- No 1206 **Anders Larsson:** System-on-Chip Test Scheduling and Test Infrastructure Design, 2005.  
No 1207 **John Wilander:** Policy and Implementation Assurance for Software Security, 2005.  
No 1209 **Andreas Käll:** Översättningar av en managementmodell - En studie av införandet av Balanced Scorecard i ett landsting, 2005.
- No 1225 **He Tan:** Aligning and Merging Biomedical Ontologies, 2006.  
No 1228 **Artur Wilk:** Descriptive Types for XML Query Language Xcerpt, 2006.  
No 1229 **Per Olof Pettersson:** Sampling-based Path Planning for an Autonomous Helicopter, 2006.  
No 1231 **Kalle Burbeck:** Adaptive Real-time Anomaly Detection for Safeguarding Critical Networks, 2006.  
No 1233 **Daniela Mihailescu:** Implementation Methodology in Action: A Study of an Enterprise Systems Implementation Methodology, 2006.
- No 1244 **Jörgen Skågeby:** Public and Non-public gifting on the Internet, 2006.  
No 1248 **Karolina Eliasson:** The Use of Case-Based Reasoning in a Human-Robot Dialog System, 2006.  
No 1263 **Misook Park-Westman:** Managing Competence Development Programs in a Cross-Cultural Organisation-What are the Barriers and Enablers, 2006.
- FiF-a 90 **Amra Halilovic:** Ett praktikperspektiv på hantering av mjukvarukomponenter, 2006.  
No 1272 **Raquel Flodström:** A Framework for the Strategic Management of Information Technology, 2006.

No 1277 **Viacheslav Izosimov:** Scheduling and Optimization of Fault-Tolerant Embedded Systems, 2006.

No 1283 **Håkan Hasewinkel:** A Blueprint for Using Commercial Games off the Shelf in Defence Training, Education and Research Simulations, 2006.

FiF-a 91 **Hanna Broberg:** Verksamhetsanpassade IT-stöd - Design teori och metod, 2006.

No 1286 **Robert Kaminski:** Towards an XML Document Restructuring Framework, 2006

No 1293 **Jiri Trnka:** Prerequisites for data sharing in emergency management, 2007.

No 1302 **Björn Hägglund:** A Framework for Designing Constraint Stores, 2007.

No 1303 **Daniel Andreasson:** Slack-Time Aware Dynamic Routing Schemes for On-Chip Networks, 2007.

No 1305 **Magnus Ingmarsson:** Modelling User Tasks and Intentions for Service Discovery in Ubiquitous Computing, 2007.

No 1306 **Gustaf Svedjemo:** Ontology as Conceptual Schema when Modelling Historical Maps for Database Storage, 2007.

No 1307 **Gianpaolo Conte:** Navigation Functionalities for an Autonomous UAV Helicopter, 2007.

No 1309 **Ola Leifler:** User-Centric Critiquing in Command and Control: The DKExpert and ComPlan Approaches, 2007.

No 1312 **Henrik Svensson:** Embodied simulation as off-line representation, 2007.

No 1313 **Zhiyuan He:** System-on-Chip Test Scheduling with Defect-Probability and Temperature Considerations, 2007.

No 1317 **Jonas Elmqvist:** Components, Safety Interfaces and Compositional Analysis, 2007.

No 1320 **Håkan Sundblad:** Question Classification in Question Answering Systems, 2007.

No 1323 **Magnus Lundqvist:** Information Demand and Use: Improving Information Flow within Small-scale Business Contexts, 2007.

No 1329 **Martin Magnusson:** Deductive Planning and Composite Actions in Temporal Action Logic, 2007.

No 1331 **Mikael Asplund:** Restoring Consistency after Network Partitions, 2007.

No 1332 **Martin Fransson:** Towards Individualized Drug Dosage - General Methods and Case Studies, 2007.

No 1333 **Karin Camara:** A Visual Query Language Served by a Multi-sensor Environment, 2007.

No 1337 **David Broman:** Safety, Security, and Semantic Aspects of Equation-Based Object-Oriented Languages and Environments, 2007.

No 1339 **Mikhail Chalabine:** Invasive Interactive Parallelization, 2007.

No 1351 **Susanna Nilsson:** A Holistic Approach to Usability Evaluations of Mixed Reality Systems, 2008.

No 1353 **Shanai Ardi:** A Model and Implementation of a Security Plug-in for the Software Life Cycle, 2008.

No 1356 **Erik Kuiper:** Mobility and Routing in a Delay-tolerant Network of Unmanned Aerial Vehicles, 2008.

No 1359 **Jana Rambusch:** Situated Play, 2008.

No 1361 **Martin Karresand:** Completing the Picture - Fragments and Back Again, 2008.