Forensic Analysis Of Large Capacity Digital Storage Devices

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS OF EDINBURGH NAPIER UNIVERSITY FOR THE AWARD OF DOCTOR OF PHILOSOPHY IN THE SCHOOL OF COMPUTING

Philip Penrose

February 2017

Acknowledgements

My thanks must go firstly to my wife and daughter, Margaret and Amanda, who sacrificed my company for many long hours and offered unstinting support throughout this journey. This could not have been done without them.

I would like to thank my Director of Studies, Professor William J Buchanan, for his infectious enthusiasm and the encouragement he has given me over the last three years. My second supervisor, Rich Macfarlane, has been a great support during this time and always gave insightful advice along with the beer.

I am forever grateful for the astounding depth of knowledge possessed by Bruce Ramsay about the low-level workings of processors and operating systems and of the digital forensic processes within law enforcement agencies. I learned so much from him.

Finally I would like to thank all my fellow researchers and staff in the School of Computing who always provided encouragement and support.

Declarations

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

CONTENTS

List Of Figures

List of Tables

Abstract

Digital forensic laboratories are failing to cope with the volume of digital evidence required to be analysed. The ever increasing capacity of digital storage devices only serves to compound the problem. In many law enforcement agencies a form of *administrative triage* takes place by simply dropping perceived low priority cases without reference to the data itself. Security agencies may also need days or weeks to analyse devices in order to detect and quantify encrypted data on the device.

The current methodology often involves agencies creating a hash database of files where each known contraband file is hashed using a forensic hashing algorithm. Each file on a suspect device is similarly hashed and the hash compared against the contraband hash database. Accessing files via the file system in this way is a slow process. In addition deleted files or files on deleted or hidden partitions would not be found since their existence is not recorded in the file system.

This thesis investigates the introduction of a system of triage whereby digital storage devices of arbitrary capacity can be quickly scanned to identify contraband and encrypted content with a high probability of detection with a known and controllable margin of error in a reasonable time. Such a system could classify devices as being worthy of further investigation or not and thus limit the number of devices being presented to digital forensic laboratories for examination.

A system of triage is designed which bypasses the file system and uses the fundamental storage unit of digital storage devices, normally a 4 KiB block, rather than complete files. This allows fast sampling of the storage device. Samples can be chosen to give a controllable margin of error. In addition the sample is drawn from the whole address space of the device and so deleted files and partitions are also sampled. Since only a sample is being examined this is much faster than the traditional digital forensic analysis process.

In order to achieve this, methods are devised that allow firstly the identification of 4 KiB blocks as belonging to a contraband file and secondly the classification of the block as encrypted or not. These methods minimise both memory and CPU loads so that the system may run on legacy equipment that may be in a suspect's possession. A potential problem with the existence of blocks that are common to many files is quantified and a mitigation strategy developed.

The system is tested using publically available corpora by seeding devices with *contraband* and measuring the detection rate during triage. Results from testing are positive, achieving a 99% probability of detecting 4 MiB of contraband on a 1 TB device within the time normally assigned for the interview of the device owner. Initial testing on live devices in a law enforcement environment has shown that sufficient evidence can be collected in under four minutes from a 1TB device to allow the equipment to be seized and the suspect to be charged.

This research will lead to a significant reduction in the backlog of cases in digital forensic laboratories since it can be used for triage within the laboratory as well as at the scene of crime.

1 Introduction

1.1 Digital Forensics

Digital forensics is the practice of collecting and analysing digital data in a way that is admissible in court. Digital evidence is any data stored or transmitted using a computer system that a party in a court case may use at trial. The evidence can range from images of child pornography to encrypted data used to further criminal activities [1]. Indeed computers and networks have become such an integral part of our daily lives that the use of computers in such crimes as fraud, drug trafficking and terrorism seems commonplace. In consequence the increasing number of instances of digital equipment being used in the commission of crime has led to an increase in the number of digital forensic examinations being undertaken. Such examinations need a great deal of knowledge, experience and skill on the part of the forensic analyst and the ever increasing volume of data involved in each case has led the digital forensic community to look for new ways to deal with the vast amount of data involved [2].

1.2 Context

The growing number of digital forensics cases is compounded by an increase in the volume of data needing analysed, and this is a current and growing problem facing the digital forensic community. The capacity of digital storage devices has been increasing at a rate that leaves them struggling to cope with the volume of data needing analysed. The rate of increase is now lower than the historical Kryders's law equivalent of ~40% per annum [3] but is still between 20% and 30% per annum [4]. Terabyte devices are now commonplace. There is an acknowledgement that current forensic tools have failed to keep up and have left digital forensic services struggling to cope [5]. Many common forensic tools were developed in an age when the storage capacity of devices was measured in megabytes and cannot, or simply have not been scaled to the capacity of current devices.

In traditional forensic analysis the first step is to make a forensic image of the device. With well-specified equipment Roussev et al. [6] benchmarked the acquisition of a fast 3 TB hard disk drive using a standard acquisition utility at over 11 hours. The image is then submitted to a digital forensic laboratory for analysis. To find any contraband, if it exists, in this amount of data takes a considerable amount of time. Because of this, a forensic backlog has developed. In the UK three months between equipment being seized and a forensic report being available is considered normal [7] and delays of 12 months are not uncommon [8], [9].

In one law enforcement organisation the case backlog was found to be three years [10]. In addition, budgetary constraints and the high level of training required has led to a shortage of digital forensic analysts and this has added to the time taken between an investigator sending the digital evidence for analysis and receiving an analytical report [11]. Pollitt [12] argues that where data storage is measured in terabytes, a thorough examination of such devices is unrealistic. In an attempt to overcome these problems a form of *administrative triage* takes place in many organisations where perceived low priority jobs are delayed or dropped without reference to the data itself [13]. However, it is more than an administrative problem. In the case of equipment seized when possession of child pornography is suspected, then during the wait for any seized equipment to be examined many suspects have committed suicide, even when eventually no illegal content was found on the devices [14], [15]. In the United States the Federal Bureau of Investigation acknowledges that to some offenders, the investigation

into their alleged offences may lead them to view suicide as a preferred alternative to the loss of respect of family and friends and the possible loss of their job, home, reputation, and freedom and are therefore mindful of this when dealing with such cases [16].

A further problem caused by the large capacity of digital storage devices presents itself when devices are being inspected at border posts, where the volume of data even on laptop computers makes a full scan of a computer for contraband material impractical to be done in any reasonable time. In this situation the absence of contraband files is not the only problem. The quantity of encrypted data on a digital storage device may significantly affect the attitude of a border guard to the profile of the device owner. An inability or unwillingness to produce the encryption key would lead, at the very least, to the device being held for further detailed examination. Thus in this situation a reliable reporting of the quantity of encrypted material on a device is crucial.

This thesis develops methods to mitigate these problems. To address the problem of the backlog in digital forensic labs and the time needed for discovery of contraband material a method of triage is developed where devices may be sampled quickly and, depending on the contraband material found, a decision made on whether the device is worthy of further investigation or not. In addition, if any contraband material is found, its location is recorded and this eases the subsequent task of locating contraband material on the device. At the same time as the device is being scanned for contraband the same samples can be classified by original file type to estimate the quantity of encrypted content.

The triage process developed is based on sampling the devices. During consultation with Police Scotland (the national police body in Scotland) it was specified that the methodology must ensure that triage is completed in a reasonable time, which they define as the time that would normally be allocated for the interview of a suspect in their home which is generally around one hour. If there is contraband on a device then it must be 99% certain of detecting the contraband, even in cases where the suspect has attempted to delete the contraband or hide it in areas such as deleted and hidden partitions or in unallocated space on the disc.

At present this can only be achieved by detailed analysis of the device by a digital forensic analyst. The current methodology is file based, with contraband files being detected by their hash signature. The hash of each file in the contraband corpus is calculated and stored in a database. During analysis in forensic laboratories, each file on a disk is read, hashed and the hash compared with those in the contraband hash database. This methodology, even when automated, is slow and, since it depends on the file system, deleted or hidden areas of the device are not examined. This thesis develops methodology where the smallest addressable unit of storage on a device, a cluster, is sampled and used for identification of contraband. This allows the whole address space of the device to be sampled quickly and so deleted, hidden and unallocated areas of the device are also examined.

In parallel with the identification of the cluster it can be classified as encrypted or not during the triage process. Current methodology can classify a small fragment of a file such as the 4 KiB clusters read from the device during the triage process quite accurately for common file types such as JPEG, HTML or DOC. However these methods fail to differentiate between high entropy file types such as compressed or encrypted fragments. The entropy of a file fragment measures the amount of randomness or disorder in the data within it. Compressed and encrypted files have little pattern or order to detect and Roussev and Garfinkel [17] suggest that it is not possible using statistical and machine learning techniques to differentiate such fragments. Garfinkel et al. [18] state that the classification of high entropy file fragments is in its infancy and needs further research and it is suggested that future work should investigate methods to improve classification performance on such fragments. This thesis

develops methodology whereby encrypted and compressed file fragments can be differentiated with better than 90% accuracy.

The methods developed in this thesis speed up the digital forensic process in several ways. Firstly devices can quickly be *triaged*, and a decision made whether a device needs further investigation or not. This reduces the number of devices passed to the forensic facility and thus reduces workload. Secondly, if a device is found to have contraband, the location on the device of the sampled contraband is already known; thus aiding the analyst in the analysis of the device. Thirdly, as the triage progresses, the device content is profiled and the proportion of encrypted material reported.

1.3 Thesis Aims and Objectives

1.3.1 Aim

To mitigate the problems identified in the previous section the aim of this thesis is to produce a system of forensic triage that will quickly identify if a device is worthy of further investigation because of either contraband or encrypted content.

1.3.2 Objectives

To meet this aim the following objectives need to be achieved:

- to identify new approaches that may improve existing systems by investigating and critically evaluating current practices in the identification of contraband and encrypted data on large capacity digital devices.
- to create an application that can scan digital storage devices of arbitrary capacity to identify contraband with a high probability of detection with a known and controllable margin of error, under the constraints imposed by low specification legacy equipment, in a reasonable time.
- to create an application that can scan digital storage devices of arbitrary capacity to estimate the quantity of encrypted data with a known and controllable margin of error, under the constraints imposed by low specification legacy equipment, in a reasonable time.
- to test that systems developed meet the criteria for a new system of digital forensic triage.

The system developed in this thesis meets these aims and objectives and has been field tested by Police Scotland and found to meet the criteria for a fast initial scan to classify a device as worthy of further analysis or not.

1.4 Contributions and Impact

This research has already had considerable impact in the digital forensic community. Three papers which directly derive from this research and its findings have been published in peer-reviewed journals:

- Approaches to the classification of high entropy file fragments. *Digital Investigation*, 10(4), pp.372–384 [19]
- Fast contraband detection in large capacity disk drives. *Digital Investigation*, 12, pp.S22–S29 [20]. This paper was selected for presentation at the 2015 Digital Forensic Research Workshops (Europe) conference [21].
- The Effect Of Non-probative Blocks On Disk Sampling For Forensic Triage. *Digital Investigation,* DIIN-D-16-00034, in press [22].

These publications have led to collaborative work with leading researchers in the field – Simson Garfinkel on non-probative blocks and Frank Breitinger on cuckoo filters and sampling of network streams.

The research has won the Research Excellence Award for The Information Society (2016) at the annual research conference hosted by Edinburgh Napier University.

Three patents have been applied for on different aspects of this research.

Funding towards proof of concept for commercial applications based on this research was received from Scottish Enterprise and after a competitive process further funding towards commercialisation was funded by The Home Office. A spinout company which is developing applications based on this research has been funded by venture capital and is now actively developing applications.

Police Scotland tested the software developed during the course of this research and have found that it fulfils their criteria for a triage tool: being 99% accurate; give results in a reasonable time; and execute on low-specification legacy equipment. As an example, in a recent live case, sufficient evidence was collected in four minutes from a one terabyte drive to justify the seizure of the equipment and arrest the suspect (Police Scotland, personal communication, 5th May 2016). The system of triage is now being evaluated against the UK Child Abuse Image Database [23].

This thesis advances the knowledge and techniques in the area of digital forensic analysis. There are three major contributions. The technique of detecting contraband files by sampling raw disk clusters and bypassing the file system has been introduced. This replaces the traditional file hashing method for the detection of contraband and combines cluster hashing with sampling and the use of Bloom filters in a novel way to provide a method that can detect contraband on a large capacity digital storage device to a required degree of accuracy in a reasonable time.

Secondly the problem of how to deal with non-probative disk clusters was raised in the literature [22], [24]. Non-probative clusters are those that may appear in many files and so can't be used to prove the existence of a particular file on the media. This could invalidate any claim that a device contained contraband. Although this problem has been acknowledged in current research, its overall effect on the triage process has not been investigated. In this thesis the scale of this problem has been quantified by the analysis of several multi-million block corpora and an effective mitigation strategy has been devised.

Thirdly, a method of identifying encrypted data on a device has been introduced. Previous research into the classification of high entropy file fragments such as compressed and encrypted data is very limited since this area of research has been recognised as difficult [18], [25]–[27]. By the novel use of statistical tests for randomness and neural networks for classification this thesis has introduced a method which can correctly detect over 95% of encrypted and 75% of compressed disk clusters. In addition, the method developed can report on the volume of encrypted data on a device consistently to within 3% of the actual content.

The overall contribution of this thesis is to have advanced research in the area of digital forensic triage by devising novel methodologies and implementing prototypes that will reduce the digital forensic backlog that currently exists in digital forensic labs and at border posts [8], [9]. It delivers a more ethical process for the digital forensic community whereby suspects, even those whose devices are eventually found to be free of contraband material, are not left in a state of suspense for many months while waiting for devices to be forensically examined.

1.5 Structure of The Thesis

The remainder of this thesis can be seen as being made up of two main themes: file fragment identification; and file fragment classification. Chapters 2 and 3 investigate the problem of file fragment identification and methods of contraband detection are proposed and tested. Chapters 4 to 7 investigate the more intractable problem of classifying encrypted data and a new methodology is developed and tested. Chapter 8 summarises the achievements of this research and looks forward to future work. The content is as follows.

- Chapter 2. The objectives of developing methods that can identify contraband on a digital storage device with a high probability of detection in a reasonable time are investigated. Previous work on triage solutions is critically reviewed and methods that could possibly lead to a better triage solution assessed. These methods are examined further and a quantitative analysis undertaken to establish if they can be used in a system that will meet the requirements of a triage system. A design is established using sampling of clusters - the fundamental unit of storage on a device - for a contraband detection system which uses Bloom filters for fast cluster identification and this is implemented and tested.
- Chapter 3. The problem posed by non-probative clusters is investigated. A non-probative cluster is one that may appear in many files and so cannot be used to prove that the original file was originally present on the device. The scale of problem is quantified and mitigation strategies developed.
- Chapter 4. The current state of the art in file fragment classification is critically reviewed and the problem with classification of high entropy fragments is highlighted. This leads to formulating hypotheses which may be used to classify high entropy fragments such as compressed and encrypted data.
- Chapter 5. Experiments to test the hypotheses are designed using publically available corpora for testing so that the results may be validated by other researchers.
- Chapter 6. The proposed classification system is implemented and tested.
- Chapter 7. Methods are developed to improve the accuracy of the classification and a proof of concept classifier implemented and tested.
- Chapter 8. This chapter summarises the contributions of this research and then looks ahead to extending the scope. There are many possibilities for future work and some of these are considered.

2 Cluster Identification

2.1 Introduction

Kryder [28] shows that the areal density - which is the number of bits stored per unit area of disk - has been increasing at 40% per year, and this is projected to continue for the foreseeable future since the technology is, as yet, nowhere near fundamental limits. This is referred to as *Kryder's Law* [3] and is analogous to Moore's Law for semiconductors. Garfinkel [5] claims that, because of this, much of the progress made in digital forensic tools over the last decade is becoming irrelevant. These tools were designed to help forensic examiners to find evidence, usually from a relatively low capacity hard disk drive, and do not scale to the capacity of digital storage devices commonly available today.

In the UK the Association of Chief Police Officers (ACPO) [29] has acknowledged this situation. Many digital forensics units thus have large backlogs and the rate of technological change is likely to accelerate and so exacerbate the situation. They say that where there is insufficient time or resources to cope with the volume of digital devices being presented a system of forensic triage should be introduced. In this thesis the term triage will be used to mean a fast initial scan by sampling a digital device, conducted perhaps under severe time and resource constraints, to prioritise the device for possible further detailed investigation. Casey [30] maintains that there is still a pressing need for such tool development to help detect and analyse the ever-increasing volume of digital evidence. Young et al. [31] argue that it is critical for forensic investigators to have such a triage process so that they can quickly detect illegal files on large capacity devices.

In discussions with analysts from the Police Scotland Digital Forensics Department they listed their requirements for such a forensic tool as a system that should:

- Be 99% certain of detecting contraband including contraband that may have been deleted, or in hidden partitions.
- Give results in a reasonable time.
- Run on possibly low specification legacy equipment.
- Be able to be used by trained personnel who are not necessarily digital forensic analysts.

The term *reasonable time* is defined as the time between an initial interview with the suspect at first contact in the suspect's home, and which generally lasts at least an hour. The system of triage developed in this thesis will be designed to meet these requirements.

This chapter thus aims to develop a system that can quickly scan a storage device and report with good accuracy if the device contains contraband. Techniques for achieving this are explored and explained and a proof of concept system developed and tested.

2.2 Related Work

Previous work in the field shows that such a system of triage is needed within the digital forensic community and also provides some pointers to possible methods that may be adapted to produce such a system.

Pollitt [32] argues that the process of digital forensic triage is an admission of failure. The backlog of cases is often due to the systemic failure of the digital forensic process, and of digital forensic software. These have not adapted to the vast increase in digital data that is involved in a modern case. Triage has become necessary because investigators often prefer some useful evidence quickly rather than wait, perhaps some considerable time, for all detectable evidence to be found. He argues that by focusing on a particular outcome such as the existence of specific types of data, important information is missed, such as logs or e-mail that might reveal a wider group of suspects. However triage must not be the only tool used in an investigation. If any incriminating evidence is found during the process of triage, the device should be subject to a full digital forensic analysis. Shaw and Browne [13] note that administrative triage already takes place in many organisations and criteria are used to either prioritise, or exclude a device, from an examination, without considering the device content.

Horsman et al. [33] maintain that organisations may also be cautious because there is a perception that there is a risk of missing evidence where triage only samples a device. However, especially in the scenario of detecting contraband, there is a lesser risk in a system of triage allowing the timely analysis of a device using forensically sound boot media, and with a controllable probability of missed evidence, than a system of administrative triage which operates without any reference to the physical media. A survey of practising digital forensic analysts by Harichandran et al. [34] found that 88% accepted that automatic triage systems were a future challenge needing research.

2.2.1 Existing triage solutions

There are a number of triage packages available, both open source and proprietary, such as Strike, EnCase Portable, AD Triage, Triage IR, Kludge. These packages typically perform data collection, often with no intervention by the operator, of such things as internet history, the registry, file metadata, recently used files, image files, hash lookup of user home directory files and comparing to a selected hash database, indexing, keyword matching and so on. Some even do full disk imaging as part of triage. File carving from unallocated disk space is also an option on some. They uniformly behave as versions of full forensic analysis packages, collecting appropriate data for later examination. Most are designed so that they can be used by an untrained operative, and give on-screen display of images or analysis results as it is produced.

Casey et al. [35] view this type of product as freeing forensic analysts from the routine task of acquiring forensic evidence and empowering them to concentrate on the more interesting aspects of their work. However, these tools might be looked at as automating the acquisition stage of a forensic investigation rather than as triage. In addition, these tools are complex and incorrect configuration could easily lead to the wrong conclusions being drawn regarding the result of the triage process [13]. All the operations performed tend to be I/O and processor intensive and defeat the purpose of the definition of triage used in this thesis - a fast initial scan to ascertain if a device contains images or documents of interest.

Quick and Choo [36] suggest that, during device acquisition, only *files of interest* such as registry, documents, images, and other relevant file types are retained for examination thus cutting the workload. This is a *data reduction* strategy rather than triage. It does not reduce the data acquisition time, as the entire file system has to be examined for files of interest and could easily be defeated by the simple measure of changing the file type extension. It also does not take into account deleted or hidden data. Neither does it reduce the number of cases needing analysis.

Roussev et al. [6] treat triage as an intrinsic part of the digital forensic process. They advocate that target acquisition and forensic processing should be done in parallel, with results being

reported as soon as they are available. Their model requires that data is analysed as it is being acquired so that analysis should finish at the same time as the data acquisition. This requires that an analysis, including cryptographic hashing and lookup, similarity hashing, decompression, file content extraction and indexing, should be done in parallel at the speed of data acquisition. However they acknowledge that to achieve this goal requires 2 to 4 servers with 48 or 64 cores with sufficient RAM for in-RAM processing. Garfinkel [37] developed the application *bulk_extractor* to scan an entire disk image. It used a number of filters running in parallel, each optimised to detect patterns indicative of one of the common artefacts required for a digital investigation such as telephone numbers, e-mail addresses or credit card numbers and did not address the detection of contraband files since it did not refer to a database of known content.

2.2.2 Hashing and Bloom filters – identifying known content

Present practice for the identification of known files attempts to ascertain whether the device contains material that has originally come from some known 'good' corpus (whitelists) such as the National Software Reference Library hash database (NSRL) issued by the National Institute of Standards and Technology (NIST), or from a corpus of illegal material such as that of child pornography held by Police Scotland or the Team Cymru Malware Hash Registry (blacklists). These databases hold fixed-length cryptographic hashes of each file, for example the 128-bit MD5 hash or the 160-bit SHA-1 hash, rather than the files themselves. An unknown file may be identified by calculating its hash and comparing this hash against the database. It was argued previously that such file-based systems cannot be used when cluster sampling is being employed since the disk is being accessed without reference to any file system.

Roussev et al. [38] suggested that block hashes could be used in forensics for detecting similar files. If two files were each hashed at the block level then the number of hashes common to both could be used as a measure of similarity between the two files. However, they noted that the hash database for block level MD5 hashes of a 512 GB hard disk would require 32 GiB of RAM which was too large to use in a standard PC. The use of a Bloom filter could reduce this by an order of magnitude for the cost of a small false positive rate. Bloom filters were therefore an efficient way to store large sets of hashes. Kornblum [39] developed ssdeep, a system of fuzzy hashing which created a similarity hash of a file that could be used to detect similar files. Roussev [40] developed the application sdhash which improved on the performance of ssdeep by using similarity digests created from statistically improbable features. Roussev et al. [6] used this tool in streaming mode to hash data blocks at a disk read speed and query a reference database. They showed that, using a 48-core server, the maximum size of reference database that could be queried at a read speed of 100MB/s was 15 GB. Similarity digests produced by sdhash are up to 2.6% of the original data size thus the Police Scotland database could just be accommodated but the system requirements are beyond what can be expected in the field. They suggested that similarity digests could only be used in the field with reference databases up to 1 GB which would not be usable in the envisaged scenarios such as the Police Scotland database of child pornography which holds over 6 million images.

Farrell et al. [41] investigated the use of Bloom filters for the distribution of the NSRL hash database. Although they rejected the idea of distributing the database in this format since it is relatively easy to engineer any malicious file to give a false positive if the content of the Bloom filter is known, they found that Bloom filters were useful for providing high-speed matches against hash sets.

Garfinkel et al. [18] looked mainly at file fragment type discrimination but mention that statistical sector sampling could be used for detecting contraband data. EnCase [42] includes a

script – File Block Hash Map Analysis – which block hashes known files and then searches selected areas of the disk. However it states that due to performance issues it is only suitable for a small number of target files. Young et al. [31] came closest to meeting the requirements. They created a database of 1 billion cluster hashes with the intention of deploying the system on a laptop. They used a hybrid approach by using a Bloom filter to screen out negative results so that only queries for hashes that may be in the corpus were passed to their customised hash database. This use of pre-filtering had been suggested by [18], [41] and makes use of the Bloom filters property of very fast lookup for items not in the set. Using a SQL database on a Dual Xeon processor setup, each with 16 cores and 128 GiB RAM they could not perform hash lookups fast enough to keep up with the cluster reads. However they found that using a well-specified laptop with 8GiB of RAM and an SSD, their customised database could perform lookups faster than sectors can be read from the device being triaged.

In summary, Bloom filters can provide compact representation of a hash set and provide high-speed matches against them and that statistical cluster sampling can be used to detect contraband. In the next section these methods are examined further to see if their combination can be used to create a suitable triage system. A quantitative analysis is undertaken of hash set sizes, filter sizes and sample sizes that would be required.

2.3 A Triage System

To create a triage system to meet the requirements listed by Police Scotland, several factors need to be considered. This section gives a background to and justification of some of the ideas, methods, mathematical and statistical tools that will need to be used if fast contraband detection in large capacity storage devices is to be achieved.

It was noted in the introduction that even with well specified equipment, the imaging (forensic copying) of a large capacity storage device can take a considerable time. Even if the device could be analysed in parallel with the imaging so that analysis was completed in the same time, this is well beyond the 'reasonable time' specified in the requirements. It is thus established that if results are to be achieved in a reasonable time then the device can only be sampled since every piece of data cannot be inspected. But how is the device to be sampled? If complete files on a disk are sampled then the file metadata is read from the file system and then the file is accessed. This involves considerable physical head movement in a disk drive.

Fujitsu [43] benchmarked a fast hard disk drive which showed random access throughput at 3 MB/s and sequential access at 200MB/s. Thus, random sampling of files would be considerably slower for triage purposes. In addition, file access relies on the file system metadata and so would not cover unallocated space on the disk where illicit material may well be hidden. Statistically sampling disk clusters overcomes both problems. If the random selection of cluster addresses is sorted numerically before accessing the disk, then this incurs only a single sequential pass over the disk since the disk is treated as simply a sequence of blocks. Additionally, since the sample will be chosen from the whole physical address range of the disk, it would sample all areas including unallocated space, deleted and hidden partitions and deleted files. An additional benefit is that using cluster sampling bypasses the file system by sampling the physical disk and so the nature of the file system on the disk is irrelevant.

However sampling clusters in this way poses some problems. Firstly it needs to be determined if a cluster belongs to a contraband file. To achieve this, methods are developed to determine quickly, to a given accuracy, whether a cluster belongs to a large corpus of contraband files. In contrast to the traditional methodology of hashing complete files, each 4 KiB block of the files in the original contraband corpus is hashed and the hashes stored in a database. A

sampled cluster can then be hashed and its hash compared with those in the database. When saving a file, modern file systems align the start of each 4 kibibyte block of the file on a cluster boundary [31]. Thus by hashing the contraband file in 4 kibibyte blocks, these hashes can be directly compared with the hash of a cluster that has been read from the device.

There are other issues need to be considered in order to achieve a fast initial scan of a device. Firstly computer systems may use a variety of cluster sizes. The optimum cluster size needs to be considered. Secondly, since samples need to be identified at the read speed of storage devices and not be a bottleneck, the structure and speed of the required contraband block hash database needs to be designed accommodate this. The block hash database contains many more hashes than a file hash database and so a compact representation of the database is required if it is to be accessed in RAM for speed of lookup. Thirdly, sample sizes needed to give the required degree of confidence in the results of the triage need to be investigated.

2.3.1 Choice of cluster size

A cluster is the minimum amount of data that can be accessed (either read or written) by the system. Many file systems use a cluster size of 4 KiB by default. The default cluster size in all Microsoft operating systems since NT3.51 is 4 KiB in all disk sizes up to 16 TB [44]. Since 2011, all disk drive manufacturers have standardised on a sector size of 4096 bytes, using an emulation mode (AF 512e) to present 512 byte sector size to a legacy file system where needed [45]. Since most files of interest will be considerably larger than 4 KiB the decreased granularity of working with 4096 byte clusters rather than 512 bytes will be less significant because of the large number of blocks belonging to any one file [18]. Garfinkel [46] also showed that the random sampling of disk drives using 512 byte sectors or 4096 byte clusters gave very similar results. Using 4 KiB cluster size will reduce the hash database size by a factor of 8 compared to that needed to store hashes of 512 byte sectors. For these reasons a cluster size of 4096 bytes is used in the remainder of this research. There is a potential problem in that if the file system with cluster size of 4 KiB is used to write to an old disk with 512 byte sectors then the cluster might not be aligned on an 8-sector boundary and thus not be detected by the system [31]. However obtaining the disk partition alignment information direct from the physical disk Master Boot Record or GUID Partition Table surmounts this problem.

2.3.2 Choice of sample size

Calculating the sample size for detecting contraband on a disk drive is critical for the accuracy of any process depending on it. The sample size depends on the storage capacity of the device, the expected size of the target (i.e. the quantity of contraband on the device) and the probability required for the sample to contain one or more 'hits' i.e. contain one or more samples from the target. Initial contact with digital forensic investigators at Police Scotland has indicated that cases have many files. The most recently reported had over 12 GB of contraband. For the purposes of this research, a target size of 4 MiB is used which is approximately the size of two medium resolution images so that results obtained can be considered as 'worst case'. For comparison the results for a target size of 20 MiB are also given.

When sampling the clusters on the device a sample of k clusters from a population of n clusters is chosen without replacement. No cluster can be selected twice since, once selected, it is not returned to the population for sampling. This is represented by the Hypergeometric distribution [47]. Suppose a disk drive has n clusters. Further suppose that it has t clusters of interest - the target - from some illicit file or files. When k clusters are chosen randomly from the total of n on the disk then the chance of **not** obtaining any target clusters is the same as drawing the sample entirely from the none-target clusters.

The probability of drawing the sample from the non-target clusters is

$$\frac{\text{Number of ways of choosing a sample of } k \text{ clusters from } (n-t) \text{ non} - \text{target clusters}}{\text{Total number of ways of choosing a sample of } k \text{ clusters from } n \text{ clusters}}$$

Using the standard representation for combinations [48], this gives

$$Prob(\ Sample\ has\ no\ target\ clusters) = \frac{\binom{n-t}{k}}{\binom{n}{k}} \quad (1)^1$$

The standard definition for these binomial coefficients is

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!} = \frac{\Gamma(\mathrm{n}+1)}{\Gamma(\mathrm{k}+1)\Gamma(\mathrm{n}-\mathrm{k}+1)} \quad (2)$$

The factorials have been re-written in terms of the Gamma function (Γ) since, when sample sizes of 1 million in populations of hundreds of millions are used, such as the number of 4096 byte clusters in a 1 TB disk drive, then the factorials quickly become large and so logarithms need to be used to avoid numerical overflow. The Gamma function for positive integers is defined as $\Gamma(n) = (n-1)!$ [48] and many numerical applications have a highly accurate log Gamma function. Using the basic laws of logs in equation (2) gives

$$log\binom{n}{k} = log\Gamma(\mathrm{n}+1) - log\Gamma(\mathrm{k}+1) - log\Gamma(\mathrm{n}-\mathrm{k}+1)$$

Therefore

$$log\frac{\binom{n-t}{k}}{\binom{n}{k}} = [log\Gamma(\mathrm{n}-\mathrm{t}+1) - log\Gamma(\mathrm{k}+1) - log\Gamma(\mathrm{n}-\mathrm{t}-\mathrm{k}+1)] - [log\Gamma(\mathrm{n}+1) - log\Gamma(\mathrm{k}+1) - log\Gamma(\mathrm{n}-\mathrm{k}+1)]$$

$$= log\Gamma(\mathrm{n}-\mathrm{t}+1) - log\Gamma(\mathrm{n}-\mathrm{t}-\mathrm{k}+1) + log\Gamma(\mathrm{n}-\mathrm{k}+1) - log\Gamma(\mathrm{n}+1)$$

If this calculated log value is represented by x, then the probability of no hits in the sample is the inverse log of x which is e^x. Hence the probability of 1 or more hits is $1 - e^x$. (3)

Using this formula, Table 1 and Table 2 show the probabilities of obtaining at least one cluster in the sample from a small 4 MiB and a larger 20 MiB target for a variety of sample sizes for common device capacities.

Table 1 - Target 4 MiB. Probability of a sample containing one or more clusters of the target.

Disk Size (GB) \ Sample Size	100000	200000	300000	400000	500000	600000	700000	1000000
120	0.96	1.00	1.00	1.00	1.00	1.00	1.00	1.00
250	0.79	0.96	0.99	1.00	1.00	1.00	1.00	1.00
320	0.71	0.91	0.97	0.99	1.00	1.00	1.00	1.00
500	0.54	0.79	0.90	0.96	0.98	0.99	1.00	1.00
1000	0.32	0.54	0.69	0.79	0.86	0.90	0.94	0.98

[1] The formula p =1 - $\prod_{i=1}^{n} \frac{((N-(i-1))-m)}{(N-(i-1))}$ can also be used [31] but does not lead to such an easily computable result.

Table 2 - Target 20 MiB. Probability of a sample containing one or more clusters of the target.

Disk Size (GB)	Sample Size 50000	100000	200000	300000
120	1.000	1.00	1.00	1.00
250	0.980	1.00	1.00	1.00
320	0.953	1.00	1.00	1.00
500	0.858	0.98	1.00	1.00
1000	0.623	0.86	0.98	1.00

Although these sample sizes are calculated to ensure a 99% probability that the sample will contain at least one cluster of the contraband, the expected number of hits in any sample is higher. In statistical theory the Central Limit Theorem states that the distribution of sample means is approximately normally distributed. A sample size of 30 or more will result in a sampling distribution for the mean that is very close to a normal distribution [49]. Using this, confidence limits on the expected number of hits in a sample can be calculated. To calculate the confidence intervals the mean and standard deviation of the data are required. The expected value (mean) and variance of the Hypergeometric Distribution are given by [47]:

$$E(x) = k \times \frac{t}{n} \quad and \quad Var(x) = k \times t \times (n-t) \times \frac{(n-k)}{n^2 \times (n-1)} \qquad (4)$$

where k is the sample size, t is the target size and n is the population (disc) size. The standard deviation is simply the square root of the variance.

Using the equations in (4) the expected number of hits given the sample size calculated to give a 99% probability of one or more hits is shown in Table 3 for the 4 MiB case.

The values for other disk and target sizes will be similar since the sample size is always calculated to give a 99% probability of one or more hits in the sample. Thus as the disk size increases, the sample size increases correspondingly. As the target size increases, the sample size decreases correspondingly. Small differences in the expected number of hits are due to the 'goal seeking' method of calculating the sample size. The process may converge on a probability of 99% from above or below so that the process may terminate with small differences in calculated sample size. It can be seen that although the sample size is calculated to give a 99% confidence of one or more hits in the sample, the expected number of hits in the sample is higher.

Table 3 - Expected number of hits for 4 MiB target

Disk Size (GB)	Sample Size	Expected number of hits	99% Confidence Interval
250	278,379	4.67	± 0.01
320	347,310	4.55	± 0.009
500	539,329	4.52	± 0.007
1000	1,089,364	4.47	± 0.005

2.3.3 Database structure and lookup

Block hashing the contraband file corpus leads to a much larger database than the traditional file hashing technique. Taking the size of a typical medium resolution JPEG image to be

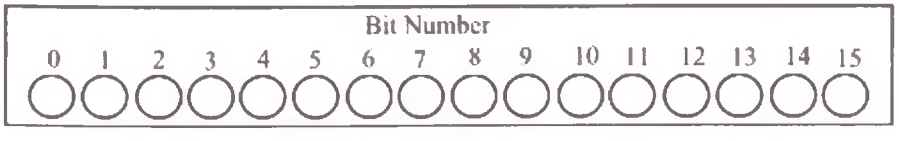
1MiB then this contains 256 × 4 KiB clusters. Thus on block hashing, the hash database will be 256 times larger than the equivalent file hash database which only stores one hash for each file. In addition, if clusters are to be identified and classified at a speed which matches the read speed of a storage device then the use of a disk-based database is not appropriate because of the slow access speed of these databases. An in-memory solution for hash lookup must be devised. However, as an example, the Scottish Police database of child pornography holds 5.1 million images in Category 1 (the most serious) and a further one million in categories 2 to 5 with an average file size of 100 KiB. Each of these six million images will contain on average 25 × 4 KiB disk clusters.

A database of these cluster hashes would therefore contain 150 million hashes and so a database size of 200 million hashes will be needed to allow for expansion. Even if a weak hash algorithm such as MD5 which produces a 16-byte hash were used to hash the clusters, a database of 200 million hashes each 16 bytes long is needed. This would require over 3 GiB of main memory which is not available on legacy systems which may well have a 32-bit operating system or limited RAM. This is too large a database to hold in internal memory on such a system. If the database were disk based then hash lookups would be far slower than the sequential read being used for disk sampling and this would prove to be a severe bottleneck. It was noted in section 2.2.2 that one solution to provide a space efficient means of testing whether or not an element is a member of a set, at the controllable risk of false positives, is to use a Bloom filter [50]. This gives compactness and speed, at the cost of a small controllable false positive rate.

2.3.4 Bloom filters

A Bloom filter consists of a set of hash functions and a bit array of a fixed length. All bits in the bit array are initialised to 0. A very simple example is used to illustrate the principle which will ease the understanding of the theory.

This bit array holds 16 bits:

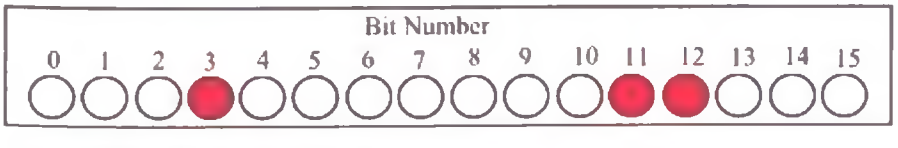


A number of independent hash functions are needed, each of which takes input of arbitrary length and produces a hash value between 0 and 15, the address range of the bit array. Let's call them $h_1()$, $h_2()$ and $h_3()$.

If the string 'Jupiter' is to be added to the filter, 'Jupiter' is hashed with each of the hash functions. Suppose they generate:

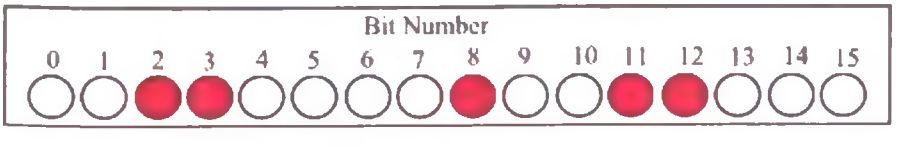
$h_1(\text{'Jupiter'}) = 3$ $h_2(\text{'Jupiter'}) = 12$ $h_3(\text{'Jupiter'}) = 11$

The corresponding bits 3, 12 and 11 in the bit array are set:



Now adding 'Venus' in the same manner gives us:

$h_1(\text{'Venus'}) = 11$ $h_2(\text{'Venus'}) = 2$ $h_3(\text{'Venus'}) = 8$



Notice that bit 11 was already turned on. It is now storing information about both 'Jupiter' and 'Venus'. As more items are added to the filter it may store information about some of those as well. It is this overlap which gives Bloom filters their compactness. Any one bit may be storing information about multiple items simultaneously.

To check if an item already exists in the filter the same process of hashing the item is followed and the bits at all those positions are checked. If they are all set then the item is probably in the filter. It is only 'probably' in the filter because there is the possibility of a false positive. Suppose 'Neptune' is tested for its presence in the filter:

h_1('Neptune') = 11 h_2('Neptune') = 3 h_3('Neptune') = 8

All these bits are set so the filter reports that 'Neptune' belongs to the set.

Thus a false positive is possible in a Bloom filter. However a false negative is not. If any bit is 0 for an examined element, then that element cannot possibly be in the set otherwise its calculated hash position would have been set to 1. This illustrates why searching elements not in the set is considerably faster. The search can be terminated as soon as an unset bit is encountered.

In the example above there are a number of variables:

- m = the number of bits in the array which represents the Bloom filter. Initially all bits are set to 0
- n = the number of elements added to the filter
- k = the number of independent hash functions h_1, h_2, ... , h_k used.

These variables are important building blocks for the theory. Notice that the Bloom filter had $m = 2^4$ bits and the hashes needed to generate a hash with a bit length of 4 bits. In general a Bloom filter of size 2^p needs a hash of length p bits.

Each hash function generates a hash value in the range $(0, \ldots\ldots, m-1)$ for any input. To insert an element e_i from the set $E = \{e_1, e_2, \ldots\ldots\ldots, e_n\}$ of elements to be added to the filter, the hash values $h_1(e_i)$, $h_2(e_i)$, ... , $h_k(e_i)$ are calculated and the corresponding locations in the bit array are set to 1. This is repeated for each element to be added to the filter. To check if an element belongs to the set, its k hashes are calculated. If all these bits are set to one then the item may be in the set. If not all bits are 1 then the element is definitely not in the set.

2.4 Design

In the previous section tools and methodologies have been introduced that will be used in the design of the triage system. Cluster sampling will provide fast lookup of the entire device address space thus inspecting deleted files and deleted or hidden partitions. Sampling can be controlled to ensure a high probability of detecting even small amounts of contraband. A Bloom filter can be used for compact in-memory representation of the contraband hash database and will provide fast lookup with a controllable false positive rate. In this section a Bloom filter for a typical scenario of a large database of contraband files and capable of meeting the required false positive rates for the sample size being used is designed.

2.4.1 Designing the filter

Initial contact with forensic investigators at Police Scotland has indicated that cases have many files. The most recently reported case in July 2015 had over 46,000 Category 1 – 5 images with an average size of 270KB. On disk this gives a target size of approximately 12 GB. If this is taken to be excessive and for the moment assume that an average case would involve a target size of 500 MB, then on a 1 TB disk a sample size of only 13,300 would be needed to give a probability of 99.9% of hitting the target. The sample sizes required for a

variety of target and disk sizes are shown in Table 4. A minimum target size of 4 MiB was chosen in conjunction with forensic analysts at Police Scotland since this is the size of two standard JPEG images and contraband of any smaller size was regarded as insignificant and would lead to increased sample sizes, and hence triage run time, for little advantage.

Table 4 - Sample size required for 99% probability of hitting target for a variety of target and disk sizes

	Target Size (MiB)							
Size(GB)	4	20	50	100	200	300	400	500
250	278500	55104	22339	11000	5468	3619	2696	2152
320	347310	71820	27950	14102	6937	4603	3459	2753
500	539329	112034	43683	22035	10854	7239	5469	4356
1000	1089364	220066	89305	44056	21559	14483	10853	8745

These sample sizes can be used to choose design parameters for the Bloom filter.

Mitzenmacher and Vadhan [51] show that the false positive rate for a Bloom filter in terms of the values m, n and k is

$$P(\boldsymbol{false\ positive}) \approx \left(1 - e^{-\frac{kn}{m}}\right)^k \tag{5}$$

And that k is optimal when $k = \frac{m \log(2)}{n}$

Using Equation (5), the Bloom filter can be engineered for the application. It was calculated that the envisaged database would contain 150 million 4 KiB blocks and so if n is taken to be 200,000,000 then growth in the database is allowed for. Table 5 shows the probability of a false positive for a variety of combinations of the number of hashes k and the Bloom filter size m with the values generated using equation (5).

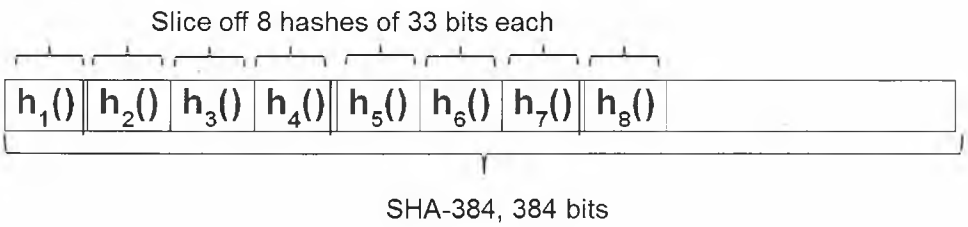
Table 5 – Probability of a false positive for varying values of m and k

	m = Bloom Filter Size (MiB)					
k	512	600	700	800	900	1024
4	0.000834149	0.000466407	0.000263157	0.000159487	0.000102189	0.000062537
6	0.000209786	0.000091112	0.000039869	0.000019271	0.000010073	0.000004912
8	0.000087538	0.000030242	0.000010473	0.000004100	0.000001769	0.000000696
10	0.000051133	0.000014373	0.000004016	0.000001292	0.000000466	0.000000149
12	0.000037893	0.000008855	0.000002034	0.000000546	0.000000166	0.000000044
14	0.000033433	0.000006629	0.000001274	0.000000289	0.000000075	0.000000017
16	0.000033607	0.000005763	0.000000942	0.000000183	0.000000041	0.000000008

Ideally there should be no false positives in the sample. In a worst case scenario, if a 1 TB drive has a small 4MiB target, then a sample size of over one million is needed as shown in Table 4. It would be preferable to have no false positives in a sample due to the Bloom filter and so generating a filter with a false positive rate of less than one in a million would be preferable. From Table 5 it can be seen that to meet this requirement the Bloom filter is going to be approximately 800 MiB in size and use 10 hashes, or 1 GiB in size, and use eight hashes for a probability of a false positive to be less than one in a million. If the device being examined has a 1 TB drive then it is very likely to have 2 GiB RAM or more and a 1 GiB array would be acceptable. The larger Bloom filter size is a trade-off against the number of hashes. This is important since the speed of the algorithm is inversely proportional to the number of hashes. The more hashes that need calculated the slower the process.

It was noted earlier that a bloom filter of size 2^b bits needed a hash of length b bits. The 1 GiB array has 2^{33} addressable bits and so a hash of length 33 bits is needed. In order to do this, eight independent hashes generating a 33 bit hash which each produce a uniform distribution over the range would have to be developed. Developing a suitable hash function is a non-trivial exercise [51] and so the development of eight such hashes would be challenging. However a solution exists in that these hashes can be generated efficiently by using SHA-384 as the hash algorithm. This generates a 384-bit hash. The first 264 bits of the hash will be used as 8 x 33 bit hashes. These hashes meet the requirement for independence since the SHA384 hash can be assumed to be collision resistant and therefore any single bit or subset of bits may be taken to be independent uniformly distributed random variables [38].

Figure 1 - Hashes created from single SHA-384 calculation



Since eight hashes are being used, each of which is used to set one bit in the Bloom filter, each entry in the filter requires a maximum of 8 bits or the equivalent of one byte (as noted before, some of the bits might already be set). So adding 200 million bytes created from the Police Scotland cluster hash database to the filter leaves a 1 GiB Bloom filter lightly loaded. The filter is loaded directly into main memory and treated as a 1GiB array.

Having designed the filter to cope with the worst case scenario of a small 4 MiB target on a large capacity device then, given the target sizes envisaged by Police Scotland, there will be many hits. This is shown in Table 6, where the expected number of hits is calculated using the expected mean value of the number of hits from equation (4).

If time is a critical factor and the target size is expected to be large then sample sizes could be reduced to those given in Table 4. In addition the triage can be stopped when it is obvious that the device contains contraband, for example after 10 hits.

Table 6 - Number of hits when sample size is calculated for a 4 MiB target

Actual Target Size (MiB)	20	50	100	200	300	400	500
Expected Number Of Hits	23	58	116	233	350	467	583

2.5 Implementation

Software to implement a Bloom filter, block hash any collection of files and add the hashes to the filter was created. A filter was generated with a large amount of random data to simulate a block hash database for a large contraband corpus. To this was added the block hashes of 100 MiB of real images. A selection of these real images could be added to a test device which could then be scanned to see if the contraband was detected.

2.5.1 Bloom Filter Creation

Sufficient data from random.org [52] was used to generate entries for 200 million 4 KiB blocks in the Bloom filter. The Bloom filter was tested first by checking the hashes of one million of the block hashes originally included in the filter and all were reported to be present. Then 1 million fresh block hashes were created with further data from random.org and tested for inclusion in the filter. There was one positive match which agreed with theory since the

calculated false positive rate for the Bloom filter is 0.000 000 696 or one in 1,437,184. The false positive could be caused by one of the generated random blocks matching one previously used. However whether the positive was caused by a matching random block or whether it was a true false positive caused by the filter, the filter is operating as designed. Finally the block hashes for 100 MB of real images were added to the filter. These same block hashes were then each tested against the filter and all were reported present and so the Bloom filter was working as it should.

2.6 Testing

Testing was done on two different computers - a desktop PC with a Core i3 processor and 4 GB RAM, HDD, SSD and external USB HDD drives and a netbook with an Intel Atom CPU, 2 GB RAM, a 250 GB SSD and a USB attached HDD. The tests were carried out by booting the computers into a Windows forensic environment where the drives to be tested were mounted as read-only [53]. Software was written to access any of the physical storage devices directly. The address range of the device, from first to last cluster, was determined, the sample size calculated and that number of cluster addresses was randomly chosen from the range. This random selection was then sorted numerically so that when the device is accessed in that order any read head movement is minimised because of the sequential nature of the reads.

A selection of 4 MiB of the real images was added to each of the disk drives being tested. These images were variously deleted or added to a second partition and the partition deleted. The tests were run multiple times – three times with the files saved on the device as normal, three times with the files deleted and three times with the files saved on a second partition and the partition deleted. Between each type of test, the contraband was securely deleted by overwriting with random data so that remnants of previous tests did not influence results. The sampling, hashing of samples and Bloom filter lookup was done and the timings and results recorded. 20 MiB of contraband was then added to each disk and the tests run as before to give results for the 20 MiB test.

The Microsoft NTFS File Sector Information Utility [54] was used to look up each hit to ascertain if it was a true hit or a false positive. This utility allows the user to enter the drive and cluster number and reports from the MFT the full path of the file, if any, which the sector belongs to. All hits during testing were verified to come from files within the contraband corpus. If a file was deleted or on a hidden partition then it was carved using the cluster address – the disk is scanned forward and backwards from the cluster address for the file header and footer and then re-assembled.

Tests were run three times in each configuration listed and the averages are shown in Table 7 and

Table 8. No false positives were encountered. The methodology can be considered 'fit for purpose' since these results would all have been positive for the possession of contraband data. The tests showed that the method will detect deleted contraband and contraband on deleted partitions. All timings are within the *reasonable time* specified.

It is clear that since the SSDs on the PC and Netbook were similar, the performance of the Intel Atom processor is the bottleneck rather than the SSD. This effect is less marked with the slower I/O of the HDD.

Table 7 – Core i3 Desktop PC sampling accuracy and timings

Disk and Size	Target Size	Samples	Hits	False Positives	Time min:sec
250 GB SSD	4 MiB	281000	5	0	00:13
250 GB SSD	20 MiB	53000	4	0	00:02
250 GB USB HDD	4 MiB	274000	6	0	28:58
250 GB USB HDD	20 MiB	55000	4	0	06:06
1 TB HDD	4 MiB	1096000	4	0	79:13
1 TB HDD	20 MiB	220000	4	0	19:30

Table 8 - Intel Atom Netbook sampling accuracy and timings

Disk and Size	Target Size	Samples	Hits	False Positives	Time min:sec
250 GB SSD	4 MB	263000	4	0	04:35
250 GB SSD	20 MB	53000	5	0	00:53
250 GB USB HDD	4 MB	274000	5	0	44:34
250 GB USB HDD	20 MB	79700	7	0	11:04

2.7 Conclusions

It has been shown that a fast initial scan of a device, possibly conducted under severe time and resource constraints, is feasible. The prototype software that has been created to test the methods is in two parts: a contraband block hash database builder; and a contraband detector. Both have been tested within Police Scotland.

A block hash database of contraband block hashes from their database of images of child pornography was created and tested on live cases. It met all their requirements for speed and accuracy. It was also noted that this methodology allows the Bloom filter of contraband block hashes to be distributed since the block hashes cannot be 'reverse engineered' from the Bloom filter that is created. The block hash database thus remains confidential. The speed at which a decision can now be made as to whether a device needs further investigation or not could ease the backlog of devices requiring an investigation. This also helps deliver an ethically more acceptable system since, while an investigation is backlogged, a wrongful accusation can result in harm to social and professional relationships of a suspect or a guilty party, possibly a danger to society, could be left at large.

A theoretical justification for the approaches to sampling that were needed to deliver the required accuracy and the use of Bloom filters to deliver the required speed and small memory footprint have been presented and the results tested by experiment.

3 Non-probative Blocks

3.1 Introduction

In the previous chapter it was shown that a system of triage implemented by sampling large capacity digital storage devices is feasible. However, Garfinkel (Simson Garfinkel, personal communication, 10 March 2015) suggested that since testing of the triage system had been conducted using random data, that it might not be applicable when it was applied to real world data. The problem arises from the existence of 4 KiB blocks (clusters) that may occur in many files such as data structures within office documents and multimedia files. These common blocks may also occur in the contraband database, and so cannot be used to identify or prove the existence of a particular file on the media. This problem was formalised later by Garfinkel and McCarrin [24]. Since this could be considered a stumbling block for the whole process of triage the situation needs consideration. In this chapter the scale of the problem is quantified by hashing several multi-million block corpora to observe the rate of common blocks. Methods have been proposed to detect common blocks, and these methods need to be examined to see if the problem can be mitigated by detecting these common blocks while the contraband database is being built and eliminate them at that stage from the contraband block hash database. Finally, in a triage situation where only a decision on whether a device contains contraband or not is required, it is shown that there is a choice of methods to mitigate the effect of the observed rate of common blocks.

3.2 Terminologies

The term *block* will be used to refer to a cluster so that the terminology relates to previous work in this area. In the previous chapter data from random.org [52] was used to create a corpus for testing. Garfinkel and McCarrin [24] show that, unlike random data where a duplicate hash is unlikely, there is a possibility of common blocks appearing in many files in a real life corpus. Blocks such as all null, whitespace or header blocks may be very common and appear in many thousands of different files. Since these blocks appear in many different files, any such block found on a suspect device cannot be used to prove that a particular file exists or existed on the device being examined and so they referred to these as 'non-probative' blocks.

During triage of large capacity devices, blocks from the device are sampled and checked against a database of block hashes created from a contraband corpus. If the database of contraband block hashes contains any non-probative blocks then there is a possibility that a device being investigated could be flagged as containing contraband when in fact it may not. Being able to detect non-probative blocks so that they could be excluded during the building of the contraband block hash database would solve this problem. In the context of the Police Scotland and the UK Child Abuse Image Database the effectiveness of ad-hoc tests for detecting non-probative blocks with particular reference to JPEG format needs to be investigated. The effect that being unable to eliminate such non-probative blocks from the contraband database will have on the results of triage needs to be considered and a mitigation strategy devised.

3.3 Prior Work

As noted by Young et al. [31] little work has been done on the use of block hashes for file identification despite it having numerous advantages over file hashing. A major advantage is that it can be used together with random sampling to detect the presence of selected data on a large-capacity device in minutes rather than hours.

Garfinkel et al. [18] characterised a distinct block as one that will not arise by chance more than once. They outlined that an instance of a distinct block found on a device is evidence that the original file containing that block was once present on that device and that if a block is shown to be distinct in a large and representative corpus then the block can be treated as distinct. Their analysis of JPEG files showed that in a real corpus many JPEGs contain common elements such as colour tables, XML elements and EXIF information and so common blocks would exist in live corpora. They presented techniques for classifying the contents of a drive by random sampling and for carving data based on sector hashes.

Foster [55] analysed the occurrence of distinct file blocks in three multi-million file corpora. Most files consisted of distinct blocks but there were certain blocks that occurred with high frequency. Although it was not possible to prove that a block is universally distinct, treating a block that only occurs once in a large corpus as if it were universally distinct made it possible to quickly find evidence that a file was present on disk. It was found that meaningful precision was not lost by using 4 KiB blocks rather than 512 B. In the Govdocs1 corpus [56] the most frequently occurring non-probative blocks found were all null and all ones. Other high-frequency blocks were data structures related to particular file types. Using 4 KiB block size it was found that 99.46% of blocks were singletons i.e. distinct. It was noted that a high percentage of the non-distinct blocks were from ASCII text files and contained a few near identical files.

Young et al. [31] used the same corpora as Foster [55] and reported a similar total of 0.54% non-probative blocks in the Govdocs1 corpus using a block size of 4 KiB. Of the common blocks 0.44% occurred only twice in the corpus (pairs) and 0.11% appeared more than twice (common). Database structures to accommodate a database of one billion block hashes that could be deployed on a laptop to detect files in large disk images were investigated and showed that database lookups could be performed faster than blocks could be read from a drive being triaged.

Garfinkel and McCarrin [24] found that many blocks that appeared distinct in one corpus were not distinct in larger corpora and they changed the terminology for blocks that have a high probability that the containing file was once present from *distinct block* to *probative block*. Three rules were developed to help with the decision as to whether a block hash database match was probative or not. A database match could be suppressed if it was flagged to match one or more of the three rules.

- **The Ramp Test.** This was designed to detect the Microsoft Office sector allocation table, which appear in the Microsoft compound document file format which is a simple pattern where alternate bytes form a monotonic increasing sequence.
- **The White Space test**. This excludes blocks where three-quarters or more of the block was white space.
- **The 4-byte Histogram test.** This was designed to detect common Apple Quick Time and Microsoft Office file formats. The block is treated as a sequence of 4-byte values and if any single value occurred more than 256 times, the block is flagged as non-probative.

In addition an alternative test was implemented by treating the block as a collection of 4-byte unsigned integers and calculating the 16-bit Shannon entropy for the block. These tests were

considered to work well for the corpora used. However, using Shannon entropy as an identifier for non-probative blocks was found to flag over 90% of the blocks in many files and so the other tests were preferred. The ad-hoc tests were run only against a set of blocks already identified as being non-probative. The question of how best to classify probative blocks in the general case was left for future work.

3.4 Testing For Non-Probative Blocks

To establish a base reference for the frequency of non-probative blocks and to create a working corpus for testing, the publically available MirFlikr corpus of one million JPEG images was downloaded from the LIACS Medialab at Leiden University [57]. A database containing both file hashes and individual 4 KiB block hashes for each file in the corpus was created. The final block in any file was padded with zeros if it did not fill the last 4 KiB completely. After removing duplicate files the database contained a total of 31,469,943 hashes of 4 KiB blocks from 999,622 JPEG files.

By examining the block hashes it was found that there were 3371 blocks which appeared more than once and these occurred a total of 76,294 times in the corpus. These 76,294 occurrences amount to 0.2% of all JPEG blocks i.e. if a block is chosen randomly from the corpus then there is a probability of 0.002 of it being a non-probative block within the corpus.

Garfinkel and McCarrin [24] devised some ad-hoc tests to detect non-probative blocks and eliminate spurious matches from their analysis. Although these tests were not designed with the JPEG file format in mind and were used on blocks already identified as target hits, their effectiveness was evaluated on the corpus since any efficacy in removing non-probative blocks during the building of a contraband database would be of use. These tests were applied to the JPEG corpus of over 30 million blocks but it was found that they did not work well as a discriminator for non-probative blocks in this situation. This was expected since the tests were not designed for this purpose. Each of the four tests was applied to each 4 KiB block in the corpus. The tests each flagged a block as either probative or non-probative. Of the 76,294 non-probative blocks, the tests detected some positive hits (i.e. correctly identified as non-probative) but the majority were not identified (i.e. false negatives). Of the probative blocks there were a large number identified as non-probative (false positives). The results are shown in Table 9.

Table 9- Ad-hoc test results – Total fragments flagged as non-probative by each test

	Four-Byte Histogram	Ramp	Whitespace	Entropy
Positive	6,171	12,557	965	6,503
False Negative	70,123	63,737	75,329	69,791
False Positive	754,604	348,371	254,178	600,862

The proportion of actual non-probative blocks flagged correctly was small compared to the number of blocks flagged incorrectly. It can be seen that the number of false positives and false negatives is so large as to make these tests unusable for filtering out non-probative blocks from the JPEG corpus. It is obvious that these tests cannot be of use as a discriminator between probative and non-probative blocks in the JPEG corpus. They would introduce far more error into the block hash database than they would remove. For example the entropy test flags over 600,000 blocks as non-probative that are actually probative and so if this test were used during the building of the database these 600,000 blocks would not be added to the contraband database. This would lead to many contraband files not being detected.

Many fragments were flagged by multiple tests, such as where Whitespace always occurred when the entropy was also flagged, which is to be expected since a block consisting largely of whitespace will have low entropy.

3.5 An alternative test for JPEG non-probative blocks

When the high-frequency non-probative blocks from the JPEG corpus were examined it was noticed that over 50% of non-probative blocks displayed a similar pattern. These were found to be ICC Profiles [58], and define a colour-space transformation system to ensure that colours are displayed as intended. These are added by applications as the file is edited or converted and are added by default by packages such as Adobe Photoshop. Alternate bytes formed a non-decreasing sequence and for each value in this sequence the intervening bytes formed a non-decreasing sub-sequence. Looking at the first sequence in Table 10 it can be seen that starting with the first byte hex value 2A, every second byte forms a non-decreasing sequence starting at 2A and eventually changing to 2B. The intermediate bytes form a non-decreasing sequence for each of these principal values. The second sequence shown illustrates the same behaviour but offset by one byte with the principal sequence commencing at the second byte. Since the 4 KiB blocks were split into 128 byte groups, these sequences will be detected even if they did not commence at the 4 KiB block boundary. Some of these patterns were offset by one byte as shown in Table 10.

Table 10 - Typical ICC sequence and offset sequence

Sequence	2A 4A 2A 57 2A 63 2A 70 2A 7D 2A 8A 2A 97 2A A3 2A B0 2A BD 2A CA 2A D7 2A E3 2A F0 2A FD 2B 0A 2B 17 2B 24 2B 31 2B 3D 2B 4A 2B 57 2B 64 2B 71 2B 7E 2B 8B 2B 98 2B A5 2B B2 2B BF 2B CC 2B D9	A non-decreasing sequence with intervening non-decreasing sequences.
Sequence offset by one byte	D7 38 14 38 50 38 8C 38 C8 39 05 39 42 39 7F 39 BC 39 F9 3A 36 3A 74 3A B2 3A EF 3B 2D 3B 6B 3B AA 3B E8 3C 27 3C 65 3C A4 3C E3 3D 22 3D 61 3D A1 3D E0 3E 20 3E 60 3E A0 3E E0 3F 21 3F 61 3F	A non-decreasing sequence with intervening non-decreasing sequences.

An ad hoc test to detect these patterns was developed. Each 4096-byte block of the corpus was split into 32 groups of 128 bytes. Each group was tested for an ICC pattern. This has the added benefit of flagging any block of constant values since such a block is non-decreasing by definition.

Thresholds of 25%, 50%, 75% and 100% of positive 128-byte groups within a block were applied to determine which was best suited. As can be seen in Table 11, this test flagged a larger number of the non-probative JPEG blocks than the ad hoc tests, but only a very small proportion of blocks flagged by the test were actually non-probative blocks.

As with the previously investigated ad-hoc tests the ICC feature is too common among all blocks of the JPEG corpus to be useful as a discriminating feature for the detection of non-probative blocks.

Table 11- ICC test results

	Threshold			
	25%	50%	75%	100%
Positive	48,246	46,938	31,634	16,212
False Negative	28,048	29,356	44,660	60,082
False Positive	1,117,534	530,190	276,925	14,096

3.6 Global and Local Uniqueness

To test the uniqueness of the blocks within the MirFlikr JPEG corpus, the JPEG files from the Govdocs1 reference data set [56] were used. After duplicate files and files under 4 KiB in size were removed the Govdocs1 JPEG data set contained 91,937 jpg files containing 8,576,061 4KiB blocks. The file hash of each file was checked and there were no files common to both corpora.

There were 26 blocks from the 31,393,649 unique (probative) blocks in the MirFlikr JPEG corpus which also appeared in the Govdocs JPEG corpus. Each appeared only once. This shows, that although these blocks were unique within the large JPEG corpus, they are not globally unique.

In the Govdocs1 JPEG corpus there was a smaller proportion of non-probative blocks. 2252 non-probative blocks occurred 11,834 times giving 0.1% of the corpus being non-probative. Only 122 of the 3371 non-probative blocks in the MirFlikr JPEG corpus also appeared in the Govdocs1 JPEG corpus. 114 of these were also non-probative in the GovDocs corpus. These non-probative blocks common to both corpora occurred a total of 2,468 times in the MirFlikr corpus and 3,671 times in the GovDocs corpus. As more corpora are added, then these blocks common to more than one corpus could form a basis for a blacklist for exclusion when building a contraband database.

3.7 Using Entropy Measures with Small Block Sizes

In information theory a number of measures of information entropy exist. The Renyi family of entropies includes the Hartley, Shannon, collision and min entropy measures [59]. However in the literature reviewed for this thesis only the Shannon 8 bit and 16 bit entropy measures were used for file fragment classification. While running the tests over a large corpus of over 85 million blocks (the one million JPEG files together with all files in the Govdocs1 corpus over 4 KiB in size) it was noticed that the maximum 16-bit Shannon entropy achieved was 11. This is to be expected since a 4096-byte block can contain only 2048 ($= 2^{11}$) different 16-bit symbols and so gives a maximum entropy value of 11 when all 2048 symbols are different.:

$$-\sum_{i=1}^{2048} P(x_i)log_2\left(P(x_i)\right) = -\sum_{i=1}^{2048} \frac{1}{2048} log_2\left(\frac{1}{2048}\right) = -2048 * \left\lfloor \frac{1}{2048} * (-11) \right\rfloor = 11$$

It is of interest when using small block sizes that the maximum entropy of m-bit symbols will be n for a block size 2^n where $n<m$, and for values of m or larger will approach the maximum entropy asymptotically as n increases. This is due to the second condition required in the derivation of Shannon entropy:

> "If all the p_i are equal, $p_i = 1/n$, then H should be a monotonic increasing function of n." [60]

This is illustrated in Figure 2 and Figure 3, where results for 16-bit and 8-bit entropy are charted for blocks of various sizes generated from high entropy data obtained from Random.org [52]. The maximum entropy observed increases as the block size increases. Hence when using or interpreting Shannon entropy care should be taken if using an absolute rather than a relative comparison, especially when block sizes are small.

Figure 2- Maximum observed 16-bit Shannon Entropy

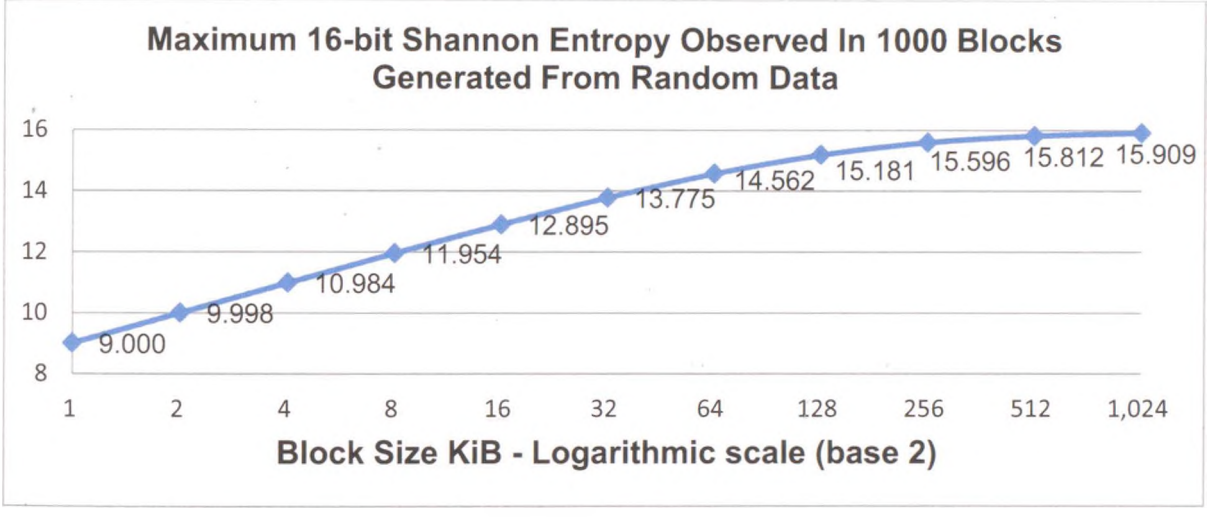
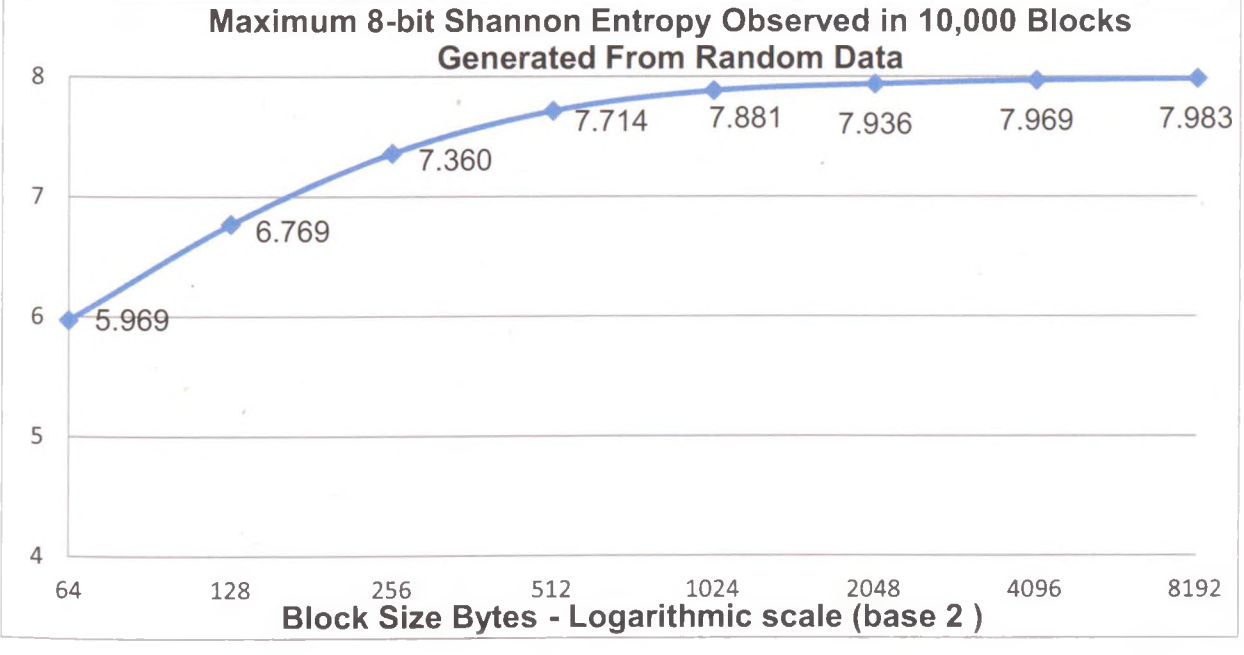


Figure 3 - Maximum observed 8-bit Shannon Entropy



3.8 Effects of Non-Probative Blocks During Triage

In a triage situation, it is required to determine whether a device is likely to contain contraband material or not. During the triage of large capacity devices, samples of more than one million blocks are normal for large capacity devices. These samples are compared against a block hash database of known contraband and a match is designated a 'hit'. Bloom filters are commonly used as a space efficient method of storing the block hash database and these are designed to keep the false positive rate below one in a million [20]. To evaluate the effect of non-probative blocks on this process, the rate of occurrence of non-probative blocks in large corpora was investigated.

As has been seen the MirFlikr JPEG corpus of 4 KiB block hashes, 0.2% were non-probative. The corresponding value was 0.1% in the Govdocs1 JPEG corpus. Published figures give rates of the same order of magnitude - Young et al. [31] reported 0.54% in the Govdocs1 corpus for a 4 KiB block size although Garfinkel and McCarrin [24] report 2% non-probative blocks in the same corpus but in this instance there is no mention of duplicate files having been removed first.

As a sanity check on non-probative rates in large corpora, two disk drives were block hashed and tested for non-probative blocks - a 256 GB disk drive containing 190 GB of mixed content including office and multimedia files and several small disk images, and a 1 TB disk containing 756 GB of mainly multi-media, office documents and compressed backups. The 256 GB drive was found to have 0.4% non-probative blocks and the 1 TB drive 0.1%. Given the observed non-probative rates, an assumed global non-probative rate of 1% is reasonable.

Using this value the probability of any sampled block being non-probative is 0.01. Let this value be denoted by P_{np}.

Using this value the probability of any sampled block being non-probative is 0.01. Let this value be denoted by P_{np}.

The probability of a sampled block being a false positive due to the designed false positive rate of the Bloom filter developed in section 2.4.1is 0.000001. Let this probability be denoted by P_{bf}.

If the overall false positive probability P_{fp} is defined as the probability that a sampled block is either a false positive due to the designed false positive rate P_{bf} of the Bloom filter or that a block is a false positive due it being non-probative with a probability P_{np} then the probability of any individual sample being a false positive is :

$P_{fp} = P_{bf} + P_{np} = 0.010001$

The probability of three samples all being false positives is therefore $(0.010001)^3 \approx 0.000001$ or one in a million. Thus if three hits is accepted as the threshold for a positive triage result the false positive rate of the sampling methodology is within the required accuracy of one in a million false positives

If the non-probative rate was as high as 5% then accepting five hits as a threshold for a 'triage positive' is enough to mitigate the effect of the high non-probative rate. In live tests it has been found that devices being scanned contain either no contraband clusters in the sample or many, and thus it is feasible to mitigate the effect of non-probative blocks in triage simply by setting the threshold for a positive result at a small number of hits.

Although all live tests done with a law enforcement agency show that a drive containing contraband will have many hits from sampling and a threshold of three is of little consequence, the effect of non-probative blocks in marginal cases where the number of hits is low needs to be considered. In the law enforcement scenario it was required to engineer the sampling so that there was a 99% certainty of a hit for a given target size. If this condition is reformulated to be 99% certain of a positive hit – a hit that is neither non-probative nor a Bloom filter false positive - then the sample size required needs to be considered. Using the hypergeometric distribution to calculate sample sizes to be 99% certain of obtaining *n* or more hits in the sample gives the results shown in Table 12 and Table 13 for illustrative disk and target sizes.

It can be seen that if it is required to have a 99% assurance of three or more hits in the sample to allow for the possible false positive rate generated by the non-probative blocks, then an

average of just over 50% increase in the sample size is required. This would entail a 50% increase in the time needed for triage.

Table 12 Sample size required for 99% probability of n or more hits - 256 GB Drive

	Target Size (MiB)		
n	4	8	16
1	403,074	204,473	102,593
2	509,127	258,953	128,358
3	607,040	308,160	154,048
4	700,013	352,643	176,474
5	789,563	396,944	198,535

Table 13 Sample size required for 99% probability of n or more hits - 512 GB Drive

	Target Size (MiB)		
n	4	8	16
1	803,735	408,725	205,163
2	1,015,092	517,328	256,697
3	1,210,301	616,409	308,096
4	1,395,510	705,348	352,951
5	1,591,040	793,013	397,136

However since the number of blocks involved is low, these marginal cases could be dealt with more quickly by examination of the files containing the blocks returned as hits by the sampling process rather than increasing the sample size. If only three clusters were reported then examining the files containing the detected clusters takes only a matter of moments.

3.9 Conclusions

Some simple methods of identifying non-probative blocks have been tested with the objective of preventing such blocks being added to a block hash database of contraband files. It has been shown that in a large corpus of JPEG files, currently available tests are not suited for this purpose. The many false negatives and false positives that they create amplify the problem.

It has also been shown that, in a triage situation, it is feasible to set the threshold for classifying a device for further examination at a small number of identified contraband blocks and the required accuracy is maintained. It has been found empirically that a non-probative rate of 1% of blocks is reasonable and that a threshold of three 'hits' during triage is sufficient to mitigate the effects of this when sample size is in the order of one million.

It has also been shown that in borderline cases during disk block sampling where only two or three hits are made it would be necessary to increase the sample size by 50% in order to be 99% sure of having a true positive hit in the sample. This would increase the time for triage by 50% and a simpler strategy when there are so few hits is to examine the files containing these three clusters. In all other cases this would be unnecessary.

4 Cluster Classification

4.1 Introduction

It has now been shown that contraband content can be detected on a device in a reasonable time but in the context of border security, military and anti-terrorist activities a quick overview of the contents of seized digital media is equally important [61]. As shown in Chapter 2, current digital forensic techniques and tools are not suited to such scenarios. They are aimed mainly at '*post crime*' analysis of digital media. In a time critical situation an investigator needs a data analysis tool that can quickly give a summary of the storage device contents. The quantity of encrypted data on a digital storage device may significantly affect the attitude of a border guard to the profile of the device owner as to whether to prioritise the device for deeper analysis or not.

Garfinkel [46] puts forward the hypothesis that the content of digital media can be predicted by identifying the content of a number of randomly chosen clusters. The hypothesis is justified by randomly sampling 2000 digital storage devices to create a '*forensic inventory*' of each. With only 10,000 4 KiB clusters sampled from drives it was found that, in general, sampled data gave similar statistics to the media as a whole. Thus, when a device is being sampled for contraband as in the previous sections, an analysis of the device contents in terms of file types can be done. Given the sample sizes that are used for contraband detection are many times the size of those used by Garfinkel, a highly accurate forensic inventory of the device can be generated.

The literature refers to 'file fragments' and the term 'fragment' will be used to refer to the 4096-byte clusters that are read from the device during contraband detection. Using the sampling methodology developed earlier to quickly produce a summary of storage device contents requires that data fragments be identified accurately. Research in the area of fragment classification has advanced over the last few years, so that many standard file types can be identified accurately from a fragment. Methods of classification fall into three broad categories: direct comparison of byte frequency distributions, statistical analysis of byte frequency distributions and specialised approaches which rely on knowledge of particular file formats. However none of these methods have been successful in the classification of high entropy file fragments such as encrypted or compressed data. This poses a problem in that recently many file formats such as the Microsoft msx (OOXML) format now use deflate compression by default. In order that an accurate estimation of the quantity of encrypted content that exists on a device it is necessary to be able to distinguish between compressed and encrypted file fragments.

The main aim of this section of the thesis is to investigate and devise methods for the classification of high entropy file fragments in a forensic environment so that an estimate of the encrypted content of a device can be given. It is envisaged that this analysis would be done at the same time and with the same samples used for contraband detection. The contraband detection system is I/O bound and so extra processing for the classification of each fragment can be handled in parallel. First a critical examination of current research into file fragment classification is undertaken. Since no current methods succeed in classifying high entropy fragments, new approaches are proposed and tested. The experiments designed to validate the methods are described and results are reported and critically analysed.

4.2 Background

Historically, file type identification has been achieved through either using the file type extension as in Microsoft operating systems or through using a library of file signatures as used by the Unix ***file*** command [62].

Many file types have these signatures, called '*magic numbers*', embedded within the file header . These tend to be 2-byte identifiers in the header of the file. For example a compressed ZIP file will have 0x504B as the first two bytes at the beginning of the file (ASCII codes for the characters PK - the initials of the developer). This is illustrated in Figure 4 which shows the output from a hex editor while viewing a zip file.

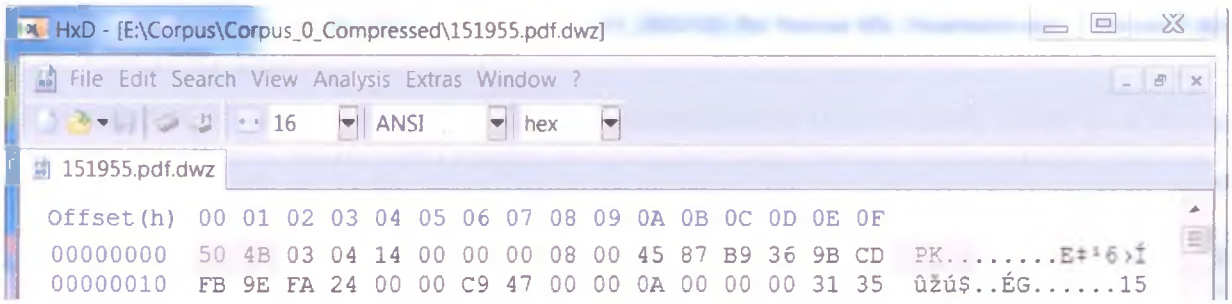


Figure 4 - Zip file showing the first two bytes as the file type 'Magic Numbers'

In addition some file formats have characteristic byte sequences repeated within them. In a JPEG file, a 0x00 byte is appended to each occurrence of the byte 0xFF within the body of the file, thus giving a unique signature to this file type. It should be noted that since the JPEG format is so easily identified, it has been excluded as a compressed type for the remainder of this work. However these simple methods of file type recognition are not available when sampling from a device. It is unlikely that the single file fragment being examined will include the file header. Similarly, other than the specialised case of JPEG files, the file fragments used are unlikely to be long enough to show any significant difference in any signature byte frequency.

Classifying a file fragment from digital media based solely on its content rather than on metadata within the header is a difficult problem. A number of approaches have been proposed which have displayed varying degrees of success. Many of these methods apply statistical analysis to the byte frequency distribution within the fragment to create a '*fingerprint*' or centroid which is held to be characteristic for each file type. An unknown fragment is then classified by its closeness, measured using some metric (a measure of the distance between objects in a vector space) to these centroids.

Others use machine learning techniques. A training set and a test set are created from a corpus consisting of a collection of files of varying types. The results of the statistical analysis done on the training set are used by the machine learning algorithm to create a classifier. The test set can then be used as input to the classifier and its performance measured. If the performance of the classifier is satisfactory then it may be used on unknown file fragments.

4.3 The Problem With High Entropy Fragments

Recent research has found that although classification of file fragments of many common file types can be done with high accuracy, the classification of high entropy file fragments is difficult and to date accuracy has been poor [25], [27]. Indeed Roussev [17] suggest that it is not possible using the current statistical and machine learning techniques to differentiate such fragments.

The entropy of a file fragment measures the amount of randomness or disorder in the data within it. Compressed and encrypted files have little pattern or order. A file is compressed by representing any repeating patterns of data with a shorter code. A well compressed file should have no apparent patterns remaining in it otherwise it could be compressed further. An encrypted file should have no patterns otherwise it would be vulnerable to cryptanalysis. Thus these types are classified as high entropy.

Garfinkel [18] states that the classification of high entropy file fragments is in its infancy and needs further research and it is suggested that future works should investigate methods to improve classification performance on such fragments. To date no such investigations have been done.

4.4 Related Work

Here the current research into file fragment classification is examined. There is a theme running through the research that results in classifying high entropy fragment types has been universally poor. Many investigations have simply excluded these fragment types from their corpora. Roussev and Garfinkel [17] question whether current machine learning or statistical techniques applied to file fragment classification can ever distinguish between these types since these fragments have no discernible patterns to exploit. These findings lead us to develop approaches to the problem.

It can be observed that there has been a trend towards specialised approaches for each file type. Analysis of byte frequency distributions has often proved insufficient to classify fragment types, and the unique characteristics of particular file formats have increased recognition accuracy. It becomes apparent that in many cases neither the digital corpora used nor the software developed is publicly available. It is therefore not possible to validate the research nor do a direct comparison against the methods which are developed. In addition, many results have been derived from small sample sets and thus the results may not be universally applicable. These observations lead us to design the investigation in a manner which will avoid such criticisms.

4.4.1 File Fragment Classification

The idea of using examination of the byte frequency distribution (BFD) of a file to identify a file type was introduced by McDaniel [63]. The BFD is simply a count of the frequency of occurrence of each possible byte value (0-255) in the fragment giving a 256 element vector. Several of these vectors are averaged by adding corresponding elements and dividing by the number of vectors to give an average vector or centroid. For each byte-value the correlation between byte frequency in each file is also recorded. These vectors were taken to be characteristic of that file type and termed the *'fingerprint'* for that type. The BFD of an unknown file type is subtracted from the fingerprint and the differences averaged to give a measure of closeness.

Another fingerprint was developed using a byte frequency cross-correlation (BFC) which measured the average difference in byte pair frequencies. This was a specialised approach which should target certain file types - HTML files, for example, where the characters '<' and '>' occur in pairs regularly. It will be seen that developing such specialised approaches to identify file fragments when byte frequencies of different types are similar is a common occurrence through the research corpus. Average classification accuracy is poor with BFD at around 27% and BFC at around 46%. This result is actually poorer than it appears since whole files were used instead of file fragments and so the file headers which contain magic numbers were included. When file headers were considered by themselves they reported an accuracy of over 90% as would be expected. The corpus consisted of 120 test files with four

of each of the 30 types considered. There is no indication as to the source of the test files and no encrypted files were included. They noted that the ZIP file format had '*a low assurance level*' and that perhaps other classification methods might be needed to improve the accuracy for this type.

Li et al. [64] extended the work of McDaniel and Heydari [63] and developed the methodology that had been introduced by Wang and Stolfo [65] for a payload based intrusion detection system. They used the term n-gram to refer to a collection of *n* consecutive bytes from a byte stream and based their analysis on 1-grams. The terms '*fileprint*' and '*centroid*' were used interchangeably to refer to a pair of vectors. One contained byte-frequencies averaged over a set of similar files from the sample, and, in the other, the variance of these frequencies. In addition, they created a multi-centroid approach by creating several such centroids for each file type since '*files with the same file extension do not always have a distribution similar enough to be represented by a single model*'. The metric used to calculate the distance of an unknown sample from each centroid was that used in [65]. It was proposed originally for computational efficiency in the scenario of a high bandwidth networked environment but this simplification meant that it was no longer suitable as a metric. It was termed a simplified version of the Mahalanobis distance and given as:

$$d(\boldsymbol{x},\boldsymbol{y}) = \sum_{0}^{n-1} \frac{(|x_i - y_i|)}{(\sigma_i + \alpha)}$$

where

- x_i and y_i are the centroid and sample byte frequencies respectively.
- σ_i is the centroid standard deviation for that byte value.
- α is a small positive value added to avoid possible division by zero.

However they note that this simplification requires that the frequency of each individual byte be statistically independent from any other. In cases such as HTML files where the symbols < and > occur frequently as pairs or JPEG files where the hexadecimal values 0xFF and 0x00 frequently occur together, this requirement does not necessarily hold and so the simplification will not be universally valid. Xing [66] describes how learning algorithms depend critically on a good metric being chosen. It has been found that many of the techniques considered below which use machine learning to create centroids give no justification for the metrics used. In this particular research only 8 file types were considered, and no encrypted or compressed files were used, although they noted that all compressed files may have a similar distribution. To create the test corpus 100 files of each type were collected from the internet using a search on Google.

To create their sample set, the first 20, 200, 500 and 1000 bytes from a file were taken. As expected, since these truncated files all contain the file header, the results were good. The average classification accuracy was around 99% for the truncated files with just the first 20 bytes of the file i.e. the file header. As the fragment size increased, accuracy decreased. Accuracy was worst when the whole file, rather than a truncated segment, was used. This could be explained by the fact that with just the first 20 bytes of a file the method reduces to the '*magic numbers*' solution referred to previously. As the file size increases the influence of these first magic numbers is diluted and hence accuracy decreases.

The method named '*Oscar*' was introduced by Karresand [67] using the same BFD vectors as Li et al. [64] to create centroids. This was soon extended to increase the accuracy of JPEG detection by introducing Rate of Change (RoC) of consecutive bytes [68]. The rate of change is maximum for pairs of bytes 0xFF followed by 0x00. As explained earlier the frequency of this byte pair is a unique marker for JPEG files.

They avoid the criticism made of [64] by using a weighted Euclidian metric:

$$d(\boldsymbol{x},\boldsymbol{y}) = \frac{\sum_0^{n-1}(x_i - y_i)^2}{(\sigma_i + \alpha)}$$

to measure distance between an unknown sample and the centroid. If this distance was below a certain threshold then it was taken to be a fragment of that file type.

There is no indication as to the source of their file corpus. 57 files were first padded with zeros to ensure that the file was a multiple of KiB to simulate an unfragmented hard disk of 4 KiB clusters. These 57 files were then concatenated to form one large 72MB file. The file was scanned for each file type separately, and each 4kB block was examined. For compressed (zip) files a fragment was marked as a hit even if it contained header information. The authors noted that compressed types were difficult to tell apart because of the random nature of their byte distribution.

4.4.2 Entropy

Hall and Davis [69] departed from the BFD approach by suggesting that the entropy and compressibility of file fragments be used as identifiers. They used the idea of a sliding window to make *n* steps through the fragment. Compressibility and entropy values calculated at each step were saved as elements of the characteristic vectors, although the window contents were not actually compressed. The LZW algorithm was used and the number of changes made to the compression dictionary used as an indication of compressibility.

Centroids were calculated as usual by averaging element values in a training set. Two metrics were evaluated – the Pearson rank order correlation coefficient and a simple difference metric similar to that used by [63]:

$$d(\boldsymbol{x},\boldsymbol{y}) = \sum_0^{n-1}(|x_i - y_i|).$$

This metric suffers from the same criticism as that of [64] and results were poor. Identification of compressed fragment was only 12% accurate and other results were not given. It was noted that the method might be better at narrowing down possible file types than actually assigning a file type. The initial corpus was a set of files from the authors personal computer.

4.4.3 Complexity and Kolmogorov Complexity

Veenman [70] combined the BFD together with the calculated entropy and estimated Kolmogorov complexity of the fragment to classify the file fragment. The Kolmogorov complexity is a measure of the information content of a string which makes use of substring order. A large corpus of 13 different file types was used with 35,000 files in the training set and 70,000 in the test set. HTML and JPEG fragments were detected with over 98% accuracy, however compressed files had only 18% accuracy with over 80% false positives. Results were presented in a confusion matrix which made them easy to interpret.

Confusion matrices are used by other researchers and will be used in presenting the results, so a short explanation is given here. A confusion matrix is a convenient method of conveying information about the actual types and predicted types made by a classification system. In the confusion matrix in Table 14 it can be seen that JPEG had 126 true positives – fragments which were actually JPEG and labelled as such.

The JPEG type had a total of 86 false negatives – fragments which were actually JPEG but misclassified as another type. It is obvious that this classifier confuses ZIP and JPEG types.

Table 14 - Example Confusion Matrix

		Predicted Type		
		JPEG	BMP	ZIP
Actual Type	JPEG	126	2	84
	BMP	4	85	14
	ZIP	63	12	34

4.4.4 Statistical Methods

Another statistical approach was suggested by Erbacher [71]. Only statistics calculated from the BFD were used rather than the BFD itself. Their analysis used a sliding window of 1 KiB to examine each block of a complete file rather than file fragments. By using the sliding window, however, they could identify component data types within the containing file e.g. a JPEG image within a PDF file. They claimed that five calculated statistics were sufficient to classify the seven data types in the corpus of five files of each type but no results were presented.

This idea was developed by Moody and Erbacher [72] where the corpus consisted of 25 files of each type examined. No compressed or encrypted files were included. It was found that several data types could not be classified because of the similarity of their structure. These files were processed through a secondary pattern matching algorithm where unique identifying characteristics such as the high rate of occurrence of '<' and '>' in HTML files was used for identification. Here again is seen the use of specialised functions for different file types.

4.4.5 Linear Discriminant Analysis

Calhoun and Coles [73] followed the approach of Veenman [70] by using the linear discriminant analysis for classification but used a selection of different statistics. Linear discriminant analysis is used to develop a linear combination of these statistics by modifying the weight given to each so that the classification is optimised. A statistic that discriminates well between classes will be given a bigger weight than one that discriminates less well. They also included a number of tests that could be classified as specialised, such as their ASCII test where the frequency of ASCII character codes 32 to 127 can be used to identify text files such as HTML or TXT. The authors noted that the data did not conform to the requirements of the Fisher linear discriminant that data should come from a multi-variate normal distribution with identical covariance matrices and this may explain some *sub-optimal* results. Overall only four file types were included in the corpus with 50 fragments of each type, and no compressed or encrypted file fragments were included. Fragments were compared in a pairwise fashion. For example JPEG fragments were tested against BMP fragments and the results of classification noted. JPEG was then tested against PDF and so on. There was no attempt at multi-type classification. Testing in such a way gives less chance of misclassification and the results should be interpreted with this in mind. Extensive tables of result for accuracy are given but again there is no data about false positives or true negatives. They noted that a modification to their methodology would be required to avoid the situation where the method fails and all fragments are classified as one type but which gives high accuracy.

4.4.6 Multi-centroid Model

Ahmed et al. [74] developed the methods of Stolfo et al. [75] but introduced some novel ideas. In creating their multi-centroid model they clustered files with similar BFD regardless of type. This implements their assumptions:

1. Different file types may have similar byte frequency distributions.
2. Files of the same type may have different byte frequency distributions.

Within each such cluster, linear discriminant analysis was used to create a discriminant function for its fragment types. Cosine similarity was used as a metric and was shown to give better results than the simplified Mahalanobis distance. The cosine similarity is defined as the cosine of the angle between the centroid, **x**, and fragment, **y**, BFD vectors:

$$\text{Similarity} = \cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}.\mathbf{y}}{|\mathbf{x}||\mathbf{y}|}$$

Since all byte frequencies are non-negative the dot product is positive and therefore the cosine similarity lies in the closed interval [0, 1]. If the cosine similarity is 1 then the angle between the vectors is 0° and they are identical other than magnitude. As the cosine similarity approaches 0, the vectors are increasingly dissimilar.

An unknown fragment was first assigned to a cluster using cosine similarity. If all file types in a cluster were of the same type then the fragment would be classified as that type. If not, then linear discriminant analysis was used to find the closest type match in the cluster. Ten different file types were used although compressed and encrypted types were excluded, and 100 files of each type were included in the training set and the test set. Whole files rather than file fragments were used and so header information was included, achieving 77% accuracy.

4.4.7 Supervised learning models

A number of researchers have used supervised learning methods to create classifiers. These include Support Vector Machines (SVM), k Nearest Neighbour (kNN) and Artificial Neural Networks (ANN). Since these methods will also be used in this research a brief background to each is given here.

A Support Vector Machine (SVM) is a supervised learning model which is used for classification. In supervised learning the training data is a set of data points each with an associated output type. The supervised learning algorithm will infer a classifier which will predict the correct output type for each input data point in the training set.

The SVM algorithm plots each training data point in n-dimensional space and constructs an optimal hyper-plane to separate the two classes. Figure 5 shows that there are many possible classifiers. The one which maximizes the distance between the nearest data points and itself is the optimal hyper-plane. A data point will be classified according to which side of the hyper-plane that it lies.

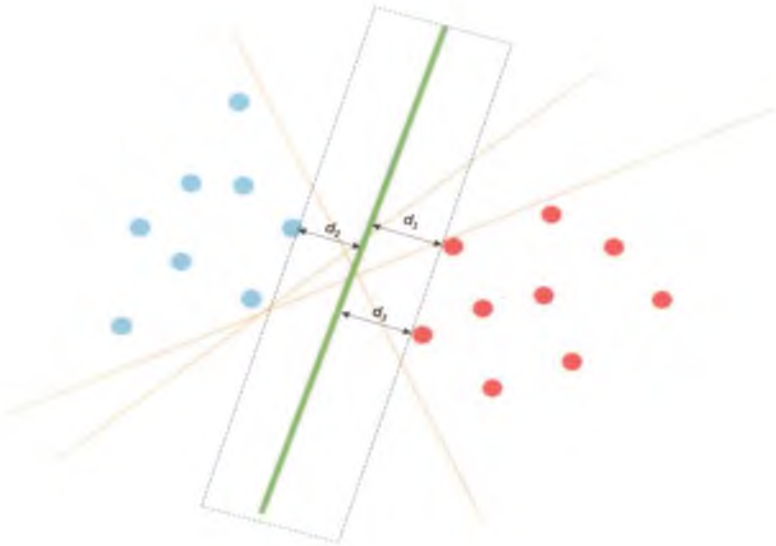


Figure 5 - Optimal Separating Hyperplane maximizes the sum of the distances $\sum_i d_i$ from the support vectors to the plane itself. Illustration based on an idea from [76]

In some cases the data points are not linearly separable in n dimensions. Ben-Hur and Weston [77] recommend the use an SVM kernel which in this instance maps the n-dimensional space into a higher dimensional feature space where the classes may be separable as can be seen in Figure 6. It is worth noting that the kernel mapping need not be linear and so the feature space may allow a separation hyper-surface rather than a hyperplane as shown in Figure 7.

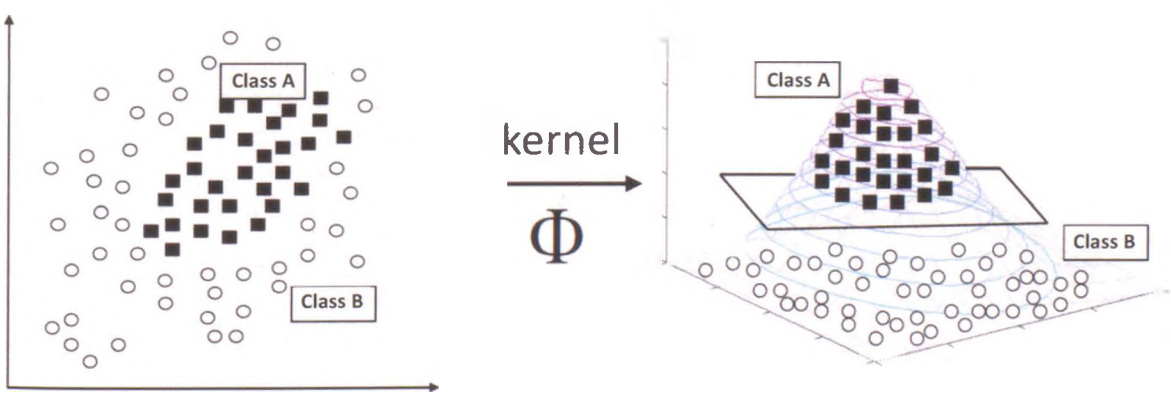


Figure 6 - Data not linearly separable in n-dimensions is mapped to a higher dimensional space by the kernel function. Illustration based on an idea from [76]

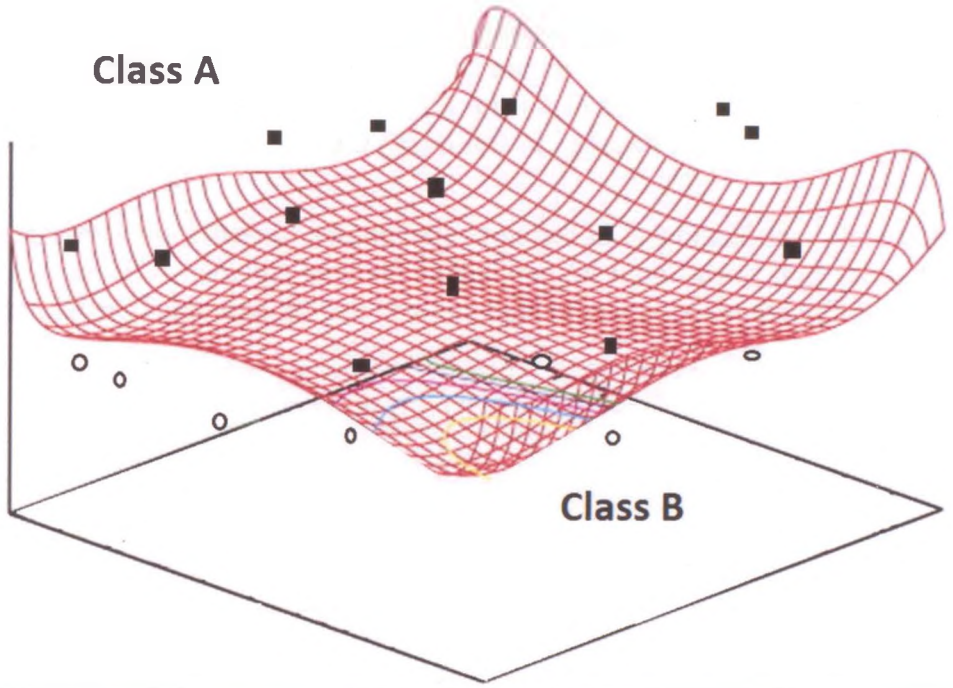


Figure 7 - Optimally separating hyper-surface. Illustration based on [78]

The k Nearest Neighbour (k-NN) approach to classification is simpler than the SVM. No classification function is computed. The training set of classification vectors is plotted in *n*-dimensional space. An unknown example is classified by being plotted and its *k* nearest neighbouring points from the training set determined. The unknown example will be assigned to the class which is most common among these *k* nearest neighbours. The best value for *k* is determined by experiment.

An artificial neural network (ANN) is a is a simulation of the biological brain. A number of processing elements (artificial neurons) are interconnected. The strength of each connection is assigned a weight *w* which is in the range [-1.0,1.0] where -1.0 is maximum inhibition and 1.0 is maximum excitation. Each processing element will have a number of inputs where each input is a value passed by a previous processing element together with its weight.

The sum of the products of the input values and corresponding weights is calculated by the processing unit. The neuron fires if the activation function evaluates to a value above a given threshold. This is illustrated for a single processing element in Figure 8.

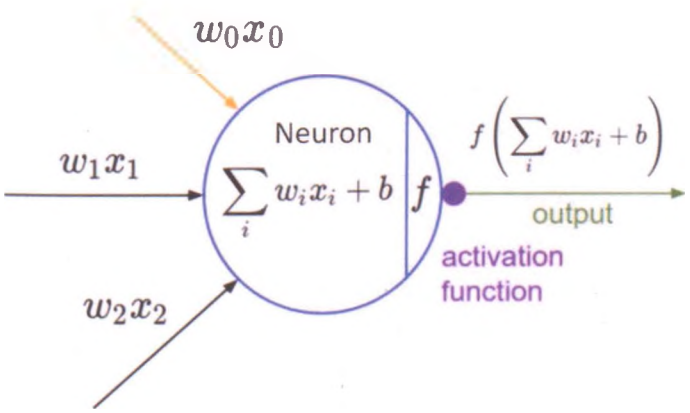
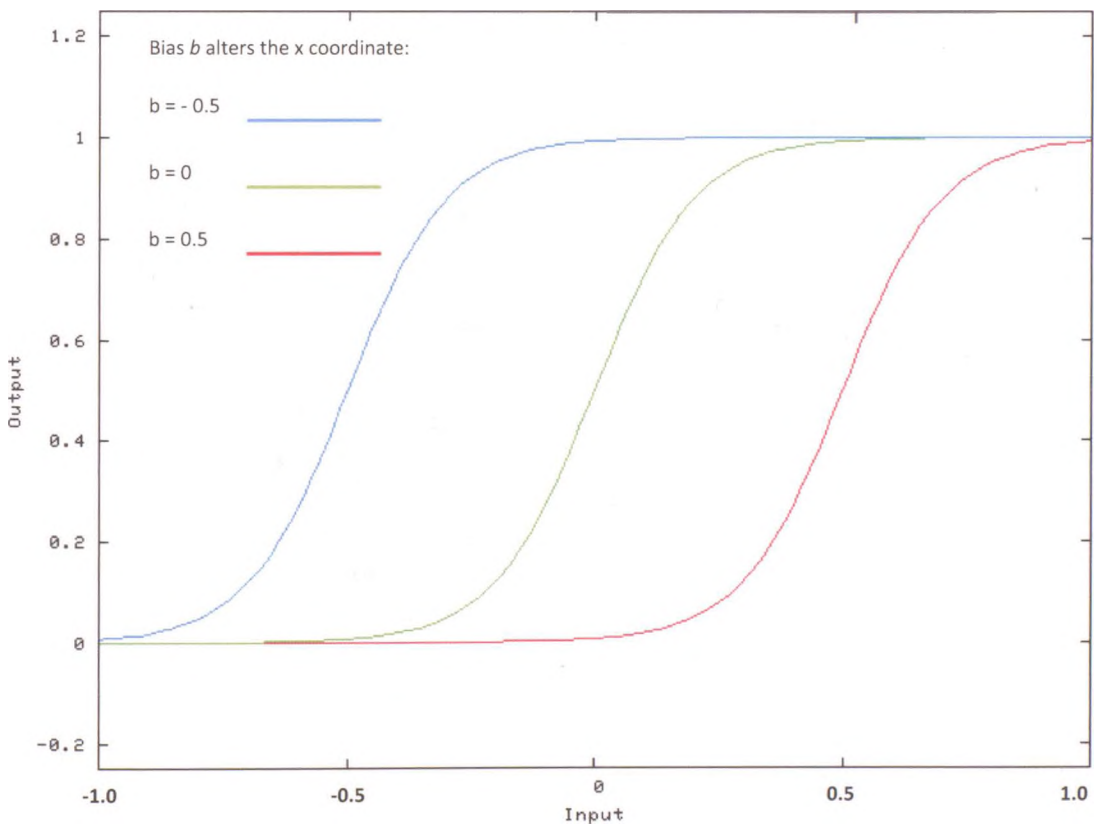


Figure 8 - Individual neuron behaviour. Based on Stanford CS231n [133]

The activation function f shown in the diagram is usually a sigmoid function since a simple step function (0 or 1) would lead to solutions which were simply a linear combination of the input values [79, p. 4]. The sigmoid function allows a smooth transition of values for the output of the processing element. The bias b in the activation formula is a constant that may be added to move the sigmoid graph left or right so that the trigger activation point may be altered (Figure 9).

In a network the artificial neurons are arranged in layers and send their outputs forward to the next layer. The network receives inputs on the input layer, and the output is given by the output layer. There may be one or more hidden layers as shown inFigure 10.



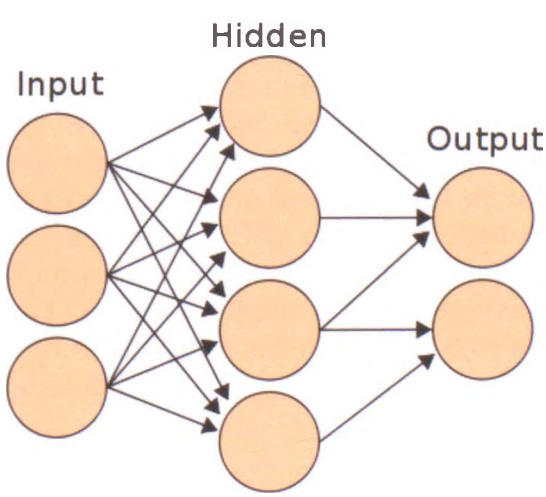


Figure 10 - neural network layers

To train the network a training set of inputs and the corresponding outputs that are required are used. The network can start with a random set of weights. The error is calculated as the difference between the actual output and the required output. The weights are adjusted and the training run repeated until the system converges on the minimum error. The system is then said to have 'learned' the training data. It can be thought of as a *goal seeking* algorithm as used in spreadsheets but with many, perhaps thousands, of possible variables. Once the network is trained it can be tested for accuracy on a test set of data different from the training set. If the system performs well then it can be used to classify live data.

Li et al. [27] used the BFD only and took an approach using a Support Vector Machine (SVM) for data fragment classification. Only four file types were used – JPEG, MP3, DLL and PDF. There were no compressed or encrypted file types other than the easily identified JPEG format. The inclusion of the PDF file type may have affected their results since it is a container type – it can embed a variety of other formats within itself such as JPEG, Microsoft Office or ZIP. Thus it might be difficult to differentiate a fragment of a file labelled PDF from some of the other types. Their training corpus was 800 of each file type downloaded from the internet. Each file was split into 4 KiB fragments and the first and last fragment discarded to ensure that header data and any possible file padding was excluded. The test set was created by downloading a further 80 files of each type from the internet and selecting 200 fragments of each type. Accuracy for classification of the four file types was 81%.

By contrast to previous researchers, Axelsson [80] used the publicly available data set govDocs1 [56], making it easier for others to reproduce the results. The normalised compression distance (NCD) was used as a metric. NCD was introduced by Cilibrasi and Vitanyi [81]. Their idea was that two objects (not necessarily file fragments) are 'close' if one can be compressed further by using the information from the other. If $C(x)$ is the compressed length of fragment x and $C(x,y)$ is the compressed length of fragment x concatenated with fragment y then

$$\mathrm{NCD(x, y)} = \frac{C(x,y) - \min(C(x),C(y))}{\max(C(x),C(y))}$$

The k-Nearest-Neighbour algorithm (kNN) was used for classification. The results were poor and average accuracy was 35%.

In [82], commercial '*off-the-shelf*' software was evaluated against several statistical fragment analysis methods. SVM and kNN using the cosine similarity metric methods were used. The test corpus was created from the publically available RealisticDC dataset [56] and consisted over 36,000 files of 316 file types. The file types are not listed so it is not known if any compressed or encrypted files were included. The true file types were taken to be those reported by Libmagic using the Linux 'file' command and so no account was taken of possible anti-forensic techniques such as file header modification which may have resulted in bias in the results. It was not reported if the first fragment of each file was included. This would have contained header information. The results reported on the performance on file fragments was for the SVM only and it is shown that they achieved 40% accuracy measured using the Macro-F1 measure [83] with a 4096-byte fragment size. There is no breakdown of the results by file type and so it cannot be ascertained if some high accuracy file types are masking some with very low accuracy.

4.4.8 Lempel-Ziv Complexity

Sportiello [84] tested a range of fragment features including the BFD, entropy, Lempel-Ziv complexity and some specialised classifiers such as the distribution of ASCII character codes 32 – 127 which would characterise text based file types, and Rate of Change which has been seen to be a good classifier for JPEG files. The corpus consisted of nine file types

downloaded from the Internet and these were decomposed into a total of 28,000 blocks of 512 bytes for each file type. There is no indication if file header blocks were included. No compressed or encrypted data was included.

An SVM was used but no multi-class classification was attempted. For each file type a separate SVM model was created to classify a fragment type against each of the other types individually. The experiment was actually run using a 4096-byte fragment size and no indication of how these fragments were created is given. There is no confusion matrix of the results and there is no mention of false negative results. The table of results is arranged by fragment type, feature and feature parameter (the C and γ for the SVM). These parameters vary by file type and it appears that only the results for the best parameter values for individual fragment types is given. It is therefore difficult to compare results with others.

Fitzgerald et al. [25] used the data from publicly available govDocs1 corpus. They created 9,000 fragments of 512 bytes for each of 24 file types. The first and last fragment of each file was omitted. There were an equal number of fragments of each file type. They used an SVM using standard parameters. Both unigram and bigram (pairs of consecutive bytes) along with various statistical values were used for the fileprint. They selected up to 4000 fragments for the data set and apportioned these approximately in the ratio 9: 1 as training and test sets respectively for the SVM. There is no mention of whether the 24 file types were represented equally in the selection. An overall accuracy of 47.5% was achieved but correct compressed file prediction averaged 21.8%. It was noted, as did [18], [26], [27] that the classification of high entropy file fragments was challenging.

4.4.9 Specialised Approaches

Roussev and Garfinkel [17] argue that using BFD or statistical methods is too simplistic. They advocate using all tools available for file fragment discrimination. If distinct markers are present within a fragment then these should be used. If a byte pattern suggests a particular file type then knowledge of that file format can be used to check the classification. Thus they would be using purpose-built functions for each file type. They suggest a variety of approaches. As well as the header recognition and characteristic byte sequences as explained in the introductory section they use frame recognition. Many multimedia formats use repeating frames. If the characteristic byte pattern for a frame marker is found then it can be checked if another frame begins at the appropriate offset. If it does then it is likely that the fragment will be of that media type. It should be noted that, unlike other methods, they may require previous or subsequent file fragments. If a fragment type cannot be classified, then they use '*Context Recognition*' where these adjacent fragments are also analysed. Although the govDocs1 file corpus was used, this was supplemented by a variety of MP3 and other files that were specially developed. This does not fit well with their own views expressed in [56] where a strong case was made for the use of standardised digital corpora. A discriminator for Huffman coded files was developed. However its true positive rate was 21%, and the need for further research in this area was stressed in the paper.

4.4.10 Genetic Programming

A novel approach using Genetic Programming was tested by Kattan et al. [85]. 120 examples of each six file types were downloaded at random from the Internet. No compressed or encrypted files were included. Analysis was done on whole files rather than fragments and so file headers may have been included. Features were first extracted from the BFD using Principal Component Analysis (PCA) and passed to a multi-layer artificial neural network (ANN) to produce fileprints. PCA removes redundancy by reducing correlated features to a feature set of uncorrelated '*principal components*' which account for most of the structure in the data. This removal of features is generally accompanied by a loss of information [86, p. 562]. PCA was used here to reduce the number of inputs to the next stage - a multi-layer auto-

associative neural network (ANN) which creates the fileprint. It is mentioned that file headers in themselves were not used, but would be part of the whole file. A 3 layer ANN was used with these fileprints as a classifier for unknown file types. Only 30 files of each type were used for testing. Results were reported in a confusion matrix and averaged 98% true positives. It is not clear whether the PCA would have extracted file headers as the most prominent component of the test data as a classifier. If this was so then the high detection rate would be explained.

This work was extended by Amirani [87] to include detection of file fragments. The original version using an ANN as the final stage classifier was compared with classification using an SVM. 200 files of each of the 6 file types were collected randomly from the internet. Half were used as the training set and half as the testing set. For the fragment analysis a random starting point was selected in each file, and a fragment of 1000 or 1500 bytes was taken. Results showed that the SVM classifier gave better results than the ANN for file fragments of both 1000 and 1500 bytes with extremely good results. It is puzzling that PDF detection gives 89% true positives with the 1500 byte fragments. The PDF format is a container format and might contain any of the other file types examined – doc, gif, htm, jpg and exe – as an embedded object. The random selection of 1500 bytes from within such a file could be mis-classified as the embedded object type. The high detection rate for the PDF type itself means that this must have rarely happened. Perhaps it is an indication that the sample set of 100 files is too small, or perhaps the file header has an undue influence on the PCA.

4.5 High Entropy Fragment Classification

Garfinkel et al. [18] noted that "The technique for discriminating encrypted data from compressed data is in its infancy and needs refinement". This is supported by the observation that, in the literature considered so far, there has been little mention of classification of high entropy types. Where compressed fragments have been included in the test corpus, results have been poor. There is no source that deals with classification of encrypted or random fragments. This area of research has been recognised as difficult [18], [25]–[27]. Most results rely on patterns within the data. However Roussev and Garfinkel [17] argue that compressed and encrypted file types have no such patterns. If a compressed file has patterns then it could be compressed further. If an encrypted file has patterns then it would be vulnerable to cryptanalysis. Therefore it is needed to investigate methods of fragment identification that do not rely on patterns within the data.

4.5.1 Randomness

In Chang et al. [88], the output from a number of compression algorithms and compression programs was tested for randomness. The National Institute of Standards and Technology (NIST) Statistical Test Suite [89] was used. It was found that the output of every compression method failed the NIST tests for randomness. In contrast, during the testing of the candidate algorithms for the Advanced Encryption Standard (AES) it was expected that any encrypted files should be computationally indistinguishable from a true random source [90].

Zhao et al. [91] used 7 large (100 MB) test files and compressed and encrypted them by different methods. The whole 188 NIST tests were run against each file. At this scale they achieved good discrimination of encrypted files. However the 4 KiB fragments that are being sampled during triage are very small compared to these test files.

These observations lead us to the first hypothesis - that it is possible to distinguish between compressed and encrypted fragments by testing for randomness.

4.5.2 Compressibility

Ziv [92] stated that a random sequence can only be considered such if it cannot be compressed significantly. Schneier [93] noted that "*Any file that cannot be compressed and is not already compressed is probably ciphertext*". Mahoney [94] states that encrypted data cannot be compressed. However compression algorithms always have to compromise between speed and compression [95]. It is thus unlikely that a compressed fragment is optimally compressed and therefore can be compressed further.

The second hypothesis, therefore, is that compressed and encrypted fragments can be differentiated by applying an efficient compression algorithm. A compressed file should compress more than an encrypted file.

4.6 Conclusions

Recent research has shown that many file fragment types can be identified with high accuracy. However the classification of high entropy file fragments has been found to be difficult and to date accuracy has been poor. No work has been done on encrypted file types. It has been suggested that it is not possible using the current statistical and machine learning techniques to differentiate between high entropy file fragments. Techniques to do so are in their infancy and methods to improve the classification performance need to be investigated.

A variety of approaches have been employed in classification. Support Vector Machines, Artificial Neural Networks, Genetic Programming and k-Nearest-Neighbour have all been used. However existing methods have failed to find patterns within high entropy file fragments and if patterns are not there to exploit, then simply changing the classification method will not alter the fact. A new approach will be needed that can detect such patterns.

As has been been shown above, randomness and lack of compressibility are characteristics of high entropy file fragments. These characteristics will be investigated to see if they can be used for classification of these file types.

5 Experiment Design

5.1 Introduction

To test these hypotheses a corpus needs to be created to test the classification methods that are developed to distinguish between encrypted and compressed file fragments. In the scientific method it is important that results be reproducible. An independent researcher should be able to repeat the experiment and achieve the same results. In the review of related work it has been seen that this is not generally the case. Most research has been done with private or irreproducible corpora generated by random searches on the WWW. Garfinkel et al. [56] argue that the use of standardised digital corpora not only allows researchers to validate each other's results, but also to build upon them. By reproducing the work a researcher shows that they have mastered the process involved and are then better able to advance the research. In this section the creation of the test corpus from publically available standardised corpora is described.

In the previous section it was found that lack of compressibility and randomness are characteristics of high entropy file fragments. Methods are therefore required to test these characteristics in a file fragment. Statistical methods for testing randomness have already been developed. The NIST Statistical Test Suite [90], [96] was specifically designed for testing randomness in cryptographic applications and was used in the testing of candidate algorithms for the AES encryption standard. This test suite will be used for detecting randomness in file fragments.

5.2 Building the Corpus

Standardised corpora are now available. For example the Govdocs 1 corpus contains a set of 1000 folders each containing 1000 files [56]. Five of these folders were chosen at random to create a training set and ten for a testing set. Folder 0, which had not been one of those randomly chosen, was used while developing the methodology. This avoids any bias introduced by including fragments used in development as part of the training or test corpora.

File fragments of representative compressed and encrypted types from these subsets need to be created: It can assumed that multimedia types which use lossy compression have been classified by the techniques used in the review of related work. Therefore only lossless compression methods will be considered in the remainder of the research.

5.2.1 Compression Methods

There are a number of lossless compression methods. In order that the corpus is representative of compressed files *in the wild*, it will be created using four of the most common. Compressors can be categorised as either stream based, like zip, gzip, and predictive compressors based on prediction by partial matching (PPM), or block based, like bzip2, where a whole input block is analysed at once [81].

The commonly used Deflate compressed data format is defined in RFC 1951. It uses the LZ77 compression method followed by Huffman coding. It was originally designed by Phil Katz for the compression program PKZIP [97]. It uses LZ77 which achieves compression by replacing a repeated string in the data by a pointer to the previous occurrence within the data along with the length of the repeated string. This is followed by Huffman coding which replaces common symbols within the compressed stream by short codes and less common

symbols with longer codes. This method is used by zip and gzip compressors. Although zip and gzip use similar methods, gzip is a compressor for single files whereas zip is an archiver. An archiver can compress multiple files into an archive and decompress single files from within the archive. Zip is a common format on Microsoft Windows platforms, but gzip is primarily a Unix/ Linux compressor. Both zip and gzip will be used as representative of common archivers and compressors in the creation of the corpus.

Bzip2 is a block coding compressor which uses run length encoding (RLE) and the Burrows-Wheeler transform. The B-W transform does not itself compress. It uses a block sort method to transform the data so that the output has long runs of identical symbols which can be compressed efficiently by RLE. The final output is again Huffman coded. Bzip2 is a file compressor rather than an archiver in that it compresses single files only. bzip2 will be used as representative of a block based Unix/ Linux compressor.

There are also several proprietary compression implementations that are commonly used. Winrar is one such. Its own archiving format is proprietary but it is based on LZ and PPM compression. PPM is another stream based compression method which uses an adaptive data compression technique using context modelling and prediction. PPM compressors use previous bytes (bits) in the stream to predict the next byte (bit). They are adaptive in that they adapt the compression algorithm automatically according to the data being compressed. The output is arithmetic rather than Huffman coded. Whereas Huffman coding is restricted to a whole number of bits, many modern data compressors use arithmetic coding which is not restricted by this limitation [94]. It can work with all the compression methods above. Winrar is used in the creation of the corpus as an example of proprietary formats and arithmetic coding.

5.2.2 Encryption Methods

In Microsoft Windows operating systems AES has been the default file and BitLocker drive encryption method since Windows XP. Triple DES has been available as an alternative [98]. AES is also used by popular open source encryption software such as Axcrypt and TrueCrypt.

PGP, together with the open source GnuPG conforming to the OpenPGP standard in RFC4880 is the most widely used cryptographic system [99]. It uses AES, Triple DES, Twofish and Blowfish. AES, Triple DES and Twofish will be used as representative of encryption methods while creating the corpus.

5.2.3 Corpus Creation

Most file systems store files so that the beginning of a file is physically aligned with a sector boundary [18, p. S15]. To emulate randomly sampled disk sectors it will therefore be assumed that each file begins on a sector boundary and consists of 4 KiB blocks. For the same reasons used for the choice of cluster size in contraband detection, a cluster size of 4 KiB will be used. The first sector of any file will contain header information which may be used to identify a fragment type. If the file does not fill the last cluster then this cluster may contain padding or undefined content. For this reason the first and last cluster of any file from the corpus will be excluded.

Each file in the training corpus was compressed individually by each compression method and encrypted by each encryption method. A fragment beginning on a 4 KiB boundary and excluding the first and last fragments was randomly chosen from each file. Files which were less than 12 KiB after compression or encryption were excluded. It is not possible to select a random 4 KiB fragment from such files after first and last 4KiB fragments are excluded. This generated a total of 25,000 fragments in the training corpus. Exactly the same procedure was used on the testing corpus and this generated 46,439 fragments. Each fragment together with details of its source were stored.

5.3 Fragment Analysis Tools

In this section methods are considered to test the hypotheses:

1. Compressed and encrypted fragments can be differentiated by testing for randomness.
2. Compressed and encrypted fragments can be differentiated by applying an efficient compression algorithm. A compressed file should compress more than an encrypted file.

No published work has been done in this area and so a methodology to test the hypotheses will be devised.

5.3.1 Testing Randomness – The NIST Statistical Test Suite

It is important that the output of an encryption algorithm is random, otherwise it would be subject to cryptanalysis. The NIST Statistical Test Suite [89] was used in randomness testing of the AES candidate algorithms to test if their output was truly random [90]. However Chang et al. [88] used the NIST tests and found that compressed data tended to fail randomness tests. This finding will be used to create a classifier. A compressed fragment should display poorer randomness than an encrypted one. The NIST test suite was modified so that it could operate on multiple files and output the results in the correct format for the training of classifiers and then the testing of the classifier accuracy. Classifiers were constructed using kNN, SVM and ANN algorithms for comparison of effectiveness.

The NIST Statistical Test Suite consists of a set of 15 tests. Two of the tests (Cumulative Sums and Serial Test) each return two results. These tests are summarised in Table 15. Note that all tests are done on binary data and not bytes. n is used to denote the number of bits in a sequence.

Table 15 - The NIST Statistical Test Suite

Test Name	Description	Sequence Size Recommendation
Frequency (Monobit)	Proportion of zeroes and ones	$n \geq 100$
Frequency within a block	Splits sequence into N blocks of size M and applies the monobit test on each block	$M \geq 20$, $M > 0.1n$ $N < 100$
Runs Test	Checks if total number of runs of length k, which are sequences of identical bits, is consistent with random data	$n \geq 100$
Block Runs Test	Runs test on data split into blocks of length M	For $n < 6272$, M = 8 For $n < 750K$, M= 128
Binary Matrix Rank	Checks for linear dependence between fixed length substrings	$n \geq 38912$
Discrete Fourier Test	Detects any periodic features in the sequence	$n \geq 1000$
Non-overlapping Template Matching	Checks number of occurrences of a target string of length m in N blocks of length M bits. Skips m bits when pattern found	$n \geq 10^6$
Overlapping Template Matching	As non-overlapping but does not skip	$n \geq 10^6$
Maurer's Universal Statistical Test	Detects if a sequence is significantly compressible	$n \geq 387840$
Linear Complexity	Determines if the complexity of a sequence is such that it can be considered random	$n \geq 10^6$

Serial Test	Checks for uniformity – every m bit sequence should have the same chance of occurring	$m < \lfloor \log_2 n \rfloor - 2$ i.e. floor($\log_2 n$) - 2
Approximate Entropy	Compares the frequency of all overlapping patterns of size m and $m + 1$. These frequencies are compared against what would be expected of a random sequence	$m < \lfloor \log_2 n \rfloor - 5$
Cumulative Sums	Calculates the maximum distance from zero a random walk (the cumulative sum adjusted so 0 is represented by -1, and 1 by 1) achieves.	$n \geq 100$
Random Excursions	Measures deviation from that expected of a random walk (as above) to certain states	$n \geq 10^6$
Random Excursions Variant	Calculates the number of times a given distance from origin is visited in a random walk. Detects deviations from that expected of random sequence	$n \geq 10^6$

The binary matrix rank test, overlapping template matching, Maurer's Universal Statistical test, linear complexity, random excursions and random excursions variant tests cannot be used. The bit sequence from a 4KiB fragment is not long enough to make the results of these tests statistically valid. However, since for each fragment 4096 bytes = 32768 bits are being used, the fragments allow us to use 64 binary sequences of 512 bits which satisfy the size requirements for the other nine tests.

Each test in the NIST test suite produces a result termed a P-value. A P-value is the probability of obtaining a result as extreme as the sample value assuming that the hypothesis that the fragment is random is true. It is suggested that the results of the tests are first analysed in terms of the number of sequences passing each test and then by the distribution of the P-values [89]. 64 sequences of 512 bits are used for each 4096 byte (32768 bit) fragment. For each of these sequences a P-value is calculated. At the 95% level of significance the fragment is accepted as random if the P-value ≥ 0.05. It is normal in hypothesis testing to be looking for evidence to reject the null hypothesis and thus we would be looking for a P-value ≤ 0.05. In this instance, however, we are looking for evidence to support the null hypothesis (that the fragment is random) and thus are looking for P-values ≥ 0.05. The confidence interval for the proportion of these sequences passing a test is given by $\hat{p} \pm \frac{3\hat{p}(1-\hat{p})}{m}$ where $\hat{p} = 0.95$ and m is the number of sequences [89, p. 90]. Using this formula it is expected that at least 60 of the 64 binary sequences making up the fragment will pass the test if the fragment is truly random. The number of binary sequences passing each test will be used as the first component of the characteristic vector. Secondly, if the fragment is random then the P-values calculated in the 64 separate sequence tests should be uniformly distributed. A P-value of P-values is calculated and if this P-value is greater than 0.0001 then the sequence of P values is taken as uniformly distributed [89, p. 91]. This P-value for the uniformity of the distribution of P-values for each test will be used as the second component of the characteristic vector.

A total of nine statistical tests from the test suite will be run against each file fragment. Each test generates two values - the number of the 64 sub-sequences of 512 bits passing the test and a uniformity value. Since two of the tests report two results each there will be a total of 11 pairs of values generated. These pairs of values will be used to form the characteristic vector for each fragment. Thus the characteristic vector v_f for file fragment f is defined as the

sequence $v_f = (n_1, u_1, n_2, u_2, \ldots\ldots\ldots, n_{11}, u_{11})$ where n_i is the number of the 64 sequences passing test i and u_i the uniformity P-value for those 64 tests.

5.3.2 Testing Compressibility

The probability that an encrypted (random) fragment will losslessly compress even by a small amount is low. Consider a random fragment of n bits. There are 2^n possible different fragments. Let P be the probability that the fragment will compress by four bytes (32 bits) or less. Then:

$$P = \frac{Number\ of\ fragments\ that\ compress\ by\ 32\ bits\ or\ less}{Total\ number\ of\ possible\ fragments}$$

If the fragment compresses by 32 bits or less then the fragment of size n must map to one of the fragments of size n-1, n-2, … , n-32. There are only $2^{n-1} + 2^{n-2} + 2^{n-3} + \ldots\ldots + 2^{n-32} = \sum_{m=n-32}^{n-1} 2^m$ such fragments.

Since the compression is lossless, the decompression must map back to a unique original fragment. Thus the mapping must be 1 – 1. Thus there are only this many fragments which will compress by 4 bytes or less.

Thus:

$$P = \frac{\sum_{m=n-32}^{n-1} 2^m}{2^n}$$

$$= \frac{\sum_{m=0}^{n-1} 2^m - \sum_{m=0}^{n-31} 2^m}{2^n}$$

$$= \frac{(2^n - 1) - (2^{n-30} - 1)}{2^n}$$

$$= \frac{2^n - 2^{n-30}}{2^n}$$

$$= 1 - 2^{-30}$$

The probability that a random fragment compresses by 32 bits or less is $1 - 2^{-30}$ which is, for practical purposes, indistinguishable from 1. Also the probability that a random fragment compresses by more than four bytes is therefore 1 – P(compresses by 4 bytes or less) which is for practical purposes equal to zero. Thus if a fragment compresses by more than 4 bytes it can be assumed that it is not encrypted with high confidence. This fact will be used to classify the fragments.

A compression algorithm which will meet several requirements will be needed. Firstly an algorithm which will compress more optimally than standard algorithms such as 'deflate' will be needed. Secondly, *deflate* uses <length, distance> pairs and literals which are then Huffman coded. There is a chance that literal bytes will align on a byte boundary and so a bytewise compressor might see them. However in Huffman coding the data is packed as bits and the three-bit Huffman header will throw out this alignment. Also literals are likely to become rare further into the stream. The situation is worse with dynamic Huffman coding as codes can be nearly any length. A bitwise compression algorithm, however, is not constrained by lack of byte alignment. It will be able to see repeating literals or <length, distance> pairs [100]. For these reasons zpaq will be used as the compressor. In addition to being a suitable bitwise compressor it uses bit prediction. It maintains a set of context models which independently create probabilities for the next bit. The probabilities are combined to make the prediction. It would be expected that, by definition, it would not be possible to predict the

next bit in a random fragment. In a fragment which has not been optimally compressed, however, there must be some remaining pattern and hence predictability otherwise it would be optimally compressed. Zpaq should be able to detect this and give a more optimal compression. The classifier in this instance will be simply the compressed size of the fragment.

5.4 Conclusions

In this chapter the methodology has been described and justified. It has been shown how the training and test corpora of encrypted and compressed file fragments was created. The corpora have been generated from those publically available so that an independent researcher should be able to repeat the experiment and achieve the same results. Compression and encryption methods used in the creation of the corpus have been chosen to be representative of real world usage. The possibility of file header or footer information biasing the results has been excluded.

It has been seen that compressed fragments may fail the NIST test suite for randomness. Encrypted fragments should not. The output from the tests will be used to create the characteristic vector for each fragment. These will be used in kNN, SVM and ANN machine learning algorithms to first create the classifier and then test it.

It has also been shown in 5.3.2 that the probability of a file fragment whose content is random compressing by more than 4 bytes is 2^{-30} and is therefore unlikely to compress significantly. A compressed fragment may compress further. This will be used to classify the fragments as compressed or encrypted.

6 Implementation and Results

6.1 Introduction

In this chapter the k-NN, SVM and ANN methods described previously are used to classify encrypted and compressed file fragments and the best solution chosen. As described in section 5.3.1 the characteristic vector v_f for a file fragment f is defined as the sequence $v_f = (n_1, u_1, n_2, u_2, \ldots\ldots\ldots, n_{11}, u_{11},)$ where n_i is the number of the 64 sequences passing test i and u_i the uniformity P-value for those 64 tests is derived for each fragment by the NIST test suite and used for classification.

6.2 Statistical Analysis of Randomness

The NIST test suite was modified to operate on fragments and to output the characteristic vector to be used for classification for each fragment. The characteristic vectors generated from the training corpus were used to train a classifier that was then used to classify fragments from the test corpus. k-NN, SVM and ANN analyses of the results were implemented using RapidMiner [101] and compared.

Initial testing using k-Nearest Neighbour with the development corpus found that k=3 together with the Euclidian distance as a metric was optimum for the k-NN analysis.

The SVM was created using LibSVM [102] and a radial kernel trained using default settings. The default radial kernel function was found by experiment on the development corpus to produce the most accurate results. 10-fold cross validation with stratified sampling was used to optimise the model parameters. In 10-fold cross validation the training set is partitioned into 10 subsets. The SVM is trained on 9 subsets and the remaining one is used to test the resulting model. Each of the ten subsets is used in turn as the testing set. Parameters are automatically chosen to minimise the error. In stratified sampling the subsets are chosen so that the proportion of file types in each subset reflects the overall proportion in the training set. The resulting classification model was used on the testing set.

The ANN was set up with parameters optimised using the training set. 10 fold cross-validation was again used during training.

The results of these analyses on the 46,439 test corpus fragments are shown in Table 16. In previous research where compressed fragments have been included in the test corpus the results have been poor with the best previous results by Fitzgerald et al. [25] achieving a true positive rate of 21.8% for classifying compressed fragments. There is no previous source that attempted to differentiate between encrypted and random fragments since this was seen as difficult [18], [25]–[27]. It can be seen immediately from Table 16 that classification accuracy is higher than any previously achieved. The ANN provided the best results with an overall 85% correct classification.

Table 16 - Results of classification of the test corpus by each machine learning algorithm

		Predicted Type		
Method	Actual Type	Encrypted	Compressed	Accuracy
k-NN	Encrypted	21518	1937	92%
	Compressed	9734	13250	58%
SVM	Encrypted	22758	697	97%
	Compressed	9776	13208	57%
ANN	Encrypted	21938	1517	94%
	Compressed	5665	17319	75%

6.3 Classification By Compression

In Table 16 it can be seen that classification of compressed fragments was poorer than for those that were encrypted. Each statistic calculated for compressed fragments showed a greater variance than for encrypted. This is illustrated in Figure 11 where the Shannon 8 bit entropy of encrypted fragments in a sample has a smaller variance than the compressed fragments leading to the distribution being much more tightly grouped around the mean and less chance of misclassification.

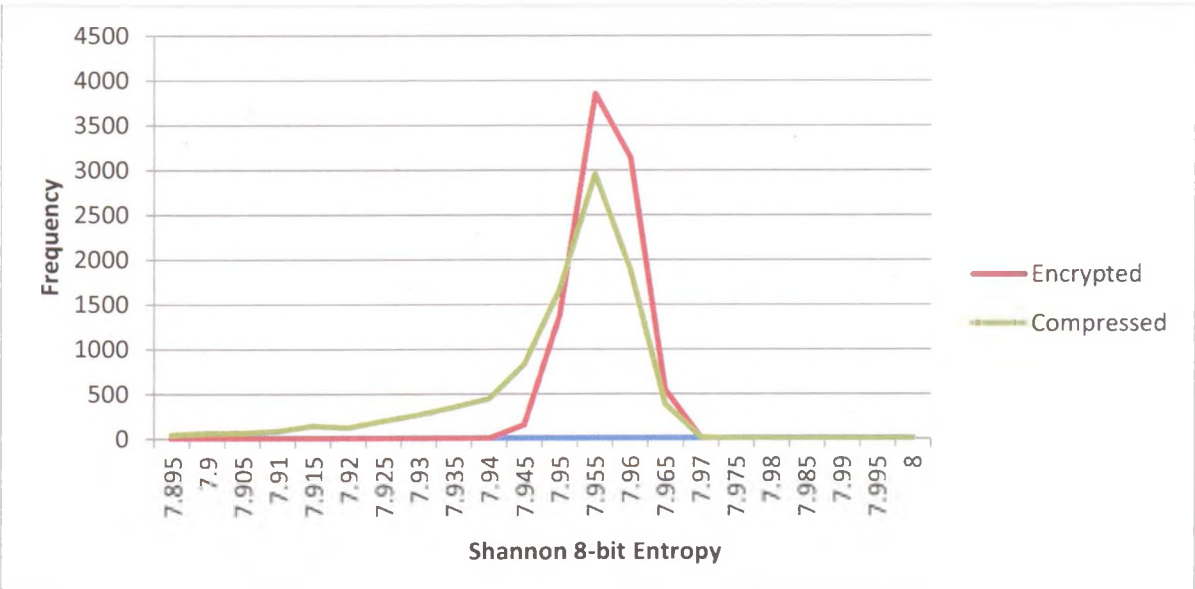


Figure 11 – Frequency distribution for the entropy of a random sample of file fragments

Analysis of the compression results is simpler than the statistical tests for encryption. There is one figure produced for each fragment – the size of the compressed fragment. The hypothesis is that a compressed fragment will compress more than an encrypted fragment. The fragments were classified by their compressed file size. If it was bigger than a given size then it was classified as compressed. Otherwise it was classified as encrypted. Since, for each fragment, only a single figure is being compared with a fixed value the problem has very small dimensionality. The fixed value (4 bytes as calculated in 5.3.2) is already known and thus there is no need for the application of machine learning techniques.

Any compressed file is accompanied by additional information which may include file headers, archive filenames, file paths, filenames, dates, Huffman tables and dictionaries depending on the compressor used. It quickly became apparent that this additional data added to a 4KiB fragment when it was compressed totally masked the small changes in file size that were trying to be measured. It turned out that most fragments became larger after

compression. This would not matter if the 'additionality' is uniform. If this is so, the small byte size changes that were expected if the hypothesis is true would still be apparent. This uniformity was achieved by using the *–tiny* flag with the compressor. Table 17 shows results for classifying the test corpus fragments using compression.

Although the classification of compressed files is better than the k-NN and SVM NIST statistical tests, the encrypted fragment detection is poorer. The overall accuracy of results at 73% are poorer than using the NIST statistical analysis. The statistical analysis of randomness is therefore the preferred method for classification.

Table 17 – Analysis of Fragment Compression classification by category

	Predicted Type		
Actual Type	Encrypted	Compressed	Accuracy
Encrypted	17826	5629	76%
Compressed	6895	16089	70%

6.4 Fragment Size

It was decided to investigate if different fragment sizes had an effect on classification accuracy. A smaller fragment size would speed up analysis but cannot be used since the NIST tests are not statistically valid with smaller fragment sizes.

6.4.1 8KB Fragments

A corpus of 8 KiB fragments was created and the ANN trained on a training set from the test corpus. The NIST tests were run on the testing set to see if there would be any improvement in classification accuracy. The results for the ANN analysis are shown in Table 18. It can be seen that there is a small improvement over the 4KB fragment size. The most notable is that compressed fragment detection has risen from 75% to 82% but the overall correct classification is similar. However this test highlighted a problem. The NIST statistical tests do not scale linearly. Doubling the fragment size made the NIST test suite take over 2 hours when sampling a 1 TB drive, which is unacceptable in a triage situation.

To investigate this the NIST tests were run with a timer and disabled one test at a time. With the Serial test enabled, each fragment took, on average, 2.54 seconds to analyse. This is unacceptably long in the front line investigative forensic environment. With the Serial test disabled each fragment analysis took 0.005 seconds on average.

It was decided to do a performance evaluation on the components of the characteristic vector to find out how much of a contribution each was making to the classification. Components can be removed if they have little influence on the classification. If the tests which produce these components are removed, then this should help to increase the speed of the analysis.

A weight optimisation system was set up using a recursive artificial neural network. This will allocate a weight for each component or attribute. The weight of each attribute reflects its overall effect on the classification. If an attribute has a very small weight then that attribute can be removed from the analysis without a major effect on the accuracy. A simple feed-forward neural network for attribute weighting could not be used because the condition that all attributes are independent could not be fulfilled. It was therefore decided to us an evolutionary genetic algorithm. This works by feeding the training data to the artificial neural network (ANN) and weights are assigned to each attribute. The weighted attributes are used in cross validation with the training data and the results fed back to the ANN which produces a new set of weights. This continues until the weights converge – when there is little or no

difference between generations. The results are shown in Table 19. The weights are shown for each of the two values that NIST produces for each test – the P uniformity measure and the number of bit streams passing each test.

The results show the relevant importance of each component of the characteristic vector for classification. It can be seen that the Serial test does have some significance but does not have a large influence on the overall classification.

The classification process was run on 8K blocks again without the Serial test. It took under 4 minutes to analyse the 4,579 fragments in the training corpus and a similar time for the testing corpus. Results are shown in Table 20.

The results are near identical to the 8KB test which used the Serial test showing that its removal did not affect the results significantly but increased the speed 30-fold. This also shows that the classification can be achieved at over 1000 fragments per minute.

Table 18 - ANN results with 8KB fragment size

	Predicted Type		
Actual Type	Encrypted	Compressed	Accuracy
Encrypted	2108	197	91%
Compressed	320	1432	82%

Table 19 - Weight Analysis of NIST statistical tests

NIST Result	Weight
FrequencyP	0
Frequency	0.922
BlockF_P	0.603
BlockF	0.261
CumSumF_P	0.235
CumSumF	0.858
CumSumB_P	0.195
CumSumB	0
Runs_P	0.600
Runs	1
LongestRun_P	0
LongestRun	0.739
DFT_P	0.257
DFT	0.428
Serial1_P	0.202
Serial1	0.031
Serial2_P	0.152
Serial2	0.083

Table 20 – ANN results with 8KB fragment size, no Serial test

	Predicted Type		
Actual Type	Encrypted	Compressed	Accuracy
Encrypted	2093	212	91%
Compressed	291	1461	83%

6.5 Conclusions

The approaches which have been developed have been implemented in a manner which makes the results available for other researchers to validate. A baseline has been set in a new area of research against which others can compare methods and results. The overall classification accuracy using 8 KiB fragments was found to be not markedly better than the 4 KiB analysis. Also the sampled clusters in the contraband detector are 4 KiB and these can be used directly with the statistical analysis. For these reasons, a 4 KiB fragment is used in the next section where the practical implementation of this classification methodology in parallel with contraband detection is considered.

7 Constructing A Classifier

7.1 Introduction

Garfinkel [46] shows that sample sizes of as little as 10,000 4 KiB blocks can give similar statistics for device contents to the media as a whole. It is intended that the encrypted fragment classifier will run in parallel with the contraband detector and so sample sizes, and hence accuracy, will be much higher. Since the entropy of each 4 KiB cluster is calculated as part of the NIST analysis, this value can be used to filter the 4 KiB blocks from the contraband detector so that any fragment that is unlikely to be encrypted or compressed can be ignored. The average entropy value for compressed fragments in the 46,439 test corpus was 7.73 with 94% with values greater than 7.5. For encrypted fragments the average entropy was 7.95 and the minimum was 7.94. Since it is the proportion of encrypted fragments that is being estimated it would therefore be reasonable to set an entropy threshold of 7.5 so that only fragments with an entropy greater than this would be passed to the classifier. This greatly reduces the number of fragments needing analysed and so the achieved speed already achieved of 1000 fragments per minute would be adequate to process fragments from the contraband detection sample.

Given that the sampling is random, the proportion of encrypted fragments in the sample will reflect the proportion of encrypted data on the device [103, Ch. 6.2]. For example if a 250 GB device is being sampled by the contraband detector and 53000 samples taken (see Table 7) then if 2000 of the samples were classified as encrypted then the proportion of encrypted fragments in the sample is 2000 divided by 53000 which is 0.038. It would be expected then that 0.038 × 250 MB = 9.5 GB of the data on the device is estimated to be encrypted. Similarly if 20 samples were found to be encrypted then the proportion would be 20/ 53000 = 0.00037 and so 0.00037 × 250 GB = 92.5 MB of data on the device is estimated to be encrypted.

7.2 Improving Accuracy

If errors are consistent then they can be managed. For example if it is known that a watch consistently reads 5 minutes slow then this error can be compensated for so that the next train is not missed. If a production line is known to produce 5% defective items then production planning can still be done quite accurately by compensating for the known defect rate. If the testing procedures used in this research produce consistent results then the same methods can be applied to compensate for the false positive rate and more accurately predict the number of encrypted fragments. In order to determine if the classification errors in the analysis of fragments can be similarly handled, consistency was tested for as follows.

Five directories from the Govdocs1 corpus [56] were selected at random to create the training set and fragments of encrypted and compressed files created as before. This generated a total of 24,637 fragments in the training corpus.

Ten randomly chosen directories from the corpus were used for the testing corpus. Exactly the same procedure was used on the testing corpus and this generated a total of 48,709 fragments. The directories chosen for testing were distinct from those used for training. Each folder of approximately 4500 fragments was analysed separately to check for consistency in classification.

The training set was used with an artificial neural network (ANN) and 10-fold cross validation to create a classification model. This model was used to analyse the test directories of fragments with the following results. As can be seen in Table 21 the classification accuracy of encrypted fragments is highly consistent. The average accuracy of encrypted files is 92% and of compressed files is 67%.

Table 21 - Consistency of Results

Directory Number	Actual Type	Predicted Type		Accuracy
		Encrypted	Compressed	
027	Encrypted	2270	207	92%
	Compressed	722	1600	69%
050	Encrypted	2231	220	91%
	Compressed	718	1635	69%
158	Encrypted	2012	192	91%
	Compressed	727	1355	65%
220	Encrypted	2140	191	92%
	Compressed	814	1473	64%
374	Encrypted	2253	201	92%
	Compressed	741	1677	69%
410	Encrypted	2212	197	92%
	Compressed	740	1457	66%
679	Encrypted	2423	235	91%
	Compressed	875	1848	68%
869	Encrypted	2251	203	92%
	Compressed	862	1551	64%
891	Encrypted	2266	214	91%
	Compressed	951	1582	62%
922	Encrypted	2411	201	92%
	Compressed	833	2018	71%

7.3 Methodology

Consider the results for folder 410 from Table 21 where row and column totals have been added for convenience. This is shown in Table 22.

Table 22 - Results for folder 410 extracted from Table 21

		Predicted Type			
		Encrypted	Compressed	Total	Accuracy
Actual Type	Encrypted	2212	197	2409	91.8%
	Compressed	740	1457	2197	66.3%
	Total	2952	1654	4606	

Using the accuracy the total number of fragments that will be predicted to be encrypted can be calculated as the number of encrypted fragments that will be predicted to be encrypted plus the number of compressed fragments that will be predicted to be encrypted:

$0.918 \times 2409 + (1 - 0.663) \times 2197 = 2951$

Similarly the predicted number of compressed fragments can be calculated as $0.663 \times 2197 + (1 - 0.918) \times 2409 = 1654$.

These can be seen to be correct from the bottom row of Table 22 given that the accuracy of 91.8% is rounded to one decimal place.

In general terms,

Let n = total number of fragments

a_e = actual number of encrypted fragments

a_c = actual number of compressed fragments

p_e = predicted number of encrypted fragments

p_c = predicted number of compressed fragments

f_e = accuracy of encrypted prediction as a fraction

f_c = accuracy of compressed prediction as a fraction

Then:

$$p_e = f_e \times a_e + (1 - f_c) \times a_c \quad (6)$$

$$p_c = f_c \times a_c + (1 - f_e) \times a_e \quad (7)$$

From Eqn. 7,

$$a_c = \frac{1}{f_c} \times [p_c - (1 - f_e) \times a_e]$$

Substituting for a_c in Eqn. 6 gives:

$$p_e = f_e \times a_e + (1 - f_c) \times \frac{1}{f_c} \times [p_c - (1 - f_e) * a_e]$$

$$= f_e \times a_e + (\frac{1}{f_c} - 1) \times [p_c - (1 - f_e) \times a_e]$$

$$= f_e \times a_e + \frac{p_c}{f_c} - \left(\frac{1-f_e}{f_c}\right) \times a_e - p_c + (1 - f_e) \times a_e$$

Re-arranging gives:

$$f_e \times a_e - \left(\frac{1-f_e}{f_c}\right) \times a_e + (1 - f_e) \times a_e = p_e + p_c - \frac{p_c}{f_c}$$

Hence:

$$a_e \times [f_e - \left(\frac{1-f_e}{f_c}\right) + 1 - f_e] = p_e + p_c \times (\frac{f_c - 1}{f_c})$$

So:

$$a_e = \frac{p_e + p_c \times (\frac{f_c - 1}{f_c})}{1 - (\frac{1-f_e}{f_c})} \quad (8)$$

Hence it has been shown that the actual number of encrypted files a_e can be calculated from the predicted number of compressed and encrypted files together with the respective accuracies. It was also shown in the previous section that the accuracies were consistent and that the average accuracies were 92% and 67% respectively for encrypted and compressed fragments, so these quantities can be used as a best estimate for f_e and f_c.

Substituting these values into Eqn. (8) gives:

$$a_e = \frac{p_e - 0.493 \times p_c}{0.881} \quad (9)$$

The best estimate of the actual number of encrypted fragments is now calculated purely from the results of the classification method.

7.4 Implementation and Results

For each of the ten test corpora the predicted number of encrypted and compressed fragments was calculated using the classification results given in Table 21 and using the formula developed for a_e in equation (9). The results are shown in Table 23. The methods developed in this thesis have achieved better than 95% accuracy on average in predicting the number of encrypted and compressed fragments in all corpora. Having shown that Equation (9) together with the fragment classification methods developed gives consistent accuracy in the order of 95% for the estimation of the number of encrypted and compressed fragments in a sample, this will be used in a subsequent section for estimating the encrypted content of storage devices.

Having shown that Equation (9) together with the fragment classification methods developed gives consistent accuracy in the order of 95% for the estimation of the number of encrypted and compressed fragments in a sample, this will be used in a subsequent section for estimating the encrypted content of storage devices.

Table 23 - Actual and predicted number of encrypted and compressed fragments

Directory	Encrypted Fragments			Compressed Fragments		
Number	Actual	Predicted	Error	Actual	Predicted	Error
027	2477	2399	3%	2322	2400	3%
050	2451	2323	5%	2353	2481	5%
158	2204	2257	2%	2082	2029	3%
220	2331	2437	5%	2287	2181	5%
374	2454	2361	4%	2418	2511	4%
410	2409	2440	1%	2197	2166	1%
679	2658	2593	2%	2723	2788	2%
869	2454	2567	5%	2413	2300	5%
891	2480	2663	7%	2533	2350	7%
922	2612	2454	6%	2851	3009	6%

Fragments that have been compressed and then encrypted as in a standard crypto system are the output of an encryption process and therefor will be detected as encrypted. Standard crypto systems do compression before encryption since it can be shown that encrypted data will not compress significantly (see 5.3.2). If a fragment is from a file that has been encrypted and then compressed, in what may be termed a 'reverse crypto system', then its classification will depend on the compression method used. Many compressors will not attempt to compress data that will not significantly compress and will just add the data in its original format to the compressed archive. In this case, since encrypted data will be stored in its original format, the fragment would also be detected as encrypted. However Johnson et al. [104] show that, in very specific cases, an encrypted file can be compressed. In these cases only it might be possible for an originally encrypted fragment which is then compressed to be misclassified as compressed. They state, however, that in general, no compression gain can be achieved. Any

such cases are likely to be rare and so this is unlikely to add significantly to the error rate of detection.

7.5 Speed Of Analysis

For implementation and testing the process was done in two stages. Firstly the file fragment characteristic vectors were created using the NIST Statistical Test Suite [96]. These vectors were then analysed using a neural network within the package RapidMiner [101]. This procedure would not be feasible in the developed triage software and so alternatives are considered here. The contraband detection system has been shown to be I/O bound and so there is processor time available during the sampling and contraband detection. The speed achievable by such an implementation is considered to determine if it would be suitable for running in parallel.

7.5.1 Statistical Testing

Sys et al. [105] have optimised the NIST Statistical Test suite and timings for the tests that are run for the fragment classification total 0.7 seconds for 20 MB of data. This equates to over 7,000 fragments per second. The contraband detector is scanning at 180 fragments per second on a hard disk drive and 25,000 fragments per second on a solid state device (Table 7). The speed of the statistical analysis far exceeds the sampling speed of the HDD and so this process would still be I/O bound, with the statistical analysis having no effect on the overall speed of the triage. With the SSD however, the statistical analysis is slower than the sampling rate and would therefore limit the triage speed. From Table 7 it can be seen that the contraband detection takes 13 seconds for 281,000 samples while detecting a target of 4 MiB of contraband. It would take the statistical analysis approximately 40 seconds for this number of samples. Even if the tasks could not be done in parallel, this would only increase the triage time by 40 seconds to 53 seconds. This can be considered a trivial increase in a typical investigation.

7.5.2 ANN Analysis

It was seen that the ANN, which has been implemented in a rapid application development graphical environment, is still capable of over 1000 classifications per minute. Although this may seem slow it should be remembered that only samples with an entropy of over 7.5 will be passed to the ANN for analysis. From the examples in the introduction to this section this could be as many as 2000 samples for a 250 GB drive with approximately 10 GB of encrypted files. This analysis would take two minutes at the speed of the development ANN. This adds very little to the overall time needed for triage in the case of an SSD with the total time being now in the order of three minutes. The added two minutes makes little difference the triage of an HDD.

7.6 Classifier Implementation

As a proof of concept the proposed system was tested on a 160 GB hard disk drive. The drive was forensically wiped since the cluster sampling employed would pick up any of the old data on a disk that had simply been formatted or files deleted. Encrypted files and compressed files were added to the disk together with files that were known to be neither compressed, encrypted nor a container type. At each stage when more files were added it was ensured that equal amounts of each type of data were resident on the disc.

The integration of the components of the classification system is still ongoing and so for the proof of concept the tests were run in stages.

1. The drive was sampled using the contraband detection software and each cluster sampled was saved.

2. Once the sampling was finished, each cluster was analysed by the NIST statistical analysis software and its characteristic vector produced and saved.
3. Each characteristic vector where the entropy was greater than 7.5 was analysed by the ANN and its classification recorded. The ANN used the classifier trained in 6.2 so a training step was unnecessary.
4. The proportion of clusters classified as encrypted in the total sample was used to calculate the estimate of encrypted data on the disk using Equation (8).

The results are shown in Table 24.

Table 24 - Estimates of encrypted content

Encrypted Content		
Actual	**Predicted**	**Error**
10 MiB	9.96 MiB	3.8%
20 MiB	19.38 MiB	3.1%
50 MiB	52.35 MiB	4.7%
100 MiB	102.93 MiB	2.9%
1 GiB	1.045 GiB	2.1%
5 GiB	5.24 GiB	4.7%
10 GiB	10.09 GiB	0.9%

It can be seen that the quantity of encrypted content of the disk is being reported with greater than 95% accuracy over a wide range of encrypted disk content.

7.7 Conclusions

The aim of this part of the thesis was to mitigate the problem caused when the volume of data on a device being inspected at border post makes a full scan of a computer for contraband material impractical to be done in any reasonable time. As has been noted previously the quantity of encrypted data on a digital storage device may significantly affect the attitude of a border guard to the profile of the device owner. An inability or unwillingness to produce the encryption key would lead, at the very least, to the device being held for further detailed examination. Thus in this situation a reliable reporting of the quantity of encrypted material on a device is crucial.

Amthod has been developed to mitigate this problem. At the same time as the device is being scanned for contraband the same samples can be classified by original file type to estimate the quantity of encrypted content. The classification methods proposed and implemented have been shown to give consistent accuracy. Given this fact, it was shown that the known error rate for classification can be compensated for in the estimation of the device content increasing the accuracy of the estimate. Thus the classification of high entropy file fragments has been shown to be feasible and that reasonable estimates of the quantity of encrypted data can be done at an acceptable speed in parallel with contraband detection.

8 Conclusions

The aim of this thesis was to investigate a solution to the current and growing problem facing the digital forensic community of the ever increasing capacity of digital devices and the consequent time taken to analyse the contents. A system of triage was envisaged where a quick and accurate scan of a device could be done and a decision made as to whether it required further analysis or not. The system was required to identify known contraband files on a device and classify the sampled file fragments as encrypted or not so that a profile of the disk contents could be made. The research in this thesis has shown that such a system of triage is possible and a working prototype has been developed and shown to meet the requirements of a fast scan with controllable accuracy that can execute on legacy equipment.

8.1 Overview

In the first section of this thesis a methodology to detect contraband files involving the sampling of devices at the physical level and bypassing the file system was implemented. The current practice of detecting contraband by file hashes was replaced by hashing 4 KiB blocks of data. These blocks align with the data stored on disk [24] and so can be used in the same manner as a file hash. The presence of a 4 KiB contraband block on disk implies that the original file is or was once present. A Bloom filter was used to store a compact representation of the block hash database so that it could be used in memory to provide fast lookup. The mathematics of Bloom filter error rates and sampling theory was used to determine appropriate sample sizes for a given degree of accuracy. The system was implemented and it was shown that a fast scan of large capacity digital storage devices could be done to the required degree of accuracy and in a reasonable time. The implementation has been trialled on live systems by law enforcement agencies and shown to work well within the agreed parameters of speed and accuracy and should help to cut the three-month backlog of digital forensic investigations that is currently the norm in the UK [8], [9].

A possible problem with the system was identified by the existence of non-probative blocks – those that may appear in many files and so can't be used to prove the existence of a file on the media [22], [24]. This was investigated to quantify the scale of the possible problem by the use of publically available multi-million block corpora and a mitigation strategy was devised. It was shown that the scale of the problem is small but mitigation is still needed if the desired accuracy is to be maintained. The contraband detection system together with the simple mitigation strategy of needing three contraband blocks to be detected during a scan in order to classify a device as requiring further analysis was implemented and shown still to meet the desired degree of confidence and speed that was acceptable in the field. Forensic analysts at Police Scotland have found that live cases have either many contraband files or none and the requirement for three hits is of little consequence. It would be a rare case that was positive with only three hits and in this instance it would be a matter of moments to manually examine the files containing these blocks (Police Scotland, personal communication, 23rd March 2016). Thus the system of triage for file fragment identification together with mitigation for non-probative blocks has been achieved.

To address the problem of detecting encrypted content from 4 KiB file fragments the current state of research in file fragment classification was critically evaluated. It was found that most common file formats can be detected from file fragments but classification of high entropy

file fragments such as those that are encrypted or compressed had not been attempted. This area of classification has been stated to be difficult [25], [27], [17], [18]. No attempt had been made at the classification of encrypted fragments. Where classification of compressed fragments has been attempted, results had been poor. Two hypothesis were proposed to distinguish between high entropy file fragments.

Firstly encrypted fragments should be statistically indistinguishable from random data [90] and therefore be 'more random' than compressed data. Compression algorithms always have to compromise between speed and compression [95]. It is thus unlikely that a compressed fragment is optimally compressed and therefore can be compressed further. If a fragment can be compressed further then it must have some pattern and therefore cannot be random.

Secondly, as has been discussed above, a compressed fragment should compress further whereas it was shown mathematically that an encrypted fragment will not. In addition a file is compressed by representing any repeating patterns of data with a shorter code. An encrypted file should have no patterns otherwise it would be vulnerable to cryptanalysis and therefore cannot be compressed.

A methodology was devised using these hypotheses which classifies encrypted and compressed file fragments. A detection rate of 97% for encrypted fragments and 78% for compressed fragments has been achieved. These results have been achieved in an area where, to date, classification had been thought to be difficult. A proof of concept implementation of the methodology was done and shown to detect encrypted content with good accuracy within a general corpus of files. The proportion of encrypted data on a device was reported consistently within 3% of the actual content. The method was shown to be capable of matching the speed of contraband detection and so the triage system achieves its aim of fast, accurate contraband and encrypted data classification which can be done in parallel with contraband detection.

The approaches developed in this thesis have been implemented using publically available corpora so that the results may be validated by other researchers. This also allows them to do a direct comparison of any advance in methodology against the methods which were developed in this thesis.

The methods developed speed up the digital forensic process in several ways. Firstly devices can quickly be triaged and a decision made whether a device needs further investigation or not. This cuts the number of devices passed to the forensic facility and thus reduces the workload. Secondly, if a device is found to have contraband, then the location on the device of the sampled contraband is already known thus aiding the analyst in the analysis of the device. Thirdly, as the triage progresses, the device content is profiled and the proportion of encrypted material reported. Thus the thesis aims of developing methodologies that will help reduce the backlog of devices needing analysed in digital forensic laboratories and the speeding of analysis of devices at border posts have been met.

8.2 Future work

The focus of this thesis has been on the application of the novel methodologies introduced to the area of digital forensic analysis with particular attention to the triage of large capacity storage devices. Opportunities exist in many areas for the application of the methods for sampling and small block identification. The payload in each TCP/IP packet in network traffic can be considered as a sample of the data being transmitted and the methodology may be modified to deal with such data. The focus has also been on working with law enforcement for the fast detection of contraband. The same methodology can be applied to the area of Data Loss Prevention where the contraband block hash database could be replaced by a block hash

database of a company's intellectual property and confidential files. Some possibilities are outlined below.

8.2.1 Secure Sharing of Contraband and Intellectual Property Fingerprints

Police forces and other law enforcement agencies are reluctant to share their contraband databases since if the hash databases become publically available the hashes can be used by potential offenders to find illegal content online. The methods developed in this thesis create a fingerprint which consists of a sequence of eight hashes for each block which are stored in a Bloom filter. It is impossible to extract the fingerprint of an individual file or block from a Bloom filter since any bit in the filter may be set by the hashes of multiple blocks. In addition only 264 bits of the 384 bit SHA384 hash are used to enter data in the Bloom filter. Even if the 264 bits could be retrieved then the remaining 120 bits would be unknown and so finding files using a partial hash would be impractical. As a result fingerprints are completely secure, and cannot be used by potential offenders or competitors to find illegal content online. This will allow police forces and other investigative agencies to share their databases in a completely secure fashion, increasing the coverage achieved by investigations.

8.2.2 Multi-agency Operation and Data Loss Prevention

There is much potential in using the contraband detector in an inter-domain model, especially in the area of data loss prevention. Each domain would have an on-site contraband builder which creates the block hash database together with metadata including location details of their confidential documents and sends it to a remote server. Since only hashes are sent and these are not reversible none of the confidential information is sent off site.

This is illustrated in Figure 12. The creation of Bloom filters and their incorporation into a bespoke triage agent which could be distributed to client hardware could be handled from the central agency.

In the client organisations any contraband hash detected would be forwarded to the remote database and its original location checked. Both the organisation where the fragment was detected and the organisation where it originated could be alerted. This is illustrated in Figure 13.

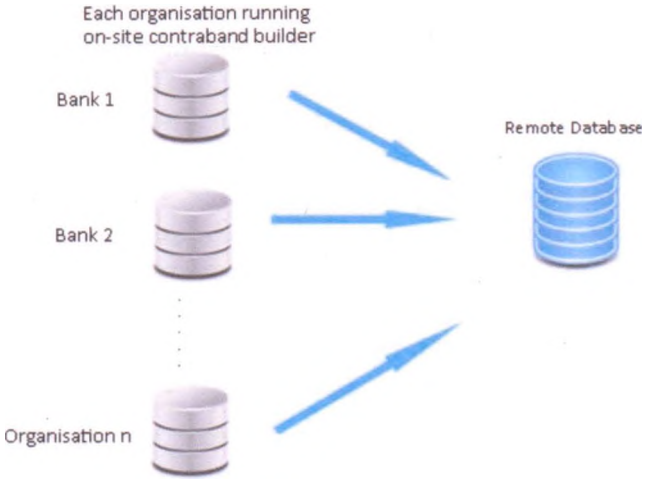


Figure 12 - Each domain sends hashes + metadata

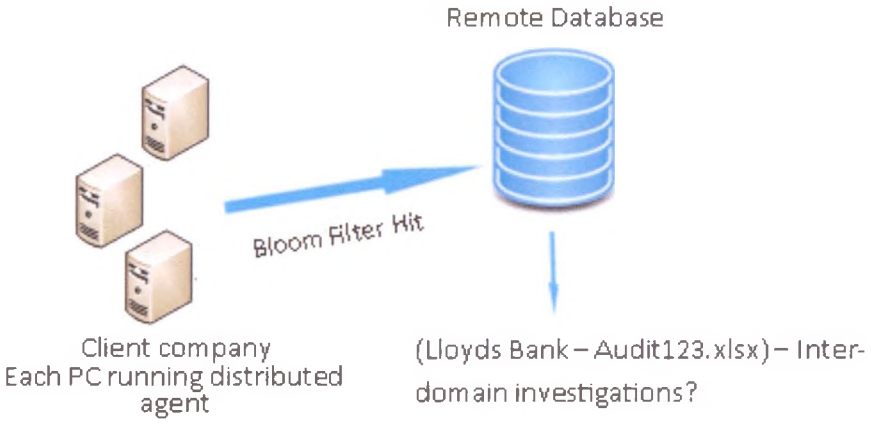


Figure 13 - Contraband discovery handled centrally

There are ethical considerations here with regard to the reporting to one organisation about data held by another. These issues would need to be agreed by all parties before implementation of any cross-domain reporting. In addition a position would have to be taken by the centralised administration as to what action to take when it has knowledge that one organisation holds confidential information from another, but there is no agreement between the two organisations for sharing results.

8.2.3 Operations in a cloud infrastructure

Investigation is needed into the application of sampling within block, file and object storage within a cloud infrastructure. Access to the hypervisor privilege level would be necessary to directly access hardware devices but given this there is no reason that a contraband detector could not be continually operational in sampling the infrastructure. Individual hits could be flagged and a hit could focus more intensive scrutiny on that storage instance. Of course there would be ethical implications here. Any client using the service should be aware that all storage is scanned for contraband. Since only block hashes are used they could be assured that at no time is any file accessible.

8.2.4 An alternative compression detection algorithm

In theory it has been shown that the compression of file fragments should lead to accurate classification of compressed and encrypted fragments. In practice, however, the small differences in compressed fragment size are masked by the additional data such as filenames, paths, archive details and journaling entries added by the compression software. In addition many compression algorithms will also produce a dictionary or a Huffman table to be appended to the compressed data for the purposes of decompression. The file fragment which is being classified has none of this additional data. The compression software which was used is open source. In order to compare like with like, it should be investigated if the source code for any of the compression programs can be modified to report the raw compressed size without the additional data. A program is needed that embeds none of the data needed for decompression nor appends any dictionaries, filenames or archive details. This may achieve results that are closer to those expected by theory

In the review of the literature it was seen that the trend in file fragment classification is toward specialised approaches. A specialised approach using knowledge of the file format could be used here. Such an approach could be used to pre-process high entropy fragments for evidence of compression. The deflate algorithm stores data in variable length blocks. A possible approach is that it should be possible to detect uncompressed data in a fragment of a compressed file by checking for a block header (BFINAL 0 or 1, BTYPE 00) and then checking that the next 32 bits complied with RFC 1951 - LEN and NLEN being ones complements [97]. In a similar manner RFC 1950 defines a 2-byte header for a ZLIB block.

The 2 bytes are constrained to be a multiple of 31 when viewed as a 16 bit unsigned integer [106]. There are 2^{16} = 65536 possible 32 bit unsigned integers. Of those, 65536 / 31 = 2114 are multiples of 31. Hence only 3% of random byte pairs will be a multiple of 31. If such a byte pair is found then there is a 97% chance that it is part of a ZLIB block header.

These three classification methods could be built in to the analysis since the fragment bit stream is already being analysed. With such pre-processing used to classify and remove these compressed fragments the overall classification should improve. These methods are likely to be very processor intensive and an investigation into whether they would be feasible in a triage situation would be necessary.

8.2.5 IP Streams

The sampling methodology has application in sampling IP streams. Fixed size 4 KiB fragments cannot be used because the payload of a TCP/IP packet is not of fixed length and so hashing will not match that of hashes of fixed 4 KiB blocks. It is not feasible to rebuild a packet stream and check for contraband. However similarity hashing, sometimes called fuzzy hashing, may be used to detect if a packet payload originated from a file in a contraband library. Proof of concept testing is already underway and a variety of similarity hashing methods have been trialled – sdhash [107], MRSHnet [108] and MRSHcf [109]. Initial tests are promising with contraband detection 100% positive while inspecting all packets at a rate of 1 GBit/sec. When sampling is introduced this can increase the speed considerably depending on the sampling rate required.

8.2.6 Cuckoo Filters

In this research Bloom filters have been used for high speed set membership tests. As has been previously shown they are especially fast when returning negative results i.e. when testing for a cluster that is not in the contraband set. This behaviour was one of the characteristics that led to the selection of a Bloom filter as the preferred option for testing set membership since during sampling it was expected that most samples would not be from the contraband set. A recent development has been a new data structure called a cuckoo filter [110]. Each entry in a cuckoo filter consists of a number 'buckets' each of which stores a shorter fingerprint for a hash value instead of the hash value itself. It has two major advantages over the Bloom filter. Items may be removed from a cuckoo filter, which is not possible in a Bloom filter. Any bit in the Bloom filter may have been set by multiple hashes and so deleting the bits for one hash may delete bits associated with multiple items. In addition the cuckoo filter can offer better compression than the Bloom filter, allowing up to 95% occupancy so that a cuckoo filter may be only 60% the size of an equivalent Bloom filter for the same false positive rate. However this compression is dependent on the fingerprint size required to achieve the required false positive rate. The fingerprint length $|f_h|$ is given by the following formula [109]:

$$|f_h| \geq \lceil \log_2(2b/\varepsilon) \rceil = \lceil \log_2(1/\varepsilon) + \log_2(2b) \rceil$$

where ε is the false positive rate and b is the number of buckets per entry.

Given that a false positive rate of less than one in a million, then this formula gives $|f_h| \geq 23$. Thus each entry would require 23 bits per entry for each hash in the contraband database giving a total size for the 200 million hashes of the contraband database of approximately 548 MiB. Writing or tailoring an implementation of a cuckoo filter needs to be done so that performance in terms of speed, size and false positive rates can be compared with that of the Bloom filter.

References

[1] E. Casey, *Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet*, 3rd ed. Academic Press, 2011.

[2] R. Attoe, *Digital forensics in an eDiscovery world.* Elsevier Inc., 2015.

[3] C. Walter, "Kryder's Law," *Scientific American*, no. August 1, 2005.

[4] B. Marchon, T. Pitchford, Y. T. Hsia, and S. Gangopadhyay, "The head-disk interface roadmap to an areal density of 4 Tbit/in 2," *Advances in Tribology*, vol. 2013, no. Figure 2, 2013.

[5] S. Garfinkel, "Digital forensics research: The next 10 years," *Digital Investigation*, vol. 7, pp. S64–S73, Aug. 2010.

[6] V. Roussev, C. Quates, and R. Martell, "Real-time digital forensics and triage," *Digital Investigation*, vol. 10, no. 2, pp. 158–167, 2013.

[7] A. Goldberg, "Child abuse cases delayed by police backlog," *BBC 5 Live Investigates*, 2015. [Online]. Available: http://www.bbc.co.uk/news/uk-34713745. [Accessed: 18-Jul-2016].

[8] Her Majestys Inspectorate of Police, "Online and on the edge : Real risks in a virtual world," 2015.

[9] The Parliamentary Office of Science and Technology, "Digital Forensics and Crime," 2016. [Online]. Available: http://www.statewatch.org/news/2016/mar/uk-post-note-digital-forensics-crime-3-2015.pdf. [Accessed: 20-Apr-2016].

[10] J. I. James and P. Gladyshev, "A survey of digital forensic investigator decision processes and measurement of decisions based on enhanced preview," *Digital Investigation*, vol. 10, no. 2, pp. 148–157, Sep. 2013.

[11] B. Hitchcock, N.-A. Le-Khac, and M. Scanlon, "Tiered forensic methodology model for Digital Field Triage by non-digital evidence specialists," *Digital Investigation*, vol. 16, pp. S75–S85, 2016.

[12] M. Pollitt, *The key to forensic success: Examination planning is a key determinant of efficient and effective digital forensics*. Elsevier Inc., 2015.

[13] A. Shaw and A. Browne, "A practical and robust approach to coping with large volumes of data submitted for digital forensic examination," *Digital Investigation*, vol. 10, no. 2, pp. 116–128, Sep. 2013.

[14] T. A. Hoffer, J. L. E. Shelton, S. Behnke, and P. Erdberg, "Exploring the Impact of Child Sex Offender Suicide," *Journal of Family Violence*, vol. 25, no. 8, pp. 777–786, 2010.

[15] S. W. Craun and P. J. Detar, "Designated as Armed and Dangerous," *Journal of Criminal Justice*, vol. 43, no. 5, pp. 437–442, 2015.

[16] T. Hoffer, J. L. E. Shelton, and C. Joyner, "Operational safety considerations while investigating child sex offenders: A handbook for law enforcement," Washington, DC, 2012.

[17] V. Roussev and S. L. Garfinkel, "File Fragment Classification-The Case for Specialized Approaches," *2009 Fourth International IEEE Workshop on Systematic*

Approaches to Digital Forensic Engineering, pp. 3–14, May 2009.

[18] S. Garfinkel, A. Nelson, D. White, and V. Roussev, "Using purpose-built functions and block hashes to enable small block and sub-file forensics," *Digital Investigation*, vol. 7, pp. S13–S23, Aug. 2010.

[19] P. Penrose, W. J. Buchanan, and R. Macfarlane, "Approaches to the classification of high entropy file fragments," *Digital Investigation*, vol. 10, no. 4, pp. 372–384, Dec. 2013.

[20] P. Penrose, W. J. Buchanan, and R. Macfarlane, "Fast contraband detection in large capacity disk drives," *Digital Investigation*, vol. 12, no. S1, pp. S22–S29, 2015.

[21] DFRWS, "Digital Forensic Research Workshops," *Digital Forensic Research Workshops (DFRWS)*, 2015. [Online]. Available: https://www.dfrws.org/2015eu/.

[22] P. Penrose, W. J. Buchanan, R. Macfarlane, O. Lo, and B. Ramsay, "The Effect Of Non-probative Blocks On Disk Sampling For Forensic Triage," *Digital Investigation*, vol. In Press, 2016.

[23] The Home Office, "The Child Abuse Image Database The Child Abuse Image Database (CAID)," 2015.

[24] S. L. Garfinkel and M. McCarrin, "Hash-based carving: Searching media for complete files and file fragments with sector hashing and hashdb," *Digital Investigation*, vol. 14, pp. S95–S105, Aug. 2015.

[25] S. Fitzgerald, G. Mathews, C. Morris, and O. Zhulyn, "Using NLP techniques for file fragment classification," *Digital Investigation*, vol. 9, pp. S44–S49, Aug. 2012.

[26] G. Conti et al., "Automated mapping of large binary objects using primitive fragment type classification," *Digital Investigation*, vol. 7, pp. S3–S12, Aug. 2010.

[27] Q. Li, A. Ong, P. Suganthan, and V. Thing, "A novel support vector machine approach to high entropy data fragment classification," *Proceedings of the South African Information Security Multi-Conference*, 2010.

[28] M. H. Kryder, "After Hard Drives—What Comes Next?," *IEEE Transactions on Magnetics*, vol. 45, no. 10, pp. 3406–3413, Oct. 2009.

[29] (Association of Chief Police Officers) ACPO, "ACPO Managers Guide Good Practice and Advice Guide for Managers of e-Crime Investigation." .

[30] E. Casey, "Editorial – A sea change in digital forensics and incident response," *Digital Investigation*, vol. 17, pp. A1–A2, 2016.

[31] J. Young, K. Foster, S. Garfinkel, and K. Fairbanks, "Distinct Sector Hashes for Target File Detection," *Computer*, vol. 45, no. 12, pp. 28–35, 2012.

[32] M. M. Pollitt, "Triage: A practical solution or admission of failure," *Digital Investigation*, vol. 10, no. 2, pp. 87–88, Sep. 2013.

[33] G. Horsman, C. Laing, and P. Vickers, "A case-based reasoning method for locating evidence during digital forensic device triage," *Decision Support Systems*, vol. 61, pp. 69–78, May 2014.

[34] V. S. Harichandran, F. Breitinger, I. Baggili, and A. Marrington, "A cyber forensics needs analysis survey: Revisiting the domain's needs a decade later," *Computers and Security*, vol. 57, pp. 1–13, 2016.

[35] E. Casey, G. Katz, and J. Lewthwaite, "Honing digital forensic processes," *Digital Investigation*, vol. 10, no. 2, pp. 138–147, Sep. 2013.

[36] D. Quick and K. K. R. Choo, "Big forensic data reduction: digital forensic images and electronic evidence," *Cluster Computing*, vol. 19, no. 2, pp. 1–18, 2016.

[37] S. Garfinkel, "Digital media triage with bulk data analysis and bulk_extractor," *Computers & Security*, vol. 29, 2013.

[38] V. Roussev, Y. Chen, T. Bourg, and G. G. Richard, "md5bloom: Forensic filesystem hashing revisited," *Digital Investigation*, vol. 3, pp. 82–90, Sep. 2006.

[39] J. Kornblum, "Identifying Almost Identical files using context triggered piecewise hashing," *Digital Investigation*, vol. 3, no. SUPPL, pp. 91–97, 2006.

[40] V. Roussev, "Data Fingerprinting With Similarity Digests," in *Advances in Digital Forensics VI*, K.-P. Chow and S. Shenoi, Eds. Springer Berlin Heidelberg, 2010, pp. 207–226.

[41] P. Farrell, S. L. Garfinkel, and D. White, "Practical Applications of Bloom Filters to the NIST RDS and Hard Drive Triage," *2008 Annual Computer Security Applications Conference (ACSAC)*, pp. 13–22, Dec. 2008.

[42] Guidance Software, "Encase Forensic v7." [Online]. Available: https://www.guidancesoftware.com/products/Pages/encase-forensic/overview.aspx. [Accessed: 23-Mar-2014].

[43] Fujitsu Technology Solutions GmbH, "WHITE PAPER FUJITSU PRIMERGY SERVER," 2011.

[44] Microsoft, "Microsoft Default Cluster Size," *Knowledge Base*, 2013. [Online]. Available: http://support.microsoft.com/kb/140365. [Accessed: 08-Jul-2014].

[45] IDEMA, "The Advent of Advanced Format," *International Disk Drive Equipment and Materials Association*, 2013. [Online]. Available: http://www.idema.org/?page_id=2369.

[46] S. Garfinkel, "Random Sampling with Sector Identification," *Naval Postgraduate School Presentation*, 2010. [Online]. Available: http://simson.net/ref/2010/2010-02-11.pdf. [Accessed: 20-Mar-2014].

[47] L. Zhang, "Hypergeometric Distribution," *Applied Statistics 1*, 2008. [Online]. Available: http://www.math.utah.edu/~lzhang/teaching/3070summer2008/DailyUpdates/jun23/sec3_5.pdf. [Accessed: 10-Aug-2014].

[48] E. Weisstein, "Combinations," *MathWorld - A Wolfram Web Resource*. [Online]. Available: http://mathworld.wolfram.com/Combination.html. [Accessed: 07-Jun-2014].

[49] W. P. (ed) Vogt, "Dictionary of Statistics & Methodology Central Limit Theorem," in *Dictionary of statistics & methodology*, 3rd edn., Thousand Oaks, CA: SAGE Publications, Inc., 2005, p. 42.

[50] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7. pp. 422–426, 1970.

[51] M. Mitzenmacher and S. Vadhan, "Why simple hash functions work: exploiting the entropy in a data stream," *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 746–755, 2008.

[52] Random.org, "Pregenerated Random Numbers." [Online]. Available: http://www.random.org/files/.

[53] B. Shavers, "Windows Forensic Environment," 2014. [Online]. Available:

http://brettshavers.cc/index.php/brettsblog/tags/tag/winfe. [Accessed: 20-Mar-2015].

[54] Microsoft, "Microsoft NTFS File Sector Information Utility." [Online]. Available: http://support.microsoft.com/kb/253066. [Accessed: 07-Jul-2014].

[55] K. Foster, "Using distinct sectors in media sampling and full media analysis to detect presence of documents from a corpus," *Masters Thesis, Naval Post Graduate School*, 2012.

[56] S. Garfinkel, P. Farrell, V. Roussev, and G. Dinolt, "Bringing science to digital forensics with standardized forensic corpora," *Digital Investigation*, vol. 6, pp. S2–S11, Sep. 2009.

[57] LIACS Medialab, "The MIRFLICKR Retrieval Evaluation." Leiden University, The Netherlands, 2013.

[58] International Color Consortium, "Image technology colour management — Architecture, profile format, and data structure," 2010.

[59] P. Bromiley, "Shannon entropy, Renyi entropy, and information," *Statistics and Information Series*, no. 2004, pp. 1–8, 2004.

[60] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. July 1928, pp. 379–423, 1948.

[61] J. Giordano and C. Macaig, "Cyber Forensics : A Military Operations Perspective," *International Journal of Digital Evidence*, vol. 1, no. 2, pp. 1–13, 2002.

[62] R. Dhanalakshmi and C. Chellappan, "File format identification and information extraction," *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, pp. 1497–1501, 2009.

[63] M. McDaniel and M. Heydari, "Content based file type detection algorithms," *Proceedings of the 36th Annual Hawaii International Conference on System Sciences, 2003.*, 2002.

[64] W. Li, K. Wang, S. Stolfo, and B. Herzog, "Fileprints: Identifying file types by n-gram analysis," *Proceedings of the 2005 IEEE Workshop on Information Assurance and Security*, pp. 64–71, 2005.

[65] K. Wang and S. J. Stolfo, "Anomalous Payload-based Network Intrusion Detection," *Recent Advances in Intrusion Detection*, vol. 3224, pp. 203–222, 2004.

[66] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell, "Distance Metric Learning , with Application to Clustering with Side-Information," *Learning (2003)*, vol. 15, no. 2, pp. 505–512, 2003.

[67] M. Karresand and N. Shahmehri, "Oscar - File Type Identification of Binary Data in Disk Clusters and RAM Pages," *Proceedings of the 2006 IEEE Workshop on Information Assurance*, vol. 201, pp. 140–147, 2006.

[68] M. Karresand and N. Shahmehri, "File type identification of data fragments by their binary structure," *Information Assurance Workshop ...*, pp. 140–147, 2006.

[69] G. Hall and W. P. Davis, "Sliding window measurement for file type identification," *IEEE Information Assurance Workshop*, 2006.

[70] C. J. Veenman, "Statistical Disk Cluster Classification for File Carving," *Third International Symposium on Information Assurance and Security*, pp. 393–398, Aug. 2007.

[71] R. F. Erbacher and J. Mulholland, "Identification and Localization of Data Types

within Large-Scale File Systems," *Second International Workshop on Systematic Approaches to Digital Forensic Engineering SADFE07, 55-70*, 2007.

[72] S. J. Moody and R. F. Erbacher, "SÁDI - Statistical Analysis for Data Type Identification," *2008 Third International Workshop on Systematic Approaches to Digital Forensic Engineering*, pp. 41–54, May 2008.

[73] W. C. Calhoun and D. Coles, "Predicting the types of file fragments," *Digital Investigation*, vol. 5, pp. S14–S20, Sep. 2008.

[74] I. Ahmed, K. Lhee, H. Shin, and M. Hong, "On Improving the Accuracy and Performance of Content-Based File Type Identification," pp. 44–59, 2009.

[75] S. Stolfo, K. Wang, and W. Li, "Fileprint analysis for Malware Detection," New York, 2005.

[76] "Optimal Separating Hyperplane," 2013. [Online]. Available: http://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Classification/SVM. [Accessed: 23-Sep-2014].

[77] A. Ben-hur and J. Weston, "A User ' s Guide to Support Vector Machines Preliminaries : Linear Classifiers," *Methods in Molecular Biology*, vol. Oliviero C, pp. 223–239, 2009.

[78] R. Berwick, "An Idiot ' s guide to Support vector machines (SVMs) SVMs : A New Generation of Learning Algorithms Key Ideas." pp. 1–28, 1990.

[79] C. Gershenson, "Artificial Neural Networks for Beginners," *Networks*, vol. cs.NE/0308, p. 8, 2003.

[80] S. Axelsson, "The Normalised Compression Distance as a file fragment classifier," *Digital Investigation*, vol. 7, pp. S24–S31, Aug. 2010.

[81] R. Cilibrasi and P. Vitanyi, "Clustering by Compression," *IEEE Transactions on Information Theory*, vol. 51, no. 4, pp. 1–28, 2004.

[82] S. Gopal, Y. Yang, K. Salomatin, and J. Carbonell, "Statistical Learning for File-Type Identification," *2011 10th International Conference on Machine Learning and Applications and Workshops*, no. DiiD, pp. 68–73, Dec. 2011.

[83] A. Özgür, L. Özgür, and T. Güngör, "Text Categorization with Class-Based and Corpus-Based Keyword Selection," *Proceedings of the 20thInternational Conference on Computer and Information Sciences*, pp. 606–615, 2005.

[84] L. Sportiello and S. Zanero, "File Block Classification by Support Vector Machine," *2011 Sixth International Conference on Availability, Reliability and Security*, pp. 307–312, Aug. 2011.

[85] A. Kattan, E. Galván-López, R. Poli, and M. O'Neill, "GP-fileprints: File types detection using genetic programming," *Genetic Programming*, pp. 134–145, 2010.

[86] B. C. Geiger and G. Kubin, "Relative Information Loss in the PCA," in *Proceedings IEEE Information Theory Workshop*, 2012, pp. 562–566.

[87] M. Amirani, M. Toorani, and S. Mihandoost, "Feature-based Type Identification of File Fragments," *Security and Communication Networks*, vol. 6, no. April 2012, pp. 115–128, 2013.

[88] W. Chang, B. Fang, X. Yun, S. Wang, X. Yu, and M. Ethodology, "Randomness Testing of Compressed Data," *Journal of Computing*, vol. 2, no. 1, pp. 44–52, 2010.

[89] A. Rukhin, J. Soto, and J. Nechvatal, "A Statistical Test Suite for Random and

Pseudorandom Number Generators for Cryptographic Applications," *NIST Special Publication (2010)*, vol. 22, no. April, 2010.

[90] J. Soto, "Randomness testing of the AES candidate algorithms," *NIST. Available via csrc. nist. gov*, p. 14, 1999.

[91] B. Zhao, Q. Liu, and X. Liu, "Evaluation of Encrypted Data Identification Methods Based on Randomness Test," *2011 IEEE/ACM International Conference on Green Computing and Communications*, pp. 200–205, Aug. 2011.

[92] J. Ziv, "Compression, tests for randomness and estimating the statistical model of an individual sequence," *Sequences*, 1990.

[93] B. Schneier, *Applied cryptography: protocols, algorithms, and source code in C*, 2nd ed. New York: John Wiley & Sons. Inc., 1995.

[94] M. Mahoney, "Data Compression Explained," *Data Compression Explained - Dell Inc.*, 2012. [Online]. Available: http://mattmahoney.net/dc/dce.html.

[95] E. E. Eiland and L. M. Liebrock, "An Application of Information Theory to Intrusion Detection," *Fourth IEEE International Workshop on Information Assurance, 2006. IWIA 2006.*, 2006.

[96] A. Rukhin, J. Soto, and J. Nechvatal, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," *NIST Special Publication (2010)*, vol. 22, no. April, 2010.

[97] P. Deutsch, "RFC 1951 - DEFLATE Compressed Data Format Specification version 1 . 3 IESG," *IETF*, vol. RFC 1951, pp. 1–15, 1996.

[98] Microsoft, "Cryptography, Crypto API and CAPICOM," *Windows Dev Centre*, 2013. [Online]. Available: http://msdn.microsoft.com/en-us/library/windows/desktop/aa380251(v=vs.85).aspx.

[99] D. Kumar, D. Kashyap, K. K. Mishra, and a. K. Misra, "Security Vs cost: An issue of multi-objective optimization for choosing PGP algorithms," *2010 International Conference on Computer and Communication Technology (ICCCT)*, vol. 1, pp. 532–535, Sep. 2010.

[100] Schnaader, "Compression Problem," 2012. [Online]. Available: http://encode.ru/threads/1614-Compression-Problem.

[101] RapidMiner, "RapidMiner Data Mining Software," 2016. [Online]. Available: http://rapidminer.com/products/studio/. [Accessed: 15-Jan-2016].

[102] C.-C. Chang and C.-J. Lin, "LIBSVM : a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, p. 27:1-27:27, 2011.

[103] D. C. LeBlanc, *Statistics: Concepts and Applications for Science Part 2*. Boston: Jones & Bartlett, 2004.

[104] M. Johnson, P. Ishwar, V. Prabhakaran, D. Schonberg, and K. Ramchandran, "On compressing encrypted data," *IEEE Transactions on Signal Processing*, vol. 52, no. 10, pp. 2992–3006, 2004.

[105] M. Sys, Z. Riha, and V. Matyas, "NIST Statistical Test Suite – result interpretation and optimization," in *SantaCrypt 2015*, 2016, no. July.

[106] P. Deutsch and J.-L. Gailly, "RFC 1950 - ZLIB Compressed Data Format," *IETF*, vol. RFC 1950, pp. 1–10, 1996.

[107] V. Roussev, "Data Fingerprinting with Similarity Digests," in *Advances in Digital*

Forensics VI: Sixth IFIP WG 11.9 International Conference on Digital Forensics, Hong Kong, China, January 4-6, 2010, Revised Selected Papers, K.-P. Chow and S. Shenoi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 207–226.

[108] F. Breitinger and H. Baier, "Similarity Preserving Hashing : Eligible Properties and a New Algorithm MRSH-v2," *Digital Forensics and Cyber Crime*, pp. 167–182, 2013.

[109] V. B. Gupta and F. Breitinger, "How Cuckoo Filter Can Improve Existing Approximate Matching Techniques," vol. 2, pp. 39–52, 2015.

[110] B. Fan, D. G. Andersen, and K. Michael, "Cuckoo Filter: Practically Better Than Bloom," in *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, 2014, pp. 75–88.

[111] B. Y. D. R. Barr and A. H. D. R. H. Shu, "A note on Kuiper's V," pp. 663–664, 1973.

[112] Y. Wu, J. P. Noonan, and S. Agaian, "A novel information entropy based randomness test for image encryption," *2011 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2676–2680, Oct. 2011.

[113] R. Lyda and J. Hamrock, "Using Entropy Analysis to Find Encrypted and Packed Malware," *IEEE Security and Privacy*, vol. 5, no. 2, pp. 40–45, 2007.

[114] J. Goubault-larrecq and J. Olivain, "Detecting Subverted Cryptographic Protocols by Entropy Checking," Centre National De La Recherche Scientifique, 2006.

[115] H. T. Teng, "Progress In Electromagnetics Research, PIER 104, 221–237, 2010," *Progress in Electromagnetics Research*, pp. 221–237, 2010.

[116] K. Yim, T. Miller, and L. Faulkner, "Chemical characterization via fluorescence spectral files and data compression by Fourier transformation," *Analytical Chemistry*, vol. 49, no. 13, pp. 2069–2074, 1977.

[117] D. L. Donoho, M. Vetterli, R. A. Devore, I. Daubechies, and S. Member, "Data Compression and Harmonic Analysis," vol. 44, no. 6, pp. 2435–2476, 1998.

[118] S. W. Smith, "The Scientist and Engineers Guide to Digital Signal Processing," in *The Scientist and Engineers Guide to Digital Signal Processing*, New York: Springer-Verlag, 2004.

[119] O. A. H. Reyna, "Cryptographic Implementations Analysis Toolkit," vol. 2, no. September, 2008.

[120] G. N. Srinivasan and G. Shobha, "Statistical Texture Analysis," *Proceedings of the World Academy of Science, Engineering and Technology*, vol. 36, no. December, pp. 1264–1269, 2008.

[121] K. Sandau, "A note on fractal sets and the measurement of fractal dimension," *Physica A*, vol. 233, pp. 1–18, 1996.

[122] C. Corbit and D. Garbary, "Fractal Dimension as a quantitative measure of complexity in plant development," *Proceedings: Biological Sciences*, vol. 262, no. 1363, 1995.

[123] S. Basu and E. Foufoula-Georgiou, "Detection of nonlinearity and chaoticity in time series using the transportation distance function," *Physics Letters A*, vol. 301, no. September, pp. 413–423, 2002.

[124] A. N. Kolmogorov and V. A. Uspenskii, "Algorithms and randomness," *Theory of Probability and Its Applications*, vol. 32, no. 3, pp. 389–412, 1987.

[125] M. Borowska, E. Oczeretko, A. Mazurek, A. Kitlas, and P. Kuc, "Application of the Lempel-Ziv complexity measure to the analysis of biosignals and medical images,"

Roczniki Akademii Medycznej w Białymstoku, vol. 50, pp. 1–30, 2005.

[126] B. Li, Y. Li, and H. He, "LZ Complexity Distance of DNA Sequences and Its Application in Phylogenetic Tree Reconstruction," *Genomics Proteomics Bioinformatics*, vol. 3, no. 4, 2005.

[127] R. Begleiter, "On Prediction Using Variable Order Markov Models," vol. 22, pp. 385–421, 2004.

[128] R. Williams, "Serial Correlation (Very Brief Overview)," 2015. [Online]. Available: https://www3.nd.edu/~rwilliam/stats2/l26.pdf. [Accessed: 21-Dec-2015].

[129] M. A. L. I. Soliman and A. M. R. El-helw, "Network Intrusion Detection System Using Bloom Filters."

[130] S. L. Garfinkel, "Automated Media Exploitation Research," 2011. [Online]. Available: http://simson.net/ref/2011/2011-09-14 UK MET Briefing.pdf. [Accessed: 23-Jul-2014].

[131] E. Bahel, "Rock-paper-scissors and cycle-based games," *Economics Letters*, vol. 115, no. 3, pp. 401–403, 2012.

[132] International Electrotechnical Commission, "International Electrotechnical Commision," 2016. [Online]. Available: http://www.iec.ch/. [Accessed: 08-Jul-2015].

[133] A. Karpathy and (Stanford University Computer Science Department), "Convolutional Neural Networks for Visual Recognition." [Online]. Available: http://cs231n.github.io/neural-networks-1/. [Accessed: 14-Jul-2016].

Appendix A Failed Discriminators

Many disciplines were surveyed in the search for methods to classify high entropy fragments. The basic premise was that encrypted fragments were more complex than compressed fragments. Compression methods were expected to balance compression effectiveness against speed and therefore leave some redundancy within the file. The search for a method to discriminate between these was wide ranging.

Initially the usual statistical analysis of the byte frequency distribution (BFD) was used. None of the standard descriptive statistics provided any differentiation. Statistics used included mean, variance, skewness and kurtosis. In addition a truly random sequence should have a uniform distribution with the probability of any byte value being equal. If one particular byte value was more probable than others then the sequence is not random. The chi square goodness of fit test was used to check the fit of the byte frequency distribution to a uniform distribution. Kuipers test [111] was also used as a test to disprove the hypothesis that the BFD came from a uniform distribution. Fragment entropy [27], [112]–[115] was calculated but failed to give any discrimination.

The Fast Fourier Transform [116]–[119] was used to test for any periodicity in the data but none appeared consistently and did not provide discrimination.

In the field of genetics it was found that the fractal dimension of a set was used as a discriminator. The fractal dimension is, among other things, a measure of the set complexity. It was found that the fractal dimension of a set could be used as a discriminator for texture [51, p.222], [55]. In addition, fractal dimension can be used as a measure of complexity [121], [115], [122]. No discrimination was found between types. Lacunarity is a measure of the texture of a fractal and can be used to discriminate between sets of similar fractal dimension [123]. Calculation of Lacunarity was implemented with no greater success.

Other methods of measuring complexity were tried. but provided no discrimination when implemented. An approximation to Kolmogorov complexity [124] was tried as one of the standard statistical techniques but provided no discrimination.

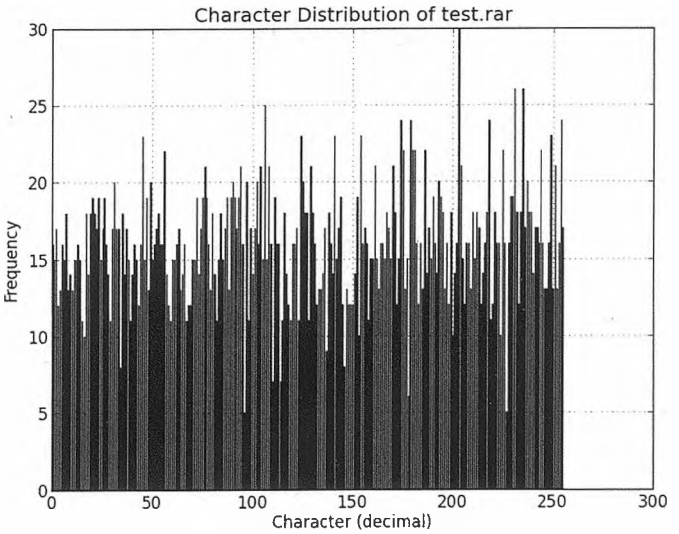


Figure 14 - Compressed file visualisation

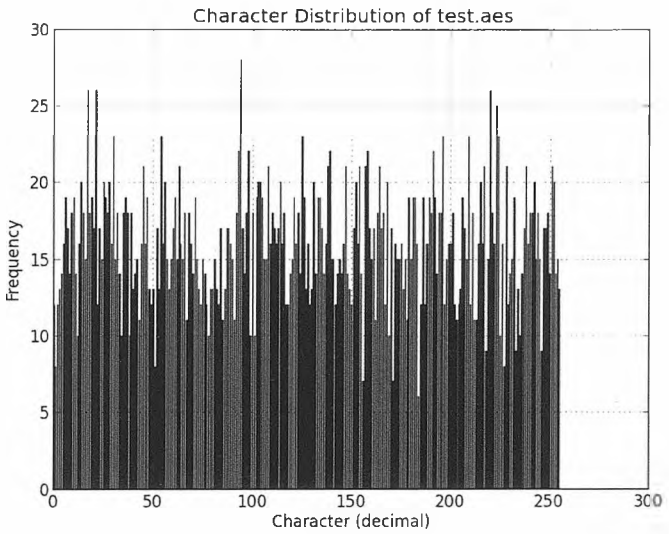


Figure 15 - Encrypted file visualisation

LZ complexity is used in the analysis of bio signals and medical imaging [125]. The complexity of the sequences was tested using both LZ Complexity [81], [126], [127] and an approximation to Kolmogorov complexity [124] but these failed to provide discrimination.

The serial correlation coefficient [26], [128] was calculated. This tests how each byte in the fragment depends on the previous byte. The value should be close to zero for a random file. It was found to provide no discrimination between types.

Another test was devised to check for predictability. A first order prediction model as used in Prediction by Partial Matching (PPM) compression was coded. If the fragment is random, then the predictions of this model should be correct about 1/256 of the time. This failed to provide discrimination between high entropy fragment types.

A k-NN implementation using the LZ complexity distance [126] as a metric was used to measure the distance of fragments from a random string generated from atmospheric noise at Random.Org. It was hypothesised that the compressed fragment distance from the random string would be greater than from the encrypted string. This turned out not to be true. The cosine similarity metric [25], [129], [130] and normalised compression distance [80] were also applied as metrics with no better effect.

Garfinkel et al. [18] used bit shifting and auto-correlation on fragments and the cosine similarity between bit shifted blocks was used to detect Huffman coding. This was implemented with no success at discriminating compressed and encrypted fragments. The algorithm given in [18] is not well specified and each possible interpretation of it was implemented.

A bijective compressor has the property that any possible sequence of bytes could be a possible output [94]. This means that any possible sequence can be uncompressed. The sequence produced is likely to be meaningless if it was not originally produced by the compressor, but nevertheless the decompression will not fail. It was decided to use a bijective decompressor on the fragments to see if there would be a significant difference between the decompressed size of encrypted and compressed fragments. No significant differentiation was achieved.

In the classic game of rock-paper-scissors the optimal strategy is a completely random choice [131]. It was hypothesised that a completely random sequence would 'defeat' a non-random

sequence. To test this an application to convert the binary sequences in the 4 KiB sample clusters into 'trits' (base 3 digit analogous to a base 2 bit) was implemented. A random sequence from a random string generated from atmospheric noise at Random.Org was converted to trits and saved as the master cluster for testing other clusters against. Each sample cluster was converted to trits. Each trit in the sequence was played against the corresponding trit in the master sequence. The number of wins for the master cluster was recorded. If this was high then it was hypothesised that the fragment being tested was not random. The method failed to discriminate well between compressed and encrypted fragments.

Appendix B Research Ethics and Integrity

This research was adhered to the principles set out in the UK Research Integrity Office Code of Practice[2]. The associated checklist has been completed in Appendix C.

All software development has been carried out in accordance with the British Computer Society Code of Conduct[3].

Where possible, data has been utilised from publically available digital corpora which have been checked for privacy and copyright concerns. When such data is insufficient, then raw random data from random.org has been used to create files of an appropriate size. This data is generated from atmospheric noise and as such contains no data which could lead to the identification of any individual.

If any privileged or private information on individuals is collected in the course of this research then in accordance with Edinburgh Napier University Code of Practice on Research Integrity[4], institutional approval will be sought. Any such data will be anonymised in accordance with the guidelines set out in Appendix 2, Annexes 1 and 2 of the UK Information Commissioner's Office Code on Anonymisation[5]. No results or statistics will be published or made available that can be used in any way to identify a data subject.

If any software created is trialled by an external agency, then the test data being used will be under the control of the agency and at no time will be transferred to any University or personal computer equipment.

An understanding has been reached with Edinburgh Napier University relating to intellectual property, publication and authorship relating to this research.

---

[2] UK Research Integrity Office Code of Practice http://www.ukrio.org/publications/code-of-practice-for-research/2-0-principles/

[3] CODE OF CONDUCT FOR BCS MEMBERS available at http://www.bcs.org/upload/pdf/conduct.pdf [Accessed 4/5/2014]

[4] http://staff.napier.ac.uk/services/vice-principal-academic/research/researchpractice/Pages/CodeofConduct.aspx [Accessed 5/9/2014]

[5] The UK Information Commissioner's Office 'Anonymisation: managing data protection risk Code of Practice' (2012). Available at http://www.ico.gov.uk/for_organisations/data_protection/topic_guides/~/media/documents/library/Data_Protection/Practical_application/anonymisation_code.ashx [Accessed 5/9/2014]

Appendix C - UK Research Integrity Office Code of Practice Checklist

Recommended checklist for researchers

The Checklist lists the key points of good practice in research for a research project and is applicable to all subject areas.

Before conducting your research, and bearing in mind that, subject to legal and ethical requirements, roles and contributions may change during the time span of the research:

1. Does the proposed research address pertinent question(s), and is it designed either to add to existing knowledge about the subject in question or to develop methods for research into it? ✓
2. Is your research design appropriate for the question(s) being asked? ✓
3. Will you have access to all necessary skills and resources to conduct the research? ✓
4. Have you conducted a risk assessment to determine:
   a. whether there are any ethical issues and whether ethics review is required; ✓
   b. the potential for risks to the organisation, the research, or the health, safety and well-being of researchers and research participants; and ✓
   c. what legal requirements govern the research? ✓
5. Will your research comply with all legal and ethical requirements and other applicable guidelines, including those from other organisations and/or countries if relevant? ✓
6. Will your research comply with all requirements of legislation and good practice relating to health and safety? ✓
7. Has your research undergone any necessary ethics review (see 4(a) above), especially if it involves animals, human participants, human material or personal data? ✓
8. Will your research comply with any monitoring and audit requirements? ✓
9. Are you in compliance with any contracts and financial guidelines relating to the project? ✓
10. Have you reached an agreement relating to intellectual property, publication and authorship? ✓
11. Have you reached an agreement relating to collaborative working, if applicable? ✓
12. Have you agreed the roles of researchers and responsibilities for management and supervision? ✓
13. Have all conflicts of interest relating to your research been identified, declared and addressed? ✓
14. Are you aware of the guidance from all applicable organisations on misconduct in research? ✓

When conducting your research:

1. Are you following the agreed research design for the project? ✓
2. Have any changes to the agreed research design been reviewed and approved if applicable? ✓
3. Are you following best practice for the collection, storage and management of data? ✓
4. Are agreed roles and responsibilities for management and supervision being fulfilled? ✓
5. Is your research complying with any monitoring and audit requirements? ✓

When finishing your research:

1. Will your research and its findings be reported accurately, honestly and within a reasonable time frame? ✓
2. Will all contributions to the research be acknowledged? ✓
3. Are agreements relating to intellectual property, publication and authorship being complied with? ✓
4. Will research data be retained in a secure and accessible form and for the required duration? ✓
5. Will your research comply with all legal, ethical and contractual requirements? ✓

Appendix D Terminology

Throughout this research distinction has been made between the base 10 and base 2 notation used by storage device and memory manufacturers respectively and as defined by the International Electrotechnical Commission (2016) [132].
For storage devices, 1 GB = 10^9 and 1 MB = 10^6.
For memory, 1 GiB = 2^{30} and 1 MiB = 2^{20}.
Also GB is referred to as Gigabyte and GiB as Gibibyte.