A MAJOR REPORT ON

RETRIVAL OF MOBILE DATA WITHOUT USING THE STORAGE META-DATA

Submitted in partial fulfillment of the Requirement for the award of the degree of

MASTER OF TECHNOLOGY

(INFORMATION SYSTEMS)

Submitted by

MOHAMMED MUZZAMIL.H

(04/IS/2K10)

Under the Guidance of

Ms. RITU AGARWAL

(Asst. Professor)

Dept. of Information Technology

DEPARTMENT OF INFORMATION TECHNOLOGY

DELHI TECHNOLOGICAL UNIVERSITY

BAWANA ROAD, DELHI-110042

2010-12

CERTIFICATE

This is to certify that **Mr. Mohammed Muzzamil. H** (04/IS/2k10) has carried out the major project titled “Retrieval of mobile data without using the storage meta-data” as a partial requirement for the award of Master of Technology degree in Information Systems by Delhi Technological University.

The major project is a bonafide piece of work carried out and completed under my supervision and guidance during the academic session 2010-2012.The matter contained in this report has not been submitted elsewhere for the award of any other degree.

(Project Guide)

Asst. Professor

Ms. RITU AGARWAL

Department of Information Technology

Delhi Technological University

Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

I express my gratitude to my major project guide **Ms. Ritu Agarwal**, Asst. Professor, IT Dept., Delhi Technological University, for the valuable support and guidance he provided in making this major project. It is my pleasure to record my sincere thanks to my respected guide for her constructive criticism and insight without which the project would not have shaped as it has.

I would like to thank **Prof. O.P. Verma**, HOD, IT Dept., Delhi Technological University, for his useful insights and guidance towards the project. His suggestion and advice proved very valuable throughout. I am thankful to all teaching or non-teaching staff at D.T.U., and my fellows, who have helped or indirectly in completion of this thesis report.

MOHAMMED MUZZAMIL. H
M.TECH (INFORMATION SYSTEM)
ROLL NO. 04/IS/2K10

ABSTRACT

Normally recovery software is using file system information to recover the data which have been deleted. File system information are file tables, meta-data etc., So, when a disk is formatted then all the information and the metadata will also be lost. At this scenario, normal recovery techniques are not useless. For this comes, File carving, which is the process of recovering files from a disk without the help of a file system information. Even without the knowledge of the memory structure, we can carve out the data from a formatted disk. In forensic, this is a very helpful technique in searching the hidden and recently formatted disk contents. We already have many tools available for file carving, like foremost, scalpel etc., File carving is most often used to recover files from the unallocated space in a drive. Unallocated space refers to the area of the drive which is no longer holding any file information as indicated by the file system structures like the file table. In the case of damaged or missing file system structures; this may involve the whole drive. While deleting a file from the disk, the will not be deleted directly from the physical memory, the logical memory containing the information regarding the starting address of the file is deleted or made to zero. Thus the original data in the physical memory will be available, unless it is overwritten by another file. So we use this technique to find the lost data in mobile devices. Mobile devices are becoming very important mode of communication and becoming evidence for a variety of crimes. The information like SMS, MMS, images and other data from the mobile phone are very important to identify the criminals. So carving the data from the mobile devices will be terribly useful for the upcoming years.

CONTENTS

LIST OF FIGURES

Chapter 1 INTRODUCTION OF FILE CARVING

File carving is the technique of recovering the information from any file system which has been corrupted or whose data has been deleted. File carving is a powerful technique for analyzing the physical memory dumps when memory structures are unknown.

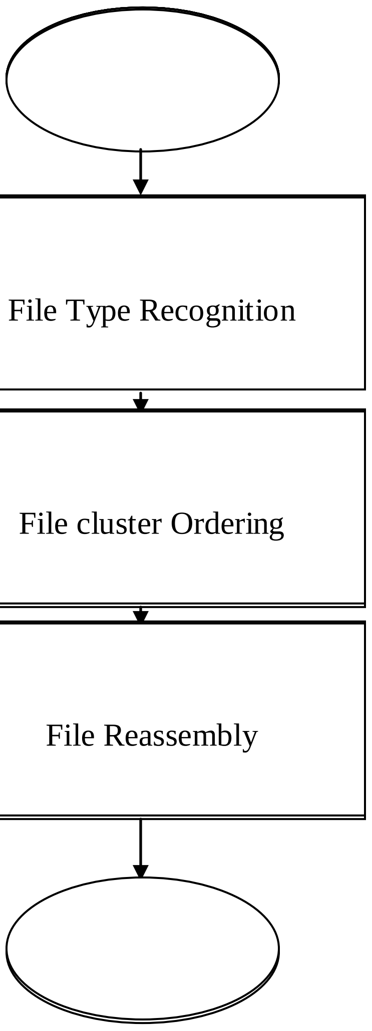


Figure 1.1 : The General steps of file carving

1.1 **TRADITIONAL RECOVERY TECHNIQUE**

Traditional data recovery techniques rely on file system structures like file tables to recover data that has been deleted. This is because most file systems do not touch the physical location of the files during deletion; they simply mark the location as being available for storing information.

After deletion, the entry of the file in the file table may still be present and the information linking the clusters to the file deleted may also still be present, and as a result, such a file can be easily recovered. Another advantage of accessing file system structures is to also be able to identify and quickly extract existing undeleted data, therefore, only the areas in the disk that are considered unallocated, need to be parsed. However, when the file system structures are not present, corrupt, or have been deliberately removed, the data, while present cannot be accessed via traditional means.

Once it became clear that traditional recovery techniques may fail on data sets, additional techniques needed to be introduced to recover forensically important user files. Some examples of these files are Microsoft Office documents, digital pictures, and e-mails. More often than not, the files of importance for forensic recovery are those that are created and modified by the users. Operating system and application files can be re - installed, however, user files not backed up and deleted require recovery. File carving is a forensics technique that recovers files based merely on file structure and content and without any matching file system meta-data. File carving is most often used to recover files from the unallocated space in a drive. Unallocated space refers to the area of the drive which is no longer holding any file information as indicated by the file system structures like the file table. In the case of damaged or missing file system structures. This may involve the whole drive.

1.2 **CURRENT DAY SENARIO**

The current day forensic department is using this forensic tool to find the criminals. Nowadays each and every person is having his or her own personal PC and mobile phone. These, electronic devices has become very normal in every persons' life. So, whenever a crime has occurred, these devices are becoming the first line of evidences. For collecting evidence from these electronic devices, PCs, mobile phones, CDs, floppies, and other electronic devices, we need some special type of tools to recover data. The file carving came for this purpose, to recover the deleted or corrupted files from the hard disk or any storage media. Pervious we can only recover the files from the file table alone. But now we can even recover files without the help of file tables or any other metadata.

1.3 **FILE CARVING AN IMPORTANT TOOL IN FORENSIC**

Data recovery is a key component of the disaster recovery, forensics, and e-discovery markets. Digital data recovery can consist of both software and hardware techniques. Hardware techniques are often used to extract data from corrupted or physically damaged disks. Once the data has been extracted, software recovery techniques are often required to order and make sense of the data. In this article, we will be solely discussing software techniques for recovery of data with a focus on digital forensics. We will begin by providing a quick overview of traditional data recovery techniques and then describe the problems involved with such techniques. We then introduce the techniques involved in file carving.

File system structures are not used during the process. File carving is a powerful technique for recovering files and fragments of files when directory entries are corrupt or missing. The block of data is searched block by block for residual data matching the file type-specific header and footer values. Carving is also especially useful in criminal cases where the use of carving techniques can recover evidence. In certain cases related to child pornography, law enforcement agents are often able to recover more images from the suspect’s hard disks by using carving techniques. Another example is the hard disks and removable storage media US Navy Seals took from Osama Bin Laden’s campus during their raid. Forensic experts used file carving techniques to squeeze every bit of information out of this media.

1.4 **FILE RECOVERY VS CARVING**

There is a big difference between file recovery techniques and carving. File recovery techniques make use of the file system information that remains after deletion of a file. By using this information, many files can be recovered. For this technique to work, the file system information needs to be correct. If not, the files can’t be recovered. If a system is formatted, the file recovery techniques will not work either.

Chapter 2 EVOLUTION OF FILE CARVING

The file carving is the techniques to recover the files from the electronic storage media without using the storage metadata.

2.1 **FILE SYSTEMS AND FRAGMENTATION**

Before describing what a file system is and how files are stored and deleted, we want to briefly introduce the physical blocks on a disk that are used to store data. Most disks are described in terms of data blocks called sectors and clusters. The cluster size is a function of the sector size (1). The disk size or capacity is calculated by multiplying the cluster size by the number of clusters (2). It is very important to note that a cluster (not a sector) represents the smallest unit of storage that is addressable (can be written to or read). Therefore, files are typically stored in terms of clusters and not sectors. Typical values of clusters range from 512–32,000 B. Cluster sizes are normally multiples of 512 B.

Cluster Size = Sector Size X Sectors per Cluster. (1)

Disk Size = Cluster Size X Count. (2)

Let us consider the file systems like FAT and NTFS separately, and we will see how these file systems work when the date is updated and deleted.

2.2 **FAT FILE SYSTEM**

2.2.1 FILE ALLOCATION

File allocation in FAT-32 is a two step process. First, the file is assigned to a directory/folder. If the assigned directory/folder is the root of the drive, then the file name is stored in the root directory entry, if not, one must parse the root directory entry to find the directory that is in the path of the final directory to which the file is assigned to. Along with the file name, the file entry contains the starting cluster number of the file. The starting cluster number represents the cluster where the file's contents begin. In reality, additional information, including access, creation, and modification time stamps along with long file names are also stored in the directory entry. To retrieve the file, the file system looks at the starting cluster and then goes to the starting cluster index in the file allocation table. The FAT can be considered to be an array of cluster numbers pointing to the next cluster of a file. Therefore, when a file is retrieved, the file system goes to the starting cluster index in the FAT, and gets the next cluster number for that file. This cluster in turn can point to another cluster. This process is repeated until a cluster has the hex value "FF" indicating end of file (EOF). The file system is then able to retrieve the file by reading the actual clusters on the disk.

An example of this is creating a file called "recovery.txt," which requires five clusters to store its contents. The file system may find that the cluster number 300 is where the file should start to be stored. As shown in Figure 2.1, a root entry for "recovery.txt" is made indicating the starting cluster of the file shown as cluster number 300. The file system stores the remaining clusters in cluster numbers 301, 302, 305, and 306 and makes the appropriate changes to the FAT. Note that clusters 303 and 304 are used for storage of another file "hello.txt."

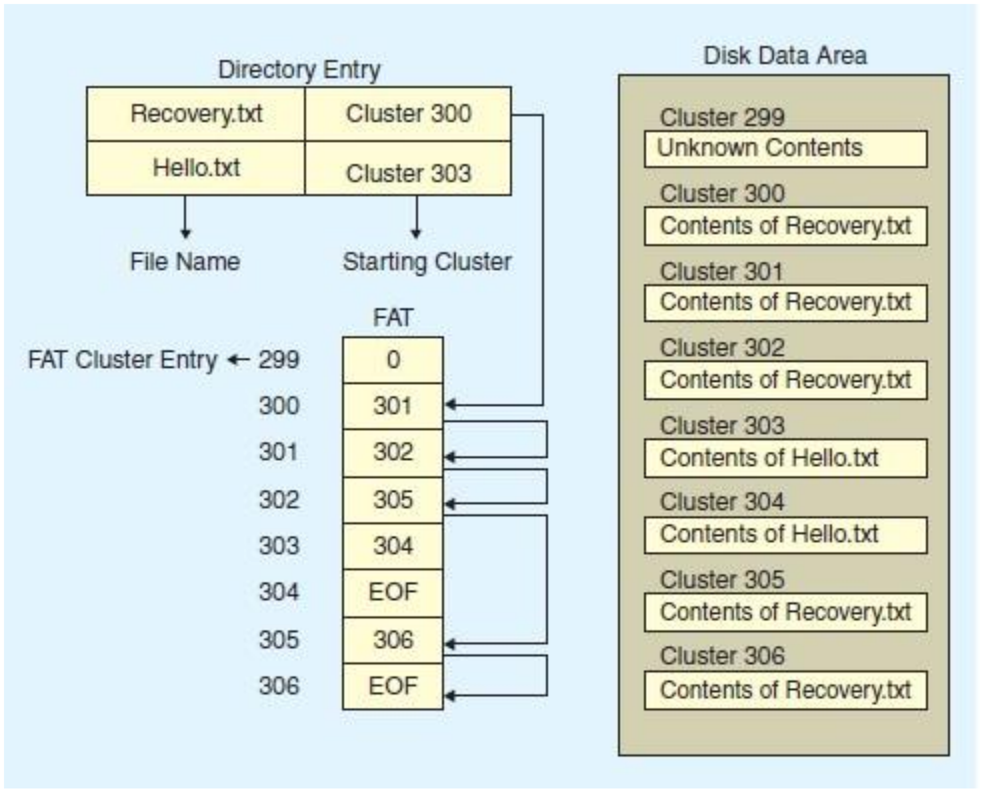


Figure 2.1 : **File Allocation**

2.2.2 DELETION

When deleting a file, the file system does not actually go to the clusters of the file and zero them out. Instead, all the file system does is go to the cluster links in the FAT and set them to hex value “00” that indicates that the clusters have been unallocated. The actual contents of the drive are still present after the deletion.

Interestingly, while the FAT cluster entries for each of the clusters of the deleted file are indicated to be unallocated, the directory entry of the file still points to the starting cluster with a special character being used to mark that this entry represents a deleted file.

Figure 2 shows what happens to the FAT when recovery.txt is deleted. The clusters represented by the file in the FAT are changed to zero to indicate availability, however, the actual cluster contents of recovery.txt have not been removed as yet.

The first byte of the name is also changed in the file entry directory to represent that the file was deleted.

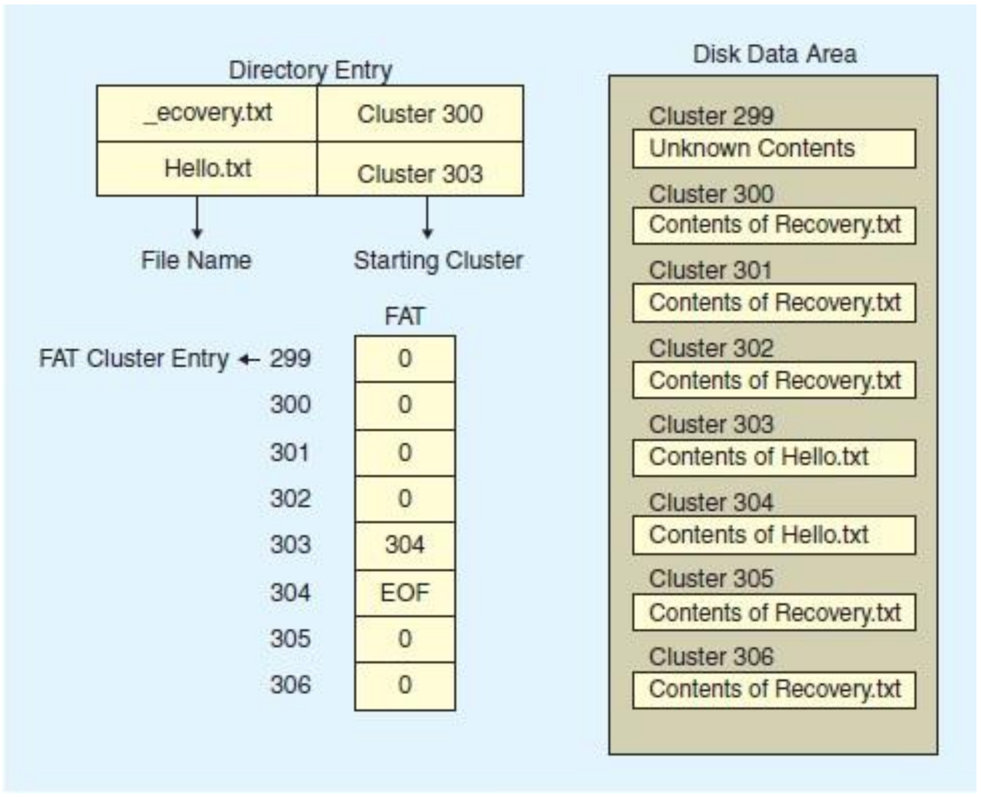


Figure 2.2 **: File Deletion**

2.2.3 RECOVERY

Since the cluster sequence in the FAT is set to hex value “00” (unallocated) on a file’s deletion, recovery programs have to check that the file starting cluster is not in use, and then assume that the rest of the file is stored in sequential clusters. Once a file is deleted, there is absolutely no way to determine the exact sequence of clusters to reconstruct the file using the file system meta-data alone. In the example using recovery.txt, traditional file recovery programs will find the entry for “_ecovery.txt” and see that the starting cluster for the deleted file was 300. Some recovery programs will only recover a partial file (clusters 300, 301, 302), while the smarter file recovery programs are able to recover the whole file because they see that clusters 303 and 304 are in use by another file, and that clusters 305 and 306 are available, so they will merge the unallocated clusters only. However, what happens if the file is broken-up into more than one piece or there are unallocated clusters belonging to another file in between? For example, if hello.txt and recovery.txt are deleted, then file recovery becomes much more difficult without analyzing the contents or structure of the files.

2.3 **NTFS File System**

FAT-32, while an improvement over earlier versions of the FAT, does not support files of greater than 4 GB. Another limitation is the time taken to determine the free space for the FAT increases with an increase in the number of clusters in the system. Finally, all versions of the FAT do not provide enhanced security features and support for meta-data. As a result, Microsoft introduced NTFS with Windows NT in 1993, as the preferred file system for its modern operating systems.

2.4 **WEAR-LEVELING ALGORITHMS IN NEXT GENERATION DEVICES**

2.4.1 FILE ALLOCATION

In NTFS, file information is stored in a B-Tree [1] , however, unlike the FAT, NTFS has a separate structure called a bitmap (BMP) to represent the allocation status of clusters. In the BMP file ($bitmap) representing clusters, each cluster is assigned a value of one if it belongs to an existing file, and a zero otherwise. By parsing the BMP, NTFS is able to quickly determine the best place to put a file in to prevent fragmentation. In addition, just as in the FAT, NTFS stores the file cluster links.

2.4.2 DELETION

When a file is deleted, its associated clusters in the BMP are set to zero. Again, just as in the FAT, the actual data is not deleted when deleting files. Unlike the FAT, the file cluster links are not deleted either so if the deleted file entry is present, then the original cluster links are also still present. This makes recovery much easier in NTFS, as the full cluster link representing the file is still present. The clusters are simply shown as being de-allocated in the $bitmap file.

2.4.3 RECOVERY

In NTFS, file recovery is quite straightforward when the file entry is still present. This is because the file's cluster numbers should also be present, and all that needs to be done is to verify that the file's clusters have not been overwritten, or are not allocated to another file. In the case where the file system is damaged or the deleted file's entries removed, file system meta-data cannot be used for recovery.

2.4.4 FRAGMENTATION

As files are added, modified, and deleted, most file systems get fragmented. File fragmentation is said to occur when a file is not stored in the correct sequence on consecutive clusters on disk. In other words, if a file is fragmented, the sequence of clusters from the start of a file to the end of the file will result in an incorrect reconstruction of the file. The example of "recovery.txt" provided in Fig. 2.1 provides a simplified example of a fragmented file. In the figure, the "recovery.txt" has been broken into two fragments. The first fragment starts at cluster 300 and ends at cluster 302. The second fragment starts at cluster 305 and ends at cluster 306. This file is considered to be bi-fragmented as it has only two fragments. Garfinkel showed that bi-fragmentation (two fragments only) is the most common type of fragmentation [2] , however, files fragmented into three or more pieces are not uncommon. Garfinkel's fragmentation statistics [2] come from identifying over 350 disks containing FAT, NTFS, and Unix file system (UFS). He shows that while fragmentation in a typical disk is low, the fragmentation rate of forensically important files (user files) likes that of e-mail, jpeg, and Microsoft Word are high. The fragmentation rate of jpegs was found to be 16%, Microsoft Word documents had 17% fragmentation, audio video interleave (AVI) (movie format) had a 22% fragmentation rate, and personal information store (PST) files (MS-Outlook) had a whopping 58% fragmentation rate.

Fragmentation typically occurs due to low disk space, appending/editing files, wear-leveling algorithms in next generation devices, and file systems.

2.5 **APPENDING/EDITING FILES**

If a file is saved on disk and then additional files are also saved starting after the cluster that the original file ended at, fragmentation may occur if the original file is then appended to (and increases in size larger than the cluster size). Some file systems, like the Amiga Smart File System [3], may attempt to move the whole file in such scenarios. Some other file systems like UFS attempt to provide "extents" that are attempts to pre-allocate longer chunks in anticipation of appending [4]. Another technique, called delayed allocation, used in file systems like XFS [5] and ZFS, reserve file system clusters but attempt to delay the physical allocation of the clusters until the operating system forces a flushing of the contents. However, while some of these techniques are able to reduce fragmentation, they are unable to eliminate fragmentation completely. Constant editing, deletion, and additions of e- mails are the primary reasons why PST files (Microsoft Outlook files) are highly fragmented (58% of pst files analyzed in the wild [2]).

2.6 **WEAR-LEVELING ALGORITHMS IN NEXT GENERATION DEVICES**

SSDs currently utilize proprietary wear-leveling algorithms to store data on the disk [6] in order to enhance its lifetime. Wear leveling is an algorithm by which the controller in the storage device remaps logical block addresses to different physical block addresses. If the controller gets damaged or corrupted, then any data extracted will be inherently fragmented, with no easy way of determining the correct sequence of clusters to recover files.

2.7**FILE STRUCTURE-BASED CARVERS**

File carving was born due to the problems inherent with recovery from file system meta-data alone. File carving does not use the file system information directly to recover files. Instead, it uses knowledge of the structure of files. More advanced carvers not only use knowledge of the structure of files but also use the contents of individual files to recover data.

Given a digital storage device, a file carver may or may not recognize the file system being used, and it may or may not trust the information in the file system to be accurate. As a result, it is up to the carver to determine the "unallocated" clusters in the disk to carve from. This may involve all clusters in the disk. To reduce the amount of a disk to recover from, a number of forensic applications are able to identify a large number of files that are common to operating systems and applications based on their MD5 Hash and keywords. Both Encase and Forensic Toolkit (FTK), the two leading commercial disk forensic software providers, provide this option to quickly eliminate common and well-known files. The first generation of file carvers used “magic numbers,” or to be more precise, byte sequences at prescribed offsets to identify and recover files. File carving techniques were first used for files that contain a “header” and “footer.” A header identifies the starting bytes of a file and the footer identifies the ending bytes of the file. Headers are used by some operating systems to determine which application to open a file with. In regard to jpegs, starting clusters must begin with the hex sequence FFD8, and footer is a cluster containing the hex sequence FFD9. A file carver relying on structure alone will attempt to merge and return all unallocated clusters between the header and footer.

There are many file types that may not contain footers but instead contain other information like the file size. An example of this is the Windows BMP file that contains the size of the file in bytes. Again, traditional file carvers would recover files by identifying the header and then merging as many unallocated sequential clusters following the header as required to equal the file size.

Foremost [7], developed by the United States Air Force Office of Special Investigations, was one of the first file carvers that implemented sequential header to footer carving and also implemented carving based on a header and the size of the file. Golden et al. [8] then proposed Scalpel, which was built on the foremost engine, but greatly improved its performance and memory usage. The problem with the first generation of file structure-based carvers was that they simply extract data between a known header and footer (or ending point determined by size), with the assumption being that the file is not fragmented and there is no missing information between the header and the footer. As a result, these file carvers frequently provide results that have “garbage” in the middle. From the image, it can be seen that a large number of clusters were decoded that did not belong to the image before the decoding completely failed. The image belongs to the DFRWS 2006 [9] file carving challenge, which does not contain any file system meta-data. The DFRWS challenge test-sets [9], [10] were specifically created to test and stimulate research in the area of fragmented file carving.

Chapter 3 FILE SYSTEM ANALYSIS

File system analysis examines data in a volume (i.e., a partition or disk) and interprets them as a file system. There are many end results from this process, but examples include listing the files in a directory, recovering deleted content, and viewing the contents of a sector. Recall that analyzing the contents of a file is application-level analysis and is not covered in this book. In this chapter, we look at the general design of file systems and different analysis techniques. This chapter approaches the topic in an abstract fashion and is not limited to how a specific tool analyzes a file system. Instead, we discuss the analysis in general terms. The remaining nine chapters discuss how specific file systems are designed and what is unique about them with respect to digital investigations.

3.1 **FILE SYSTEM**

The motivation behind a file system is fairly simple: computers need a method for the longterm storage and retrieval of data. File systems provide a mechanism for users to store data in a hierarchy of files and directories. A file system consists of structural and user data that are organized such that the computer knows where to find them. In most cases, the file system is independent from any specific computer.

For an analogy, consider a series of filing cabinets in a doctor's office. The fictitious *National Association of Medical Record Filing Procedures* (NAMRFP) could specify that all patient records must be organized into filing cabinets and sorted by the last name of the patient. The tag that is used to identify the record must be typed in English and have the last name followed by the first name. Any person trained in this procedure would be able to file and retrieve patient records at an office that uses the procedure. It doesn't matter if the office has 100 patients and one filing cabinet or 100,000 patients and 25 filing cabinets. All that matters is that the person recognizes what a filing cabinet is, knows how to open it, and knows how to read and create the tags. If that person visited an office that used the National Association of Medial Record Stacking Procedures method where all records were stacked in a corner, his filing cabinet training would be useless and he would not be able to find the needed records.

File systems are similar to these record-storing procedures. File systems have specific procedures and structures that can be used to store one file on a floppy disk or tens of thousands of files in a storage array. Each file system instance has a unique size, but its underlying structure allows any computer that supports the type of file system to process it. Some data needs internal structure and organization inside the file. This is not unlike physical documents needing structure in the form of sections and chapters. The internal structure of a file is application dependent and outside the scope of this book. This book is concerned about the procedures and techniques needed to obtain the data inside of a file or the data that are not allocated to any file.

3.2 **DATA CATEGORIES**

As we examine each of the different file system types in this part of the book, it will be useful to have a basic reference model so that the different file systems can be more easily compared. Having such a reference model also makes it easier to determine where your evidence may be located. For example, a reference model makes it easier to compare the difference between FAT and Ext3 file systems. For this basic model, we will use five categories: file system, content, metadata, file name, and application. All data in a file system belong to one of the categories based on the role they play in the file system. We will use these categories throughout this book when describing file systems, although some file systems, namely FAT, cannot be applied to this model as easily as others can. The tools in *The Sleuth Kit* (TSK) are based on these same categories.

The *file system* category contains the general file system information. All file systems have a general structure to them, but each instance of a file system is unique because it has a unique size and can be tuned for performance. Data in the file system category may tell you where to find certain data structures and how big a data unit is. You can think of data in this category as a map for this specific file system. The *content* category contains the data that comprise the actual content of a file, which is the reason we have file systems in the first place. Most of the data in a file system belong to this category, and it is typically organized into a collection of standard-sized containers. Each file system assigns a different name to the containers, such as clusters and blocks, and I will use the general term *data units* until we discuss specific file systems.

The *metadata* category contains the data that describe a file; they are data that describe data. This category contains information, such as where the file content is stored, how big the file is, the times and dates when the file was last read from or written to, and access control information. Note that this category does not contain the content of the file, and it may not contain the name of the file. Examples of data structures in this category include FAT directory entries, *NTFS Master File Table* (MFT) entries, and UFS and Ext3 inode structures.

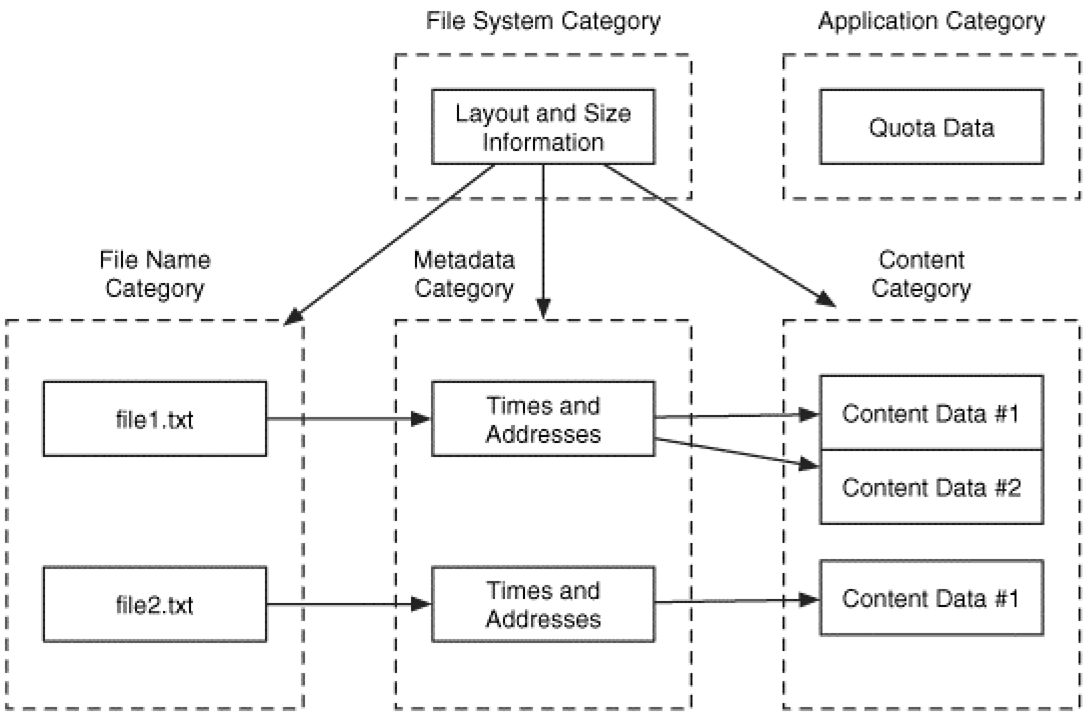


Figure 3.1: Interaction between the five data categories.

The *file name* category, or human interface category, contains the data that assign a name to each file. In most file systems, these data are located in the contents of a directory and are a list of file names with the corresponding metadata address. The file name category is similar to a host name in a network. Network devices communicate with each other using IP addresses, which are difficult for people to remember. When a user enters the host name of a remote computer, the local computer must translate the name to an IP address before communication can start.

The *application* category contains data that provide special features. These data are not needed during the process of reading or writing a file and, in many cases, do not need to be included in the file system specification. These data are included in the specification because it may be more efficient to implement them in the file system instead of in a normal file. Examples of data in this category include user quota statistics and file system journals. These data can be useful during an investigation, but because they are not needed to write and read a file, they could be more easily forged than other data.

3.3 **LOGICAL FILE SYSTEM ADDRESS**

A sector can have multiple addresses, each from a different perspective. In our discussion of Acquisition techniques, we saw that every sector has an address relative to the start of the storage media, which is its physical address. Volume systems create volumes and assign logical volume addresses that are relative to the start of the volume.

File systems use the logical volume addresses but also assign *logical file system addresses* because they group consecutive sectors to form a data unit. In most file systems, every sector in the volume is assigned a logical file system address. An example of a file system that does not assign a logical file system address to every sector is FAT.

Figure 3.2 shows a volume with 17 sectors and their logical volume addresses. Below them are the logical file system addresses. This fictitious file system created data units that were each two sectors, and the file system did not assign addresses until sector 4. This very small file system ends in sector 15, and sector 16 is volume slack.

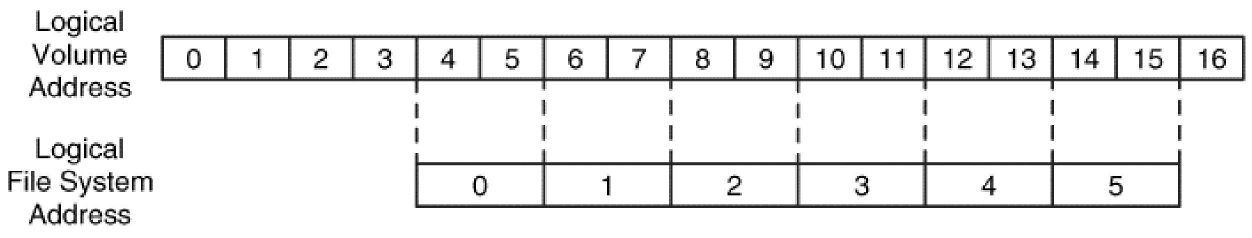


Figure 3.2 **: Logical file system addressing**

3.3.1 ALLOCATION STRATEGY

An OS can use different strategies for allocating data units. Typically, an OS allocates consecutive data units, but that is not always possible. When a file does not have consecutive data units, it is called fragmented.

A *first available* strategy searches for an available data unit starting with the first data unit in the file system. After a data unit has been allocated using the first available strategy and a second data unit is needed, the search starts again at the beginning of the file system. This type of strategy can easily produce fragmented files because the file is not allocated as a whole. For example, consider a theater that uses a first available strategy to assign seats. If a group of four wanted tickets to a show, the box office would start with the front row and scan for available seats. They may get all four seats together or two people may get seats in front and the other two in the back. If someone returns a ticket while the search is underway, the third person may end up getting a seat closer to the front than the first two people did. In the example shown in Figure 8.3, data unit 1 would be allocated next using a first available strategy. An OS that uses first available overwrites deleted data at the beginning of the file system more quickly than other algorithms will. Therefore, if you encounter a system using this algorithm, you will probably have better luck recovering deleted content from the end of the file system.

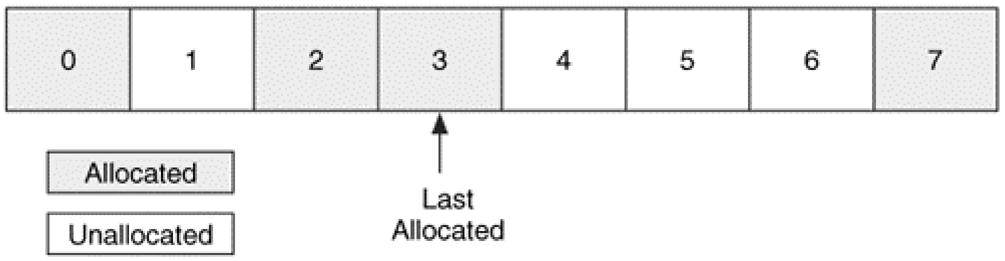


Figure 3.3 : Allocation status of 8 data units

A similar strategy is *next available*, which starts its search with the data unit that was most recently allocated instead of at the beginning. For example, if data unit 3 in Figure 3.3 is allocated, the next search starts at data unit 4 instead of 0. In our theater example, we would start the search from the last seat that was sold instead of starting in the front row. With this algorithm, if a ticket at the front of the theater was returned while the search was underway, it would not be sold until the search reached the last seat. This algorithm is more balanced for data recovery because the data units at the beginning of the file system are not reallocated until the data units at the end have been reallocated.

Another strategy is *best fit*, which searches for consecutive data units that fit the needed amount of data. This works well if it is known how many data units a file will need, but when a file increases in size, the new data units will likely be allocated somewhere else and the file can still become fragmented. If the algorithm cannot find a location for all the data, a first or next available strategy may be used. This is the algorithm that typically occurs when assigning seats in a theater. The empty seats are scanned until enough free consecutive seats can be found for the group. In our Figure 8.3 example, if we had a two-data unit file, it would be allocated to data units 4 and 5 and not split up between 1 and 4.

Each OS can choose an allocation strategy for a file system. Some file systems specify what strategy should be used, but there is no way to enforce it. You should test an implementation of the file system before assuming that it uses the strategy in the specification. In addition to testing the operating system to determine its allocation strategy, you should also consider the application that creates the content. For example, when updating an existing file some applications open the original file, update it, and save the new data over the original data. Another application might make a second copy of the original file, update the second copy, and then rename the copy so it overwrites the original. In this case, the file is located in new data units because it is part of a new file. This behavior should not be confused with the allocation strategy of the OS because the OS did not force new data units to be allocated.

3.3.2 DAMAGED DATA UNITS

Many file systems have the ability to mark a data unit as damaged. This was needed with older hard disks that did not have the capability to handle errors. The operating system would detect that a data unit was bad and mark it as such so that it would not be allocated to a file. Now, modern hard disks can detect a bad sector and replace it with a spare, so the file system functionality is not needed.

It is easy to hide data using the file system functionality, if it exists. Many consistency checking tools will not verify a data unit that the file system reports as being damaged is actually damaged. Therefore, a user could manually add a data unit to the damaged list and place data in it. Most acquisition tools report bad sectors, so that report can be compared the damaged list to identify sectors that may have been manually added to hide data.

3.3.3 ANALYSING TECHNIQUES

Now that we have looked at the basic concepts of data in the content category, we will look at how to analyze the data. This section covers different analysis techniques that can be used when searching for evidence.

3.3.4 DATA UNIT VIEWING

Data unit viewing is a technique used when the investigator knows the address where evidence may be located, such as one allocated to a specific file or one that has special meaning. For example, in many FAT32 file systems, sector 3 is not used by the file system and is all zeros, but data could be hidden there, and viewing the contents of sector 3 shows the investigator if there are non-zero data.

The theory behind this type of analysis is simple. The investigator enters the logical file system address of the data unit and a tool calculates the byte or sector address of the data unit. The tool then seeks to that location and reads the data. For example, consider a file system where data unit 0 starts at byte offset 0 and each data unit is 2,048 bytes. The byte offset of data unit 10 is 20,480 bytes, which we can see in Figure 3.4.

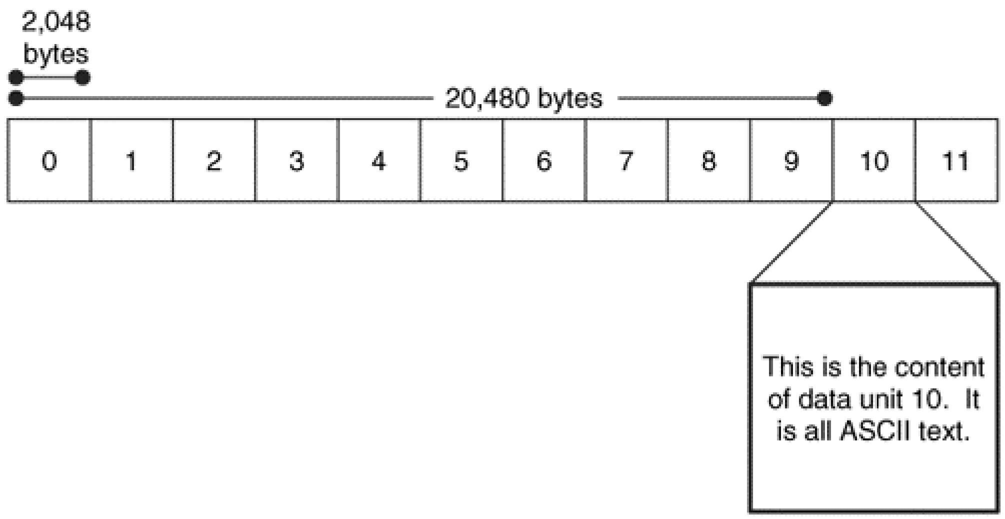


Figure 3.4 : Graphical Representation for viewing the data unit 10

There are many tools, such as hex editors and investigation tools, which perform this function. In TSK, the dcat tool allows you to view a specific data unit and displays it in raw or hexadecimal format.

3.4 **LOGICAL FILE SYSTEM-LEVEL SEARCHING**

In the previous technique, we knew *where* evidence could be, but we may not have known *what* the evidence would be. In this technique, we know what content the evidence should have, but we do not know where it is. A logical file system search looks in each data unit for a specific phrase or value. For example, you may want to search for the phrase "forensics" or a specific file header value. We can see this in Figure 3.5 where we are looking at each data unit for the string "forensics."

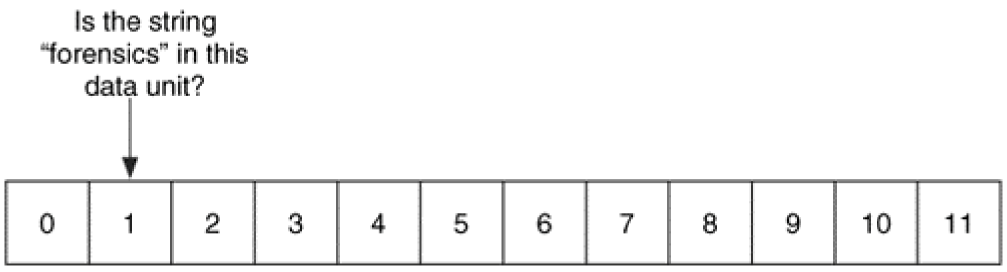


Figure 3.5 : Logical File System Search Look

This search technique has historically been called a physical search because it used the physical ordering of the sectors, but I do not feel this is accurate. This was accurate when a single disk was being analyzed, but this is not true for systems that use disk spanning and RAID. For those systems, the order of the sectors is not the physical order, and a more precise name should be used.

Unfortunately, files do not always allocate consecutive data units, and if the value you are searching for is located in two non-consecutive data units of a fragmented file, a logical file system search will not find it. We will see in the "Logical File Searching" section that a logical file search will find this value. Many forensic tools offer the capability to do both logical volume and logical file searches. To determine what search techniques your analysis tool uses, you can use some of the keyword search test images at the Digital Forensic Tool Testing (DFTT) site [Carrier 2004] that I maintain.

If we return to the theater analogy from the "Allocation Strategies" section, we may want to search for a specific family. A logical file system search of the theater for the family would start in the front row and examine each group of four consecutive people. If the family did not have consecutive seats, this search would fail. The search could be repeated by searching for only one person in the family, but that would probably result in many false hits from people who look similar.

3.5 **DATA UNIT ALLOCATION STATUS**

If we do not know the exact location of the evidence, but we know that it is unallocated, we can focus our attention there. Some tools can extract all unallocated data units from the file system image to a separate file, and others can restrict their analysis to only the unallocated areas. If you extract only the unallocated data, the output will be a collection of raw data with no file system structure, so you cannot use it in any file system analysis tools.

We can see this in Figure 3.6 where the bitmap for the first 12 data units are shown. A bitmap is a data structure that has a bit for each data unit. If the bit is 1, the data unit is allocated, and if it is 0, it is not. If we were going to extract the unallocated data units, we would extract units 2, 6, 7, and 11.

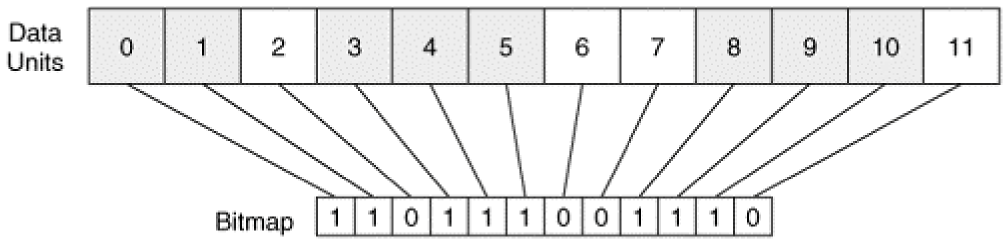


Figure 3.6 : Allocation Bitmap

Many digital forensic analysis tools allow you extract the unallocated space of a file system, although the definition of unallocated space may vary. I have observed that some tools consider any data not allocated to a file to be unallocated data, including the data in the file system and metadata categories. Alternatively, some tools recover deleted files and consider the data units of the deleted files to be allocated, even though they are technically unallocated.

Chapter 4 LITERATURE REVIEW

Mobile file carving is a new and currently growing area in the forensic science. Lots of studies are going on in this field. Some the literatures used for the study of this mobile forensic are summarized below,

4.1 **FORENSIC ANALYSIS OF INTERNAL MEMEORY OF MOBILE PHONES**

Mobile phones have become a very important tool for personal communication. It is therefore of great importance that forensic investigators have possibilities to extract evidence items from mobile phones. Modern mobile phones store evidence items on SIM-cards as well as internal memories. With the advent of modern functionality, such as camera and multimedia messaging, more and more of these items are stored in internal memory. Proper forensic examination of such memories, including recovery of deleted items, has not been possible until now.

It has been shown that a logical analysis of a memory dump can be conducted in order to enumerate evidence items present on the phone, including items that have been deleted. The existence of embedded memory managers in mobile phones was discovered. Memory manager's reorganization of memory can result in the overwriting of deleted evidence items. Therefore, the current practice of analyzing mobile phone internal memory through phone operating system should be reconsidered.

When mobile phones are seized, they must immediately be shut off. Any SIM card or external flash memory is removed and analyzed separately with a forensic analysis solution. The internal memory of the phone is imaged and analyzed using a method that has yet to be defined. Further research should be conducted in order to determine more practical methods for reading mobile phone internal memory, and enumerate evidence items found therein.

4.2 **CHARACTERIZING FAILURES IN MOBILE OS**

As smart phones grow in popularity, manufacturers are in a race to pack an increasingly rich set of features into these tiny devices. This brings additional complexity in the system software that has to fit within the constraints of the devices (chiefly memory, stable storage, and power consumption) and hence, new bugs are revealed. How this evolution of smartphones impacts their reliability is a question that has been largely unexplored till now. With the release of open source OSes for hand-held devices, such as, Android (open sourced in October 2008) and Symbian (open sourced in February 2010), we are now in a position to explore the above question. In this paper, we analyze the reported cases of failures of Android and Symbian based on bug reports posted by third party developers and end users and documentation of bug fixes from Android developers. First, based on developer reports, our study looks into the manifestation of failures in different modules of Android and their characteristics, such as, their transience in time. Next, we analyze similar properties of Symbian bugs based on failure reports. Our study indicates that Development tools, Web browsers, and Multimedia applications are most error-prone in both these systems. We further analyze bug fixes for Android and categorized the different types of code modifications required for the fixes. The analysis shows that 78% of errors required minor code changes, with the largest share of these coming from modifications to attribute values and conditions. Our final analysis focuses on the relation between customizability, code complexity, and reliability in Android and Symbian. We find that despite high cyclomatic complexity, the bug densities in Android and Symbian are surprisingly low. However, the support for customizability does impact the reliability of mobile OSes and there are cautionary tales for their further development.

Our work is a step toward the failure characterization of an OS for mobile phones. We presented a measurement based failure analysis of two operating systems—Android and Symbian—by studying publicly available bug databases. The key findings are: (1) Most of the bugs (more than 90%) in both these platforms are permanent in nature, suggesting that the codebases are not yet mature. (2) The Kernel layer in both the platforms are sufficiently robust, however, much effort is needed to improve the Middleware (Application Framework and Libraries in Android). (3) Development tools, Web, Multimedia, and Build failures are most prevalent in both the platforms. This suggests the necessity for a better mobile application development tools and need for caution in using third-party libraries. (4) Android offers a great degree of customizability in both the build and the execution processes. This customizability comes at a cost for a significant fraction of bugs—between 11% and 50% (assuming all of Modify settings, Add/modify cond, Preprocess changes, and Major changes are due to customizability). At present, the percentage of build errors is also high in Symbian (38.6%). (5) According to our analysis, a significant minority of the bugs in Android (22%) needed major code changes. Among various types of code modifications, fixing variable assignments and control flow update (adding if-else clause) are most widespread.

Our study also highlights the significant contributions of Android and Symbian in the field of mobile OS development. Both these platforms have established well-defined and well-maintained open mechanisms for reporting and dealing with bugs, without which a study like ours would not have been possible. This level of transparency is a pioneering and creditable effort in the smartphone world.

Our future work will focus on analyzing the propagation of errors among various layers, more specifically, between the Middleware and the applications. We also want to more fully explore the relationship between customizability, complexity, and reliability in a mobile operating system.

4.2.1 RELATED WORKS OF CHARACTERIZING FAILURES

Reliability literature over the years contains the results of many research efforts directed at analyzing bug reports for popular operating systems [18], [20], [21], [22]. In one of the early works on OS reliability, Sullivan et al. [18] analyzed defects of the MVS operating system based on empirical failure records documented by field IBM staff. This work categorized defects as overlay (errors that corrupt memory) and regular (those that do not corrupt memory). The frequency and effects of both types of errors were analyzed.

Chou et al. , in [4], presented their finding on OS errors by compiler attachments, which checks code for certain types of bugs, and counts bug density. The work discovered the correlation of different types of bugs with directories, function size, and file age. Our work looks into similar problem with different scope: instead of compiler attachments, our research is based on the developers' view and how bugs are fixed. In [22], the authors highlight the failure characteristics of BlueGene/L supercomputer by correlating data obtained from event loggers. Applying similar concept of failure event loggers, Cinque et al. [23] performed one of the few pieces of work that focuses on mobile OS reliability. In this work, the authors attached fault event loggers to a set of 25 Symbian OS based mobile phones to record failure events and panics (kernel-generated warnings for Symbian). Through this, the authors unveiled characteristics of the panics (burst, etc.) and the relation between panics and user-visible failures. However, since Symbian was not open-sourced at the time of this analysis, their research was limited to the manifestation of failures. Our paper extends the scope beyond these findings by classifying failures according to their root causes in the source code (for Android). We also present an analysis on customizability and complexity of a mobile OS (Android) which is distinct from previous work. Our work is also comparable to the failure modes catalog for wireless applications presented by Jha et al. [24]. In this work, the authors created a catalog for risk based testing of wireless applications based on observed and predicted failures. Our work also tries to estimate the likelihood of failure at different segments of a mobile OS which can be used for risk analysis. Comparing to this catalog [24], most of the bugs observed in our analysis were found under the category "Product Elements" which deals with mobile middleware, platform, synchronization, memory management etc.

4.3 **FILE CARVING EVOLUTION**

Year by year, the number of computers and other digital devices being used is increasing. The recent Pew Research Center Globalization Review [25] showed that 26 of the 36 countries surveyed had increased their computer usage. This increase is occurring simultaneously with an increase in usage of other digital devices, such as cell phones. In fact, in the United States alone 81% of the population now owns a cell phone, which is a 20% increase compared to 2002. Some countries, including Russia, have shown upwards of a 50% increase in cell phone ownership.

Computers are now one of many devices where digital data is stored. Devices such as cell phones, music players, and digital cameras all now have some form of internal storage or else allow data to be stored to external devices like flash cards, memory sticks, and solid-state devices (SSDs). With this huge increase in digital data storage, the need to recover data due to human error, device malfunction, or deliberate sabotage has also increased.

Data recovery is a key component of the disaster recovery, forensics, and e-discovery markets. Digital data recovery can consist of both software and hardware techniques. Hardware techniques are often used to extract data from corrupted or physically damaged disks. Once the data has been extracted, software recovery techniques are often required to order and make sense of the data. In this article, we will be solely discussing software techniques for recovery of data with a focus on digital forensics. We will begin by providing a quick overview of traditional data recovery techniques and then describe the problems involved with such techniques. We then introduce the techniques involved in file carving.

Evolutions of file carving and described in detail the techniques that are now being used to recover files without using any file system meta-data information. We have shown the benefits and problems that exist with current techniques. In the future, SSDs will become much more prevalent. SSDs will incorporate wear-leveling, which results in files being moved around so as to not allow some clusters to be written to more than others. This is done because after a certain amount of writes a cluster will fail and, therefore, the SSD controller will attempt to spread the write load across all clusters in the disk. As a result, SSDs will be naturally fragmented, and should the disk controller fail the clusters on the disk will require file carving techniques to recover. There is a lot of research yet to be done in this area for data recovery. Finally, while Pal et. al's techniques are useful for recovering text and images, new weighting techniques need to be created for video, audio, executable and other file formats, thus allowing the recovery to extend to those formats.

4.4 **ANDROID FORENSICS**

It is hardly appropriate to call the devices many use to receive the occasional phone call a *telephone* any more. The capability of these devices is growing, as is the number of people utilizing them. By the end of 2009, 46.3% of mobile phones in use in the United States were reported to be smart phones (AdMob, 2010).

With the increased availability of these powerful devices, there is also a potential increase for criminals to use this technology as well. Criminals could use smart phones for a number of activities such as committing fraud over e-mail, harassment through text messages, trafficking of child pornography, communications related to narcotics, etc. The data stored on smart phones could be extremely useful to analysts through the course of an investigation. Indeed, mobile devices are already showing themselves to have a large volume of probative information that is linked to an individual with just basic call history, contact, and text message data; smart phones contain even more useful information, such as e-mail, browser history, and chat logs. Mobile devices probably have more probative information that can be linked to an individual per byte examined than most computers -- and this data is harder to acquire in a forensically proper fashion.

Part of the problem lies in the plethora of cell phones available today and a general lack of hardware, software, and/or interface standardization within the industry. These differences range from the media on which data is stored and the file system to the operating system and the effectiveness of certain tools. Even different model cell phones made by the same manufacture may require different data cables and software to access the phone's information.

The good news is there are numerous people in the field working on making smart phone forensics easier. Already there is material available on how to conduct an examination on Blackberry phones and a growing number of resources about the iPhone. However, there is a new smart phone OS on the market named Android and it will likely gain in appeal and market share over the next year. While Android initially launched with only one phone on T-Mobile, phones are now available on Sprint, Verizon and AT&T as well.

4.4.1 ANDROID OS

Android is an operating system (OS) developed by the Open Handset Alliance (OHA). The Alliance is a coalition of more than 50 mobile technology companies ranging from handset manufactures and service providers to semiconductor manufacturers and software developers, including Acer, ARM, Google, eBay, HTC, Intel, LG Electronics, Qualcomm, Sprint, and T-Mobile. The stated goal of the OHA is to "accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience"

Android has a large community of developers writing applications that extend the functionality of the devices. Developers write primarily in a customized version of Java. Apps can be downloaded from third-party sites or through online stores such as Google lay (formerly *Android Market*), the app store run by Google. In October 2011, there were more than 500,000 apps available for Android, and the estimated number of applications downloaded from the Android Market as of December 2011 exceeded 10 billion.

4.5 FORENSIC RECOVERY OF FLASH MEMORY

Evolution in consumer electronics has caused an exponential growth in the amount of mobile digital data. The majority of mobile phones nowadays has a build in camera and is able to record, store, play and forward picture, audio, and video data. Some countries probably have more memory sticks than inhabitants. A lot of this data is related to human behavior and might become subject of a forensic investigation.

Flash memory is currently the most dominant non-volatile solid-state storage technology in consumer electronic products. An increasing number of embedded systems use high level file systems comparable to the file systems used on personal computers. Current forensic tools for examination of embedded systems like mobile phones or PDAs mostly perform logical data acquisition. With logical data acquisition it's often not possible to recover all data from a storage medium. Deleted data for example, but sometimes also other data which is not directly relevant from a user standpoint, cannot be acquired and potentially interesting information might be missed. For this reason data acquisition is wanted at the lowest layer where evidence can be expected. For hard disk based storage media it's common to copy all bytes from the original storage device to a destination storage device and then do the analysis on this copy. The same procedure is desired for embedded systems with solid-state storage media.

In this forensic analysis of flash memory we suggest a low level approach for the forensic examination of flash memory. The most important technology basics of flash memories are explained. And describes three low-level data acquisition methods for flash memories, first with so called flasher tools, then by usage of an access port commonly used for testing and debugging and finally with a semi-invasive method where the flash memory chips are physically removed from the printed circuit board.

4.5.1 FLASH MEMORY

Flash memory is a type of non-volatile memory that can be electrically erased and reprogrammed. Flash memory comes in two flavors, NOR1 flash and NAND2 flash, named after the basic logical structures of these chips. Contrary to NAND flash, NOR flash can be read byte by byte in constant time which is the reason why it is often used when the primary goal of the flash memory is to hold and execute firmware3, while parts of NOR flash that are not occupied by firmware can be used for user data storage. Most mobile media, like USB flash disks, or multimedia centred devices like digital camera's and camera phones, use NAND flash memory to create compact mobile data storage. This chapter explains the basics of flash technology first on the physical level and then from a logical perspective. An introduction to NAND flash memory can be found in [49], more in depth information can be found in [53].

4.6 **FRAGMENTATION**

Modern operating systems try to write files without fragmentation because these files are faster to write and to read. But there are three conditions under which an operating system must write a file with two or more fragments:

1. There is no contiguous region of sectors on the media large enough to hold the file without fragmentation. This is likely if a drive has been in use a long time, is filled to near capacity, and as had many files added and deleted in more-or-less random order over time.

2. If data is appended to an existing file, there may not be sufficient unallocated sectors at the end of the file to accommodate the new data. In this case, some file systems may relocate the original file, but mostly, they will simply write the appended data to another location.

3. The file system itself may not support writing files of a certain size in a contiguous manner. Or example, the Unix file system (UFS) will fragment files that are long or have bytes at the end of the file that will not fit into an even number of sectors.

Simon Garfinkel[2] researched fragmentation statistics by investigating 350 disks containing NTFS, FAT, and UFS. He showed that the fragmentation rate of user files (email, JPEG, Microsoft Word, and Microsoft Excel) is high. Microsoft Word’s fragmentation rate was found to be 17 percent; for JPEG files, it was 16 percent; and for Microsoft Outlook’s PST files, it was 58 percent. For carving fragmented files that have no beginning or a common string, we use advanced carving techniques based on file structure. An example of many techniques that can be used is “file structure carving.” File structure carving makes use of recognizable structures outside the header and footer signatures. In the example of the JPEG-layout (Figure 1), not only are the header and footer values used, but also the identifier strings and size to search block by block for JPEG files.

4.7 **FRAGMENTED FILE RECOVERY**

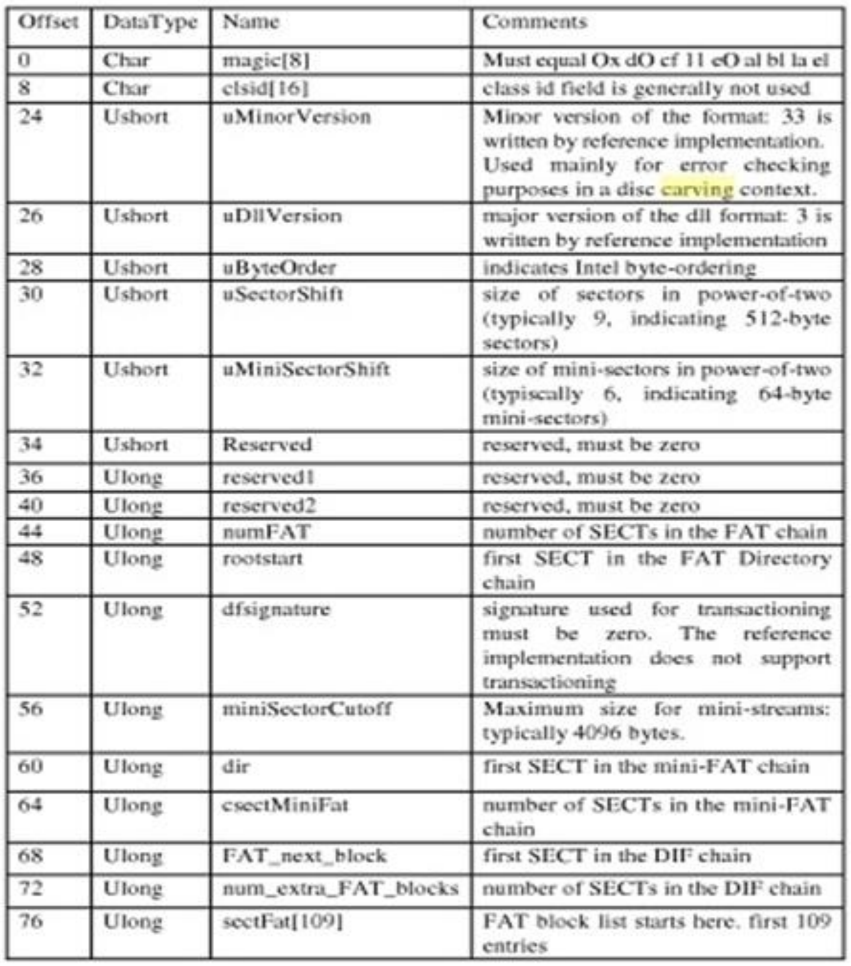
Reassembly of file fragments in the absence of file table information is a challenging problem in Digital Forensics and File Recovery. In literature we find file type specific solutions to this problem. In this paper we investigate a generalized solution that can be applied to every type of file. We present a technique to reassemble any type of fragmented file without using file table information. We have also presented time and space complexities of the algorithms in O-notation. In this technique each file on the disk is a Doubly Linked List with clusters as its nodes. We indicate some sections of Operating System that needs to be modified.

An Operating System divides a disk into two sections namely System Area and Data Area. The System Area holds data structures to implement File System and Data Area holds data of files. When a new file is created, Operating System allocates required number of clusters for it. The set of allocated clusters need not form contiguous area on disk. That is file's data may be fragmented. The list of clusters for a given file is stored in a suitable data structure in a File System. For example, in a FAT File System, the list is stored in a data structure called File Allocation Table (FAT).

When a file is deleted the clusters of a file are marked as free but data of the file is available on disk as long as the clusters are not allocated to another file. And when a used disk is formatted for reuse the data structures in System Area are initialized but the data of old files is available on disk as long as the clusters are not allocated to new files.

Sometimes files are accidentally deleted. In these cases files content is available on disk in fragmented form but the information of reassembling these files is missing. The disk is in initial state just after formatting a new disk. In this state the disk has no files. When user starts creating files on it, it moves to 'Fully Recoverable State'.

While the disk is in this state, if file(s) is/are deleted or disk is formatted or more files are created without overwriting clusters, then it remains in this state as long as used clusters are overwritten. When used clusters are overwritten, then it moves to 'Partially recoverable State'. It remains in this state as long as disk is not physically damaged.



Offset	DataType	Name	Comments
0	Char	magic[8]	Must equal Ox dO cf 11 eO al bl la el
8	Char	clsid[16]	class id field is generally not used
24	Ushort	uMinorVersion	Minor version of the format: 33 is written by reference implementation. Used mainly for error checking purposes in a disc carving context.
26	Ushort	uDllVersion	major version of the dll format: 3 is written by reference implementation
28	Ushort	uByteOrder	indicates Intel byte-ordering
30	Ushort	uSectorShift	size of sectors in power-of-two (typically 9, indicating 512-byte sectors)
32	Ushort	uMiniSectorShift	size of mini-sectors in power-of-two (typiscally 6, indicating 64-byte mini-sectors)
34	Ushort	Reserved	reserved, must be zero
36	Ulong	reserved1	reserved, must be zero
40	Ulong	reserved2	reserved, must be zero
44	Ulong	numFAT	number of SECTs in the FAT chain
48	Ulong	rootstart	first SECT in the FAT Directory chain
52	Ulong	dfsignature	signature used for transactioning must be zero. The reference implementation does not support transactioning
56	Ulong	miniSectorCutoff	Maximum size for mini-streams: typically 4096 bytes.
60	Ulong	dir	first SECT in the mini-FAT chain
64	Ulong	csectMiniFat	number of SECTs in the mini-FAT chain
68	Ulong	FAT_next_block	first SECT in the DIF chain
72	Ulong	num_extra_FAT_blocks	number of SECTs in the DIF chain
76	Ulong	sectFat[109]	FAT block list starts here. first 109 entries

Figure 4.1 : **OLE Header Structure**

When forensic analyst needs to examine disk content, he cannot rely on what is presented by the Operating System using the data structures of File System. File recovery software do not rely on what is presented by the Operating System using the data structures of File System. We need to have techniques for reassembling fragments of files. Reassembling file fragments in the absence of file table information is called File Carving.Techniques of reassembling image files of type bit map are explored in [54], document files in [55] and image files of type JPEG in [56] and [57]. File type specific techniques suffer from drawbacks. For every future file type new techniques need to be developed. Moreover every existing file type should be treated separately.

Chapter 5 MOBILE FORENSIC

Mobile phones have become a very important tool for personal communication. It is therefore of great importance that forensic investigators have possibilities to extract evidence items from mobile phones. Modern mobile phones store evidence items on SIM-cards as well as internal memories. With the advent of modern functionality, such as camera and multimedia messaging, more and more of these items are stored in internal memory. Proper forensic examination of such memories, including recovery of deleted items, has not been possible until now.

It is clear that mobile phones contain information that may have value as evidence in investigations. The mobile phone has become the modern person's primary tool for personal communication, and therefore frequently contains information about a person's activities. Obtaining information on such activities is often a primary goal in an investigation. Analyzing the content of a mobile phone is therefore an invaluable tool for the forensic investigator.

5.1 **MOBILE PHONE ARCHITECTURE**

In order to fully understand how forensic analysis of mobile phone systems can be conducted, it is important to understand how mobile phones are built. The system architecture of a mobile can generally be viewed as the architecture on.

The central unit of the phone is the CPU. The CPU controls the communication circuits of the phone, in addition to control the communication with the user. For intermediary storage, the CPU uses a RAM. RAM is used for all intermediary storage during communication and user interaction. The RAM can be implemented as a separate integrated circuit or it can be integrated with the CPU in a single integrated circuit.

The phone also needs a secondary non-volatile storage (shown as secondary storage on the figure). This is needed for storage of all data pertaining to user and communication that needs to persist during a power failure. Secondary storage can be implemented in various ways. The most common implementation today on mobile phones is a separate flash memory integrated circuit on the system board. In addition to these elements, the CPU has communication with the SIM, and optionally other external storage media. It is also common to have a special unit to control the usage of power in a mobile phone.

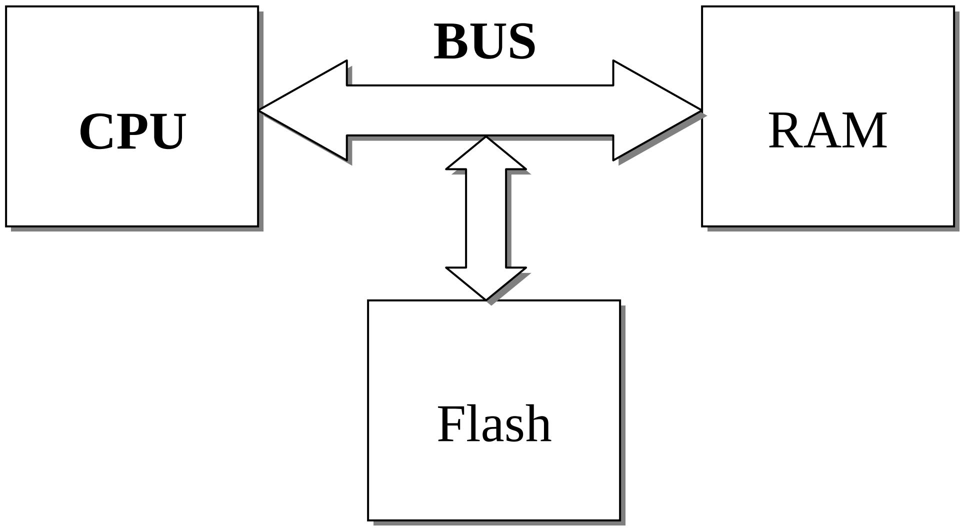


Figure 5.1: **General architecture of mobile phone**

One should note the similarity of the architecture of a mobile phone with that of a computer. Where computers normally use a hard drive for secondary storage, a mobile phone uses flash memory. Otherwise the mobile phone architecture is like a computer. In this essence, it is not unfair to say that the mobile phone is a computer. The difference in secondary storage is however very important for forensic investigators.

5.2 **SOME MOBILE PHONES AND THEIR INFORMATIONS**

5.2.1 NOKIA

Some of the models 3200, 3410, 3510i, 5110, 6110, 6150, 6210, 6230, 6310i and 6610 were analyzed. The following behavior was observed on the analyzed Nokia phones: Text messages are stored on the SIM. When the SIM is full (max 20-30 messages), the phone uses internal memory (up to 150 messages common on most models). Older models store only incoming messages, but newer models store both outgoing and incoming, but only incoming is stored on SIM. With Nokia phones, deleted messages on SIM can be recovered. Contacts can be stored on SIM or internal memory, and the user can select which memory to use. Older Nokia phones cannot store contacts in internal memory.

The analyzed Nokia phones use internal memory for all other data such as calendar events, caller logs, pictures etc. The call log file on the SIM is not used by Nokia. If the SIM is changed, the phone will delete the caller logs, but all other data will remain. Only one of the analyzed Nokia phones (the 6230) had external flash memory. This could be used as additional storage for pictures and sound files. Text messages cannot be stored on the external card. Multimedia messages will implicitly be stored in internal memory, but may be moved to external memory by the user.

5.2.2 SONY ERICSSON

The models A2618s, GH688, R380s, S868, T68, T68i, T610 and T630 were analyzed. For the analyzed Sony Ericsson phones, it was observed that text messages are stored on internal memory until it is full, and only then will the phone start to use the SIM. As a consequence, SIM cards will in most cases contain nothing when they have been used in Sony Ericsson phones. For phones with external memory cards, it is similar to Nokia only possible to copy pictures and sounds to these, and only at the user's explicit request.

5.2.3 SIEMENS

The models A60, C25, C60, C62, M55 and M65 were analyzed. The analyzed Siemens phones use SIM as primary storage for text messages and log of outgoing calls. When the SIM memory is full, internal memory is used. For contacts, the user can choose whether to use internal or SIM memory, but internal memory is the default. None of the Siemens phones deleted items when switching the SIM card. No Siemens phone with external memory was analyzed.

5.3 ANDROID PHONES

The Android platform is a software stack for mobile devices that consists of an operating system, middleware and key applications [16]. Android offers many features covering the areas of application development, Internet, media, and connectivity. These features include Application framework, Dalvik virtual machine, Integrated browser, Optimized graphics, SQLite for structured data storage, Media support for common audio, video, and still image formats, GSM Telephony, Bluetooth, EDGE, 3G, and WiFi, Camera, GPS, Compass, and a rich Development environment. The Android platform primarily consists of five layers:

5.3.1APPLICATIONS

This includes a set of core applications that come with the Android distribution like Email Client, Messaging application, Contacts application, Calendar, Map browser, Web browser etc.

5.3.2 APPLICATION FRAMEWORK

This layer has been designed to facilitate the reuse of components in Android. With the help of Application Framework elements (such as, Intents, Content Providers, Views, and Managers) in Android, developers can build their applications to execute on Android Kernel and inter-operate among themselves and with existing applications.

5.3.3 LIBRARIES

Libraries include System C library, Surface Manager, 2D and 3D graphics engine, Media Codecs, the SQL database Sqlite and the web browser engine LibWebCore.

5.3.4 ANDROID RUNTIME

The Android runtime consists of two components.

- A set of Core libraries which provides most of the functionality available in Java.
- The Dalvik virtual machine which operates like a translator between the application side and the operating system. Every Android application runs in its own process, with its own instance of the Dalvik virtual machine.

5.3.5 LINUX KERNEL

Android uses a modified version of Linux 2.6 for core system services such as Memory Management, Process Management, Network Stack, Driver Model and Security. For more information on the Android platform and a schematic of the Android architecture the readers are referred to [16].

5.4 SYMBIAN

Symbian OS, currently being used by several leading mobile phone manufacturers, account for 46.9% of global smart phone sales, making it the world's most popular mobile operating system [7]. It is a lightweight operating system designed for mobile devices and smart phones, with associated libraries, user interface, frameworks and reference implementations of common tools, originally developed by Symbian Ltd [14].

3 design principles: (i) Real time processing, (ii) Resource limitation, and (iii) Integrity and security of user data. To best follow these principles, Symbian uses a hard real-time, multithreaded microkernel, and has a request-and-callback approach to services. Symbian’s system model is segmented into 3 main layers [15]:

5.4.1 OS LAYER

Includes the hardware adaptation layer (HAL) that abstracts all higher layers from actual hardware and the Kernel including physical and logical device drivers. It also provides programmable interface for hardware and OS through frameworks, libraries and utilities etc. and higher level OS services for communications, networking, graphics, multimedia and so on.

5.4.2 MIDDLEWARE LAYER

Provides services (independent of hardware, applications or user interface) to applications and other higher-level programs. Services can be specific application technology such as messaging and multimedia, or generic to the device such as web services, security, device management, IP services and so on.

5.4.3 APPLICATION LAYER

Contains all the Symbian provided applications, such as multimedia applications, telephony and IP applications etc. Symbian is optimized for low-power battery-based devices and ROM-based systems. Here, all programming is event-based, and the CPU is switched into a low power mode when applications are not directly dealing with an event. Similarly, the Symbian approach to threads and processes is driven by reducing memory and power overheads. Readers are referred to [15] for further details on the Symbian architecture.

Chapter 6 PROPOSED APPROACH

File carving is very important in the forensic science for finding the proofs and details left over in the devices which we got at the time of the crime scene. Based on the definition of file carving, it is the process of recovering the data from a hard disk or a storage media even after the file or data which had been deleted or corrupted or damaged during some process. This recovery is done without using the storage meta-data, like file tables.

Now a days the usage of the mobile phones have been increased, each and every one is having a mobile phone with them, so, the possibility of the becoming a proof of crime scene for these mobile devices become more. Here comes the mobile forensics. After the mobile phones, PDAs or tablets retrieved from the crime locations they are sent to the forensic labs for examination. The data from these devices can be recovered by this mobile carving technique.

6.1 **RECOVERY OF DATA WITHOUT USING STORAGE META-DATA**

File system structures are not used during the process. File carving is a powerful technique for recovering files and fragments of files when directory entries are corrupt or missing. The block of data is searched block by block for residual data matching the file type-specific header and footer values. Carving is also especially useful in criminal cases where the use of carving techniques can recover evidence. In certain cases related to child pornography, law enforcement agents are often able to recover more images from the suspect's hard disks by using carving techniques. Another example is the hard disks and removable storage media US Navy Seals took from Osama Bin Laden's campus during their raid. Forensic experts used file carving techniques to squeeze every bit of information out of this media.

As long as data is not overwritten or wiped, deleted data on all storage devices can be restored using carving techniques, including multifunctional devices and even mobile phones. Depending on the conditions, it is even possible to restore data from formatted disks. With the exhaustive measures of drives since 2006, there is a big chance that the data is not overwritten. For example, let's say you have a two-terabyte drive, and you delete a document from that drive. The disk space reserved for that document will be marked "available," but it could really take a long time before this address space on the disk is overwritten. There were forensic cases where we discovered files stored on the disk years ago. In this paper, the basic techniques of file carving tools like Foremost and Photorec, are explained for recovering data from several media types.

6.2 **MOBILE PHONE MEMORY CARVING**

Recovery of data from mobile phones is different from the recovery from the computer and laptops. Here we need to recover the messages sent, both text messages and multimedia messages. The images, videos, notes, calendar items, contacts and call history. And another thing to be noted here is we have a different file system using flash memory for storing. So file carving will be more suitable for the recovery of data from the mobile phone devices.

6.3 **EVIDENCE IN MOBILE PHONE**

Mobile phones are digital media. In principle, this means that mobile phones have the same evidentiary possibilities as other digital media, such as hard drives. For example it is, as will be explored in this paper, possible to extract deleted information from a mobile phone, in the same way it is possible on a hard drive. However, mobile phones also suffer from the same evidentiary problems as other digital media. As with a computer, the content of a mobile phone is fragile and can easily be deleted and overwritten. Mobile phones should therefore be handled with great care and insight, just as any other digital media.

6.4 **EVIDENCE ITEMS**

A long range of evidence items can be found in modern mobile phones. A few is listed in [11]. In addition to the evidence related to the phone system itself, modern phones have other evidence items that should be mentioned:

- Images
- Sounds
- Multimedia messages
- WAP/web browser history
- Email
- Calendar items
- Contacts

Last but not least, the importance of SMS text messages should not be underestimated. The Short Message Service is very widely used, and messages are stored on the sending and receiving mobile phone. The ability to recover deleted text messages would have great value in many investigations. The investigation of possibilities to recover deleted text messages has been the main motivation for this research.

6.5 **STORAGE MEDIA**

With the advent of digital mobile telephone systems, such as GSM, a need for local digital storage emerged. In the GSM system, the single most popular digital mobile phone system, the SIM (Subscriber Identity Module) was specified as storage medium and implemented as a smart card that fits inside the phone. The SIM contains subscriber information and secret encryption keys necessary for the communication. It also implements storage space for contacts and text messages. As described in [11] and [12], the SIM can be soundly forensically analyzed. It is also possible to recover some deleted data from the SIM (deleted text messages on from some phone models). The SIM architecture is also used in 3G systems (USIM).

However, since the late 1990s, mobile phone manufacturers began to use mobile phone internal memory in addition to the SIM for storage of information items. The SIM has a rigorous specification allowing only for certain types of information to be stored. It does not provide a single continuous memory that can be utilized for any purpose. As the manufacturers wanted to implement new functions where storage on SIM was not available, mobile phones were gradually equipped with internal memory for storage of items such as missed and received calls, calendar events, text messages, contacts and other items.

The first models used a serial EEPROM chip for this purpose. With the growth of memory demand, it gradually became more common to implement internal memory on flash memory, either as a flash chip dedicated for information item storage, or as an area on the flash memory chip storing the phone system software. More recently, with the advent of telephones with cameras and MP3 players, it has also become common to add possibilities for external flash memory in mobile telephones. External memory can be added by inserting a memory card such as SD, MMC, CF or similar. Sound forensic analysis of external memory cards can be accomplished by using existing computer based forensic tools such as [13], [14] and [15].

6.6 **USAGE OF EXTERNAL MEMORY, INTERNAL MEMORY AND SIM**

In order to understand the value of analyzing mobile phone internal memory, one must understand where the different information items in mobile phones are stored. In order to understand this properly, a range of mobile phones with SIM, internal storage, and to a certain extent external flash storage was analyzed to determine what information items are stored on the different media types. The phones were examined by sending and receiving text messages, taking pictures, store contacts and calendar events, exchanging SIM cards and external memory cards, and observe the behaviour. The results indicate that each manufacturer is consistent in the way data is handled, but the variation between manufacturers is significant. The results are summarized in the following, grouped on each manufacturer.

6.7**MEMORY CONTENT EXPERIMENTS**

The focus of this research has been to identify if deleted items can be recovered by analyzing memory dumps and in particular if deleted text messages can be recovered. Several experiments were done to find out if deleted text messages can be recovered. The experiments were conducted by successively reading out internal memory after committing changes to the system using the phone operating system. The limitations of the memory reading methods made it difficult to do such experiments on more than a few tests. Another difficulty was the lack of a software tool to do identification and interpretation of messages inside memory dumps. The implementation of such a tool was beyond the scope of this project.

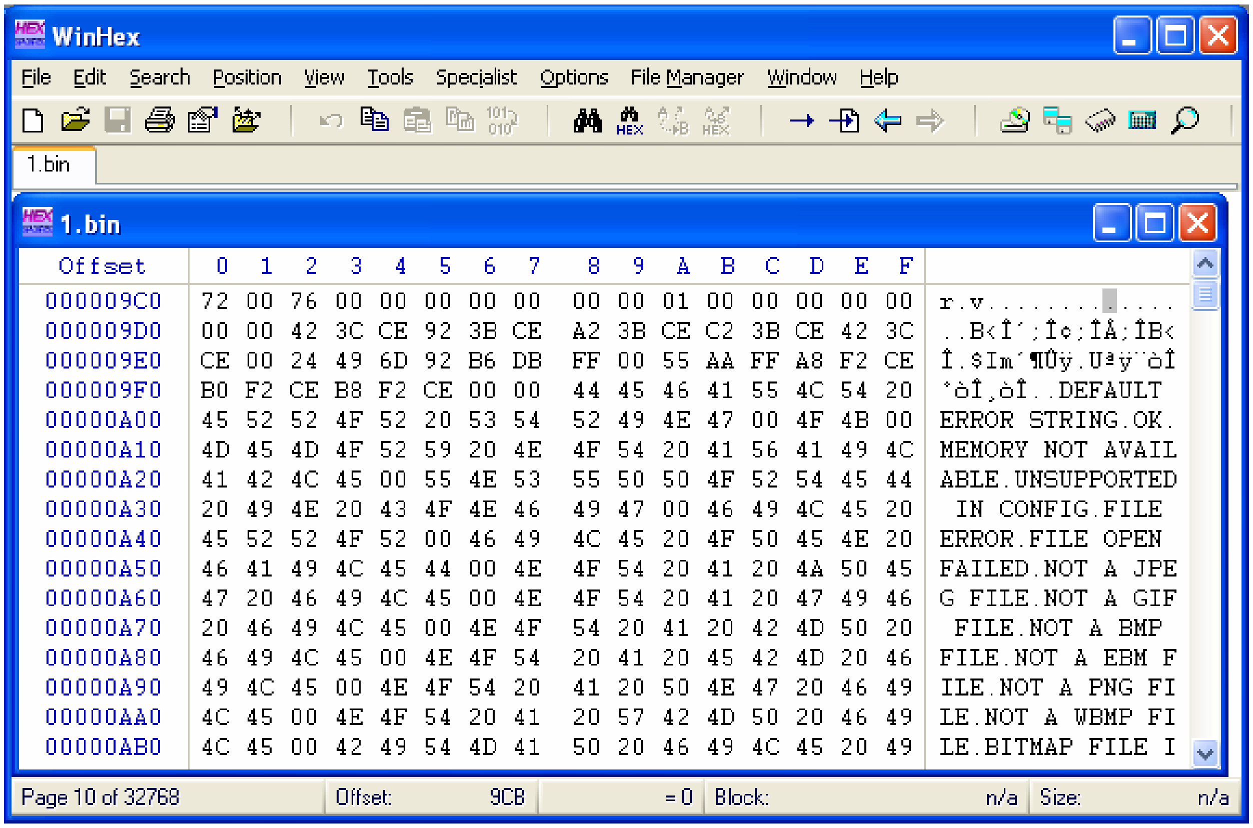
Several interesting observations were done:

Text messages were still in flash memory after they had been deleted in the operating system. It was found that images, MMS, calendar items and contacts were also in memory after they had been deleted. Contrary from what is possible on SIM-cards it is therefore possible to recover deleted items of this type from internal memory.

Copies of items from previously used SIM-cards were found in some occasions. During the analysis of successive memory dumps from the same device, another very important property was discovered. During the search for deleted text messages it was found that memory areas found in one memory dump was moved to quite another area on the following dump. This is most likely due to the existence of a Memory Manager that dynamically reallocates memory during phone usage in order to ensure optimal memory organization and utilization at all times during phone usage. Although not particularly surprising, the existence of such memory managers do have ramifications for the forensic handling of mobile phones.

6.8 MEMORY REPRESENTATION IN HEX VALUES OF THE MEMORY DUMP OF MOBILE PHONE

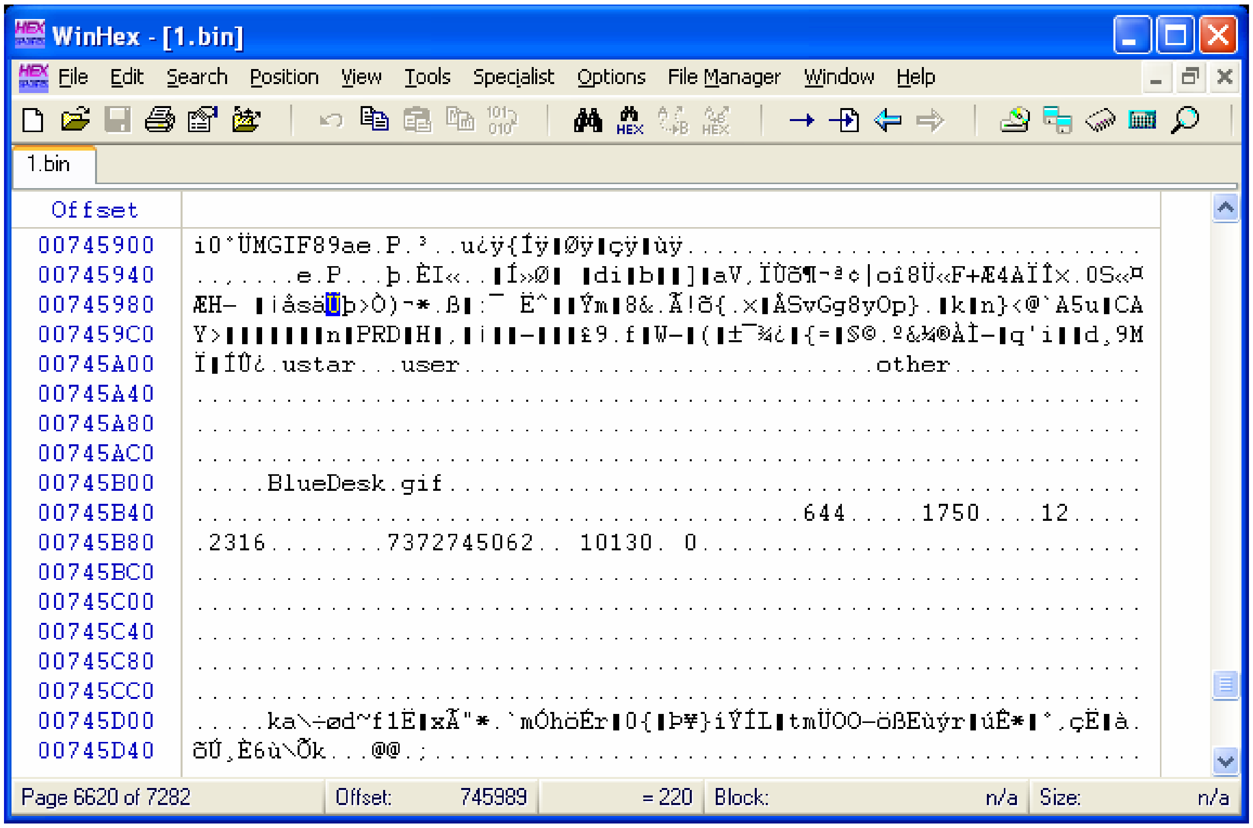
The below figures will show the hex values of the memory dump of the mobile taken for the file carving.


WinHex
File Edit Search Position View Tools Specialist Options File Manager Window Help
1.bin
1.bin

```
Offset     0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
000009C0  72 00 76 00 00 00 00 00  00 00 01 00 00 00 00 00  r.v.............
000009D0  00 00 42 3C CE 92 3B CE  A2 3B CE C2 3B CE 42 3C  ..B<Î´;Î¢;ÎÂ;ÎB<
000009E0  CE 00 24 49 6D 92 B6 DB  FF 00 55 AA FF A8 F2 CE  Î.$Im´¶Ûÿ.Uªÿ¨òÎ
000009F0  B0 F2 CE B8 F2 CE 00 00  44 45 46 41 55 4C 54 20  °òÎ¸òÎ..DEFAULT 
00000A00  45 52 52 4F 52 20 53 54  52 49 4E 47 00 4F 4B 00  ERROR STRING.OK.
00000A10  4D 45 4D 4F 52 59 20 4E  4F 54 20 41 56 41 49 4C  MEMORY NOT AVAIL
00000A20  41 42 4C 45 00 55 4E 53  55 50 50 4F 52 54 45 44  ABLE.UNSUPPORTED
00000A30  20 49 4E 20 43 4F 4E 46  49 47 00 46 49 4C 45 20   IN CONFIG.FILE 
00000A40  45 52 52 4F 52 00 46 49  4C 45 20 4F 50 45 4E 20  ERROR.FILE OPEN 
00000A50  46 41 49 4C 45 44 00 4E  4F 54 20 41 20 4A 50 45  FAILED.NOT A JPE
00000A60  47 20 46 49 4C 45 00 4E  4F 54 20 41 20 47 49 46  G FILE.NOT A GIF
00000A70  20 46 49 4C 45 00 4E 4F  54 20 41 20 42 4D 50 20   FILE.NOT A BMP 
00000A80  46 49 4C 45 00 4E 4F 54  20 41 20 45 42 4D 20 46  FILE.NOT A EBM F
00000A90  49 4C 45 00 4E 4F 54 20  41 20 50 4E 47 20 46 49  ILE.NOT A PNG FI
00000AA0  4C 45 00 4E 4F 54 20 41  20 57 42 4D 50 20 46 49  LE.NOT A WBMP FI
00000AB0  4C 45 00 42 49 54 4D 41  50 20 46 49 4C 45 20 49  LE.BITMAP FILE I
```

```
Page 10 of 32768    Offset:    9CB    = 0    Block:    n/a    Size:    n/a
```

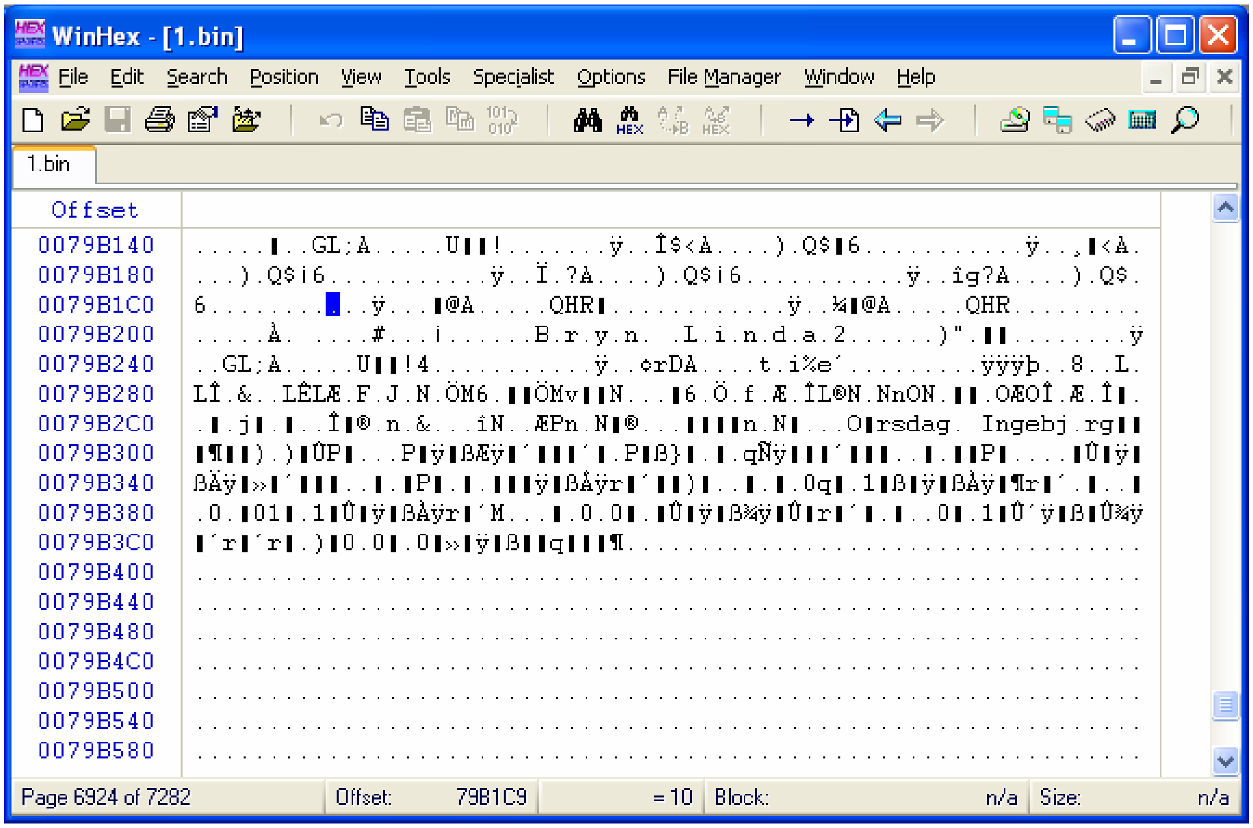
Figure 6.1 : Hex-print of a mobile memory dump



```
WinHex - [1.bin]
1.bin
Offset
00745900  i0˚ÜMGIF89ae.P.³..u¿ÿ{Íÿ▮Øÿ▮çÿ▮ùÿ..............................
00745940  ..,....e.P...þ.ÈI«..▮Í»Ø▮ ▮di▮b▮▮]▮aV,ÏÜõ¶¬ª¢|oí8Ü«F+Æ4AÏÎ×.0S«¤
00745980  ÆH– ▮iåsäÜþ>Ò)¬*.ß▮:¯ Ë^▮▮Ÿm▮8&.Ã!õ{.×▮ÅSvGg8yOp}.▮k▮n}<@`A5u▮CA
007459C0  Y>▮▮▮▮▮▮▮n▮PRD▮H▮,▮¡▮▮–▮▮▮£9.f▮W–▮(▮±¯¾¿▮{=▮S©.º&¼®ÀÌ–▮q'i▮▮d¸9M
00745A00  Ï▮ÎÛ¿.ustar..user.........................other.............
00745A40  ................................................................
00745A80  ................................................................
00745AC0  ................................................................
00745B00  .....BlueDesk.gif...............................................
00745B40  .........................................644....1750....12.....
00745B80  .2316........7372745062.. 10130. 0..............................
00745BC0  ................................................................
00745C00  ................................................................
00745C40  ................................................................
00745C80  ................................................................
00745CC0  ................................................................
00745D00  .....ka\÷ød~f1Ë▮xÃ"*.`mÓhöÉr▮0{▮Þ¥}iÝÍL▮tmÜOO–ößEùýr▮úÊ*▮˚,çË▮à.
00745D40  õÚ¸È6ù\Õk...@@.;...............................................
Page 6620 of 7282    Offset: 745989    = 220    Block: n/a    Size: n/a
```

Figure 6.2 : Content Identified as a GIF – image

The above figure 5.2 is showing the hex values of the GIF image from the memory dump of the mobile phone taken. The memory block 00745B00 is showing the name of the GIF image as "BlueDesk.gif".



WinHex - [1.bin]
File Edit Search Position View Tools Specialist Options File Manager Window Help
1.bin

```
Offset
0079B140  .....▮..GL;A....U▮▮!.......ÿ..Î$<A....).Q$▮6...........ÿ..¸▮<A.
0079B180  ...).Q$¡6...........ÿ..Ï.?A....).Q$¡6...........ÿ..îg?A....).Q$.
0079B1C0  6........▮..ÿ...▮@A.....QHR▮...........ÿ..¼▮@A.....QHR.........
0079B200  .....À.  ....#...¡......B.r.y.n.  L.i.n.d.a.2......)".▮▮.........ÿ
0079B240  ..GL;A.....U▮▮!4...........ÿ..¢rDA....t.i%e´.........ÿÿÿþ..8..L.
0079B280  LÎ.&..LÊLÆ.F.J.N.ÖM6.▮▮ÖMv▮▮N...▮6.Ö.f.Æ.ÎL®N.NnON.▮▮.OÆOÎ.Æ.Î▮.
0079B2C0  .▮.j▮.▮..Î▮®.n.&...íN..ÆPn.N▮®...▮▮▮▮n.N▮...O▮rsdag. Ingebj.rg▮▮
0079B300  ▮¶▮▮).)▮ÛP▮...P▮ÿ▮ßÆÿ▮´▮▮▮´▮.P▮ß}▮.▮.qÑÿ▮▮▮´▮▮▮..▮.▮▮P▮....▮Û▮ÿ▮
0079B340  ßÄÿ▮»▮´▮▮▮..▮.▮P▮.▮.▮▮▮ÿ▮ßÅÿr▮´▮▮)▮..▮.▮.0q▮.1▮ß▮ÿ▮ßÅÿ▮¶r▮´.▮..▮
0079B380  .0.▮01▮.1▮Û▮ÿ▮ßÅÿr▮´M...▮.0.0▮.▮Û▮ÿ▮ß¾ÿ▮Û▮r▮´▮.▮..0▮.1▮Û´ÿ▮ß▮Û¾ÿ
0079B3C0  ▮´r▮´r▮.)▮0.0▮.0▮»▮ÿ▮ß▮▮q▮▮▮¶..............................
0079B400  ................................................................
0079B440  ................................................................
0079B480  ................................................................
0079B4C0  ................................................................
0079B500  ................................................................
0079B540  ................................................................
0079B580  ................................................................
```

```
Page 6924 of 7282    Offset:    79B1C9    = 10   Block:    n/a   Size:    n/a
```

Figure 6.3 : Content Identified as Contact and Calendar Item

Chapter 7 CONCLUSION AND FUTURE SCOPE

Cell phones are becoming even more sophisticated and able. Both law enforcement and the private sector need to invest time and money into learning about new operating systems and developing new forensic methods. The technique for carving of the data from the mobile devices will be very useful to forensic department. In recent years, information criminal cases are mushrooming than ever, personal information leakage and Bot Network are no news anymore. As a result, besides users' daily prevention, how forensic examiners can help acquire information from victims' phones after harm happens becomes the most important issue now. This technique will help in the carving of the mobile data, with the help of the smartest recovery method, it will be very useful tool in the forensic. The techniques that are now being used to recover files without using any file system meta-data information.

In future the smart phones and more rich application oriented mobile phone OSes will be available in the market and carving will be done on the OS like adroid, symbian etc., These OSes are like the miniature form of the computer OS with certain functions reduced and made suitable for the small memory of the mobile phones. So the carving will be done based on the changes made. While Android forensics is still in its infancy, steps are being made to meet the new technology. CelleBrite (2010), Paraben (2008), and .XRY (Micro Systemation, 2008) all currently offer some type of Android solution and more tools will be adding support as Android gains in popularity. Android is not just for phones either; it can be used on computers, kitchen appliances, and military applications (Spencer, 2009). Expect to begin seeing it everywhere. The number of Android phones will be continuously increasing as more manufactures adopt the budding OS. As it stands now, Android sales, by some estimates, will overtake iPhone sales within the next two to three years (Lomas, 2009). While Android is powerful, complex, has multiple firmware implementations and some with manufactures making custom UIs, the standardization will make mobile forensics simpler in the long run. Indeed, as the market for Android continues to grow, learning how to forensically acquire information from these devices becomes essential for mobile device examiners.

REFERENCES

[1] B. Carrier , *File System Forensic Analysis* . Boston, MA: Pearson Education , Addison-Wesley Professional, 2005 .

[2] S. Garfinkel , " Carving contiguous and fragmented files with fast object validation", in *Proc. 2007 Digital Forensics Research Workshop (DFRWS)* , Pittsburgh , PA , Aug. 2007, pp.4S:2–12**.**

[3] Amiga Smart Filesystem [Online]. Available: http://www.xs4all.nl/ hjohn/SFS

[4] L.W. McVoy and S.R. Kleiman , " Extent-like performance from a UNIX file system", in *Proc. USENIX, Winter '91* , Dallas , TX , 1991 , pp. 33 – 43 .

[5] A. Sweeney , D. Doucette , W. Hu , C. Anderson , M. Nishimoto, and G .Peck, " Scalability in the XFS file system", in *Proc. USENIX 1996 Annu. Tech. Conf.* , San Diego , CA , 1996, pp. 1–14 .

[6] STORAGEsearch.com. Data Recovery from Flash SSDs? [Online].

Available: http://www.storagesearch.com/recovery.html

[7] *Foremost 1.53* [Online]. Available: http://foremost.sourceforge.net

[8] G. G. Richard , III and V. Roussev , " Scalpel: A frugal, high performance file carver", in *Proc. 2005 Digital Forensics Research Workshop (DFRWS)* , New Orleans , LA , Aug. 2005.

[9] B. Carrier , V. Wietse , and C. Eoghan , *File Carving Challenge 2006* [Online]. Available: http://www.dfrws.org/2006/challenge

[10] B. Carrier , V. Wietse , and C. Eoghan , *File Carving Challenge 2007* [Online]. Available: http://www.dfrws.org/2007/challenge

[11] S. Willassen, *Forensics and the GSM mobile telephone system*, International Journal on Digital Evidence 2003:2:1

[12] R. Knijff, *Embedded Systems Analysis*, Handbook of Computer Crime Investigation, Academic Press, 2002

[13] *Win-Hex*, Software Package, Commercial. Available: http://www.winhex.com/

[14] *EnCase*, Software Package, Commercial. Available: http://www.encase.com/

[15] *The Sleuthkit*, Software Package, Open Source. Available: http://www.sleuthkit.org/

[16] What is Android? http://developer.android.com/guide/ basics/what-is-android.html

[17] Svein Y. Willassen, Norwegian University of Science and Technology, “Forensic analysis of mobile phone internal memory”

[18] M.Sullivan and R.Chillarege, “Software Defects and their Impact on System Availability - A Study of Field Failures in Operating Systems,” In Proc. of 21st International Symposium on Fault- Tolerant Computing (FTCS), P. 2–9, 1991.

[19] S.Chandra and P.M.Chen. “Whither Generic Recovery from Application Faults? A Fault Study using Open-Source Software,” In Proc. Of the 30th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2000, P. 97–106, 25-28 June 2000.

[20] A.Chou, J.Yang, B.Chelf, S.Hallem, and D.Engler. “An Empirical Study of Operating Systems Errors,” In Proc. of 18th ACM Symposium on Operating Systems Principles (SOSP), P. 73–88, 2001

[21] W.Gu, Z.Kalbarczyk, R.K.Iyer, and Z.Yang. “Characterization of Linux Kernel Behavior under Errors,” In Proc. of International Conference on Dependable Systems and Networks (DSN), 2003, P. 459–468, 22-25 June 2003.

[22] Y.Liang, Y.Zhang, M.Jetta, A.Sivasubramaniam, and R.Sahoo. “Blue- Gene/L Failure Analysis and Prediction Models,” In Proc. of International Conference on Dependable Systems and Networks (DSN), 2006, P. 425–434, 25-28 June 2006.

[23] M.Cinque, D.Cotroneo, Z.Kalbarczyk, and R.Iyer. “How do Mobile Phones Fail? A Failure Data Analysis of Symbian OS Smart Phones,” In Proc. of International Conference on Dependable Systems and Networks (DSN), 2007, P. 585–594, 25-28 June 2007.

[24] A.Jha and C.Kaner, “Bugs in the brave new unwired world,” Pacific Northwest Software Quality Conference, Portland, OR, October 2003.

[25] *Pew Global Attitudes Project* [Online]. Available: http://pewglobal.org/reports/pdf/258.pdf

[26] A. Pal and N. Memon , " Automated reassembly of file fragmented images using greedy algorithms ," *IEEE Trans. Image Processing* , vol. 15, no. 2, pp. 385 – 393 , Feb. 2006 .

[27] A. Pal , K. Shanmugasundaram , and N. Memon , " Reassembling image fragments ," in *Proc. ICASSP,* Hong Kong , Apr . 2003, vol. 4, pp . IV–732-5.

[28] A . Pal , T. Sencar , and N . Memon , " Detecting file fragmentation point using sequential hypothesis testing ," *Digit. Investig.* , to be published.

[29] G. G. Richard , III and V. Roussev , " Scalpel: A frugal, high performance file carver ," in *Proc. 2005 Digital Forensics Research Workshop (DFRWS)* , New Orleans , LA , Aug. 2005

[30] K. Shanmugasundaram and N . Memon , " Automatic reassembly of document fragments via data compression ," presented at the 2nd Digital Forensics Research Workshop , Syracuse , NY, July 2002.

[31] K. Wang , S. Stolfo , " Anomalous payload-based network intrusion detection ," in *Recent Advances in Intrusion Detection* , (Lecture Notes in Computer Science) , vol. 3224 . New York: Springer-Verlag , 2004 , pp. 203 – 222

[32] W.J. Li , K. Wang , S. Stolfo , and B . Herzog , " Fileprints: Identifying file types by n-gram analysis ," in *Proc. 6th IEEE Systems, Man and Cybernetics Information Assurance Workshop* , 2005 , pp. 64 – 67.

[33] M. McDaniel and M. Heydari, " Content based file type detection algorithms ," in *Proc. 36th Annu. Hawaii Int. Conf. System Sciences (HICSS'03)—Track 9* , IEEE Computer Society , Washington , D.C. , 2003, p. 332.1.

[34] M. Karresand and N. Shahmehri , " Oscar file type identification of binary data in disk clusters and RAM pages ," in *Proc . IFIP Security and Privacy in Dynamic Environments* , vol. 201 , 2006, pp. 413 – 424 .

[35] M. Karresand and N . Shahmehri , "F ile type identification of data fragments by their binary structure ," in *Proc. IEEE Information Assurance Workshop* , June 2006 , pp. 140 – 147 .

[36] Cor J. Veenman , “ Statistical disk cluster classification for file carving ,” in *Proc. IEEE 3rd Int. Symp. Information Assurance and Security* , Manchester , U.K. , 2007, pp. 393–398 **.**

[37] *STORAGEsearch.com. Data Recovery from Flash SSDs?* [Online]. Available: http://www.storagesearch.com/recovery.html

[38] A. Wald , *Sequential Analysis* . New York : Dover , 1947 .

[39] Manning, C. (2002, September 20). YAFFS The NAND-specific flash file system. Retrieved December 21, 2009, from http://www.yaffs.net/yaffs-nand-specific-flash-file-systemintroductory-article

[40] Micro Systemation. (2008, July 1). .XRY system. *Micro Systemation Web site*. Retrieved December 21, 2009, from http://www.msab.com/en/mobile-forensic-products/XRY-Mobile-Version-Forensic-Software/

[41] Miller, R. (2009, June 25). HTC's Sense UI not coming to any "Google" branded phones. *engadget Web site*. Retrieved December 21, 2009, from http://www.engadget.com/2009 /06/25/htcs-sense-ui-notcoming- to-any-google-branded-phones/

[42] Open Handset Alliance (OHA). (2009). Open handset alliance home page. Retrieved December 21, 2009, from http://www.openhandsetalliance.com

[43] Paraben Corp. (2008). Paraben's Device Seizure - Cell phone forensic software. *Paraben Forensic Tools Web site*. Retrieved December 21, 2009, from http://www.paraben-forensics.com/cell_models.html.

[44] Purdy, K. (2009, August 21). Five great reasons to root your Android phone. *lifehacker Web site*. Retrieved December 21, 2009, from http://lifehacker.com/5342237/five-great-reasons-to-root-yourandroid- phone.

[45] Spencer, S. (2009, July 24). Android appliances on the horizon. *PocketGamer.biz Web site*. Retrieved December 21, 2009, from http://www.pocketgamer.biz/r/PG.Biz/Android /news.asp?c=14567

[46] TalkForensics. (2009, September 27). *Andrew Hoog of viaForensics talks about Android forensics* [Audio Podcast]. Retrieved December 21, 2009, from http://www.blogtalkradio.com/show.aspx?userurl=TalkForensics&year=2009&month=09&day=27&url=Andrew-Hoog-of-viaForensicstalks-about-Android-forensics

[47] The Unlockr.com. (2009, November 7). How to: Root your Sprint HTC Hero. Retrieved December 21, 2009, from http://theunlockr.com/2009/11/07/how-to-root-your-cdma-htc-herosprint- verizon/

[48] ZenThought. (2009). ASRoot2 software. *ZenThought.org Web site*. Retrieved December 21, 2009, from http://zenthought.org/tmp/asroot2

[49] Samsung Electronics, "APPLICATION NOTE for NAND Flash Memory", rev, 2, 1999. [Online]. Available: http://www.samsung.com/Products/Semiconductor/Memory/appnote/app nand.pdf. [Accessed: November 29, 2006].

[50] Sandisk, "Sandisk flash memory cards - wear leveling", October 2003. [Online]. Available: www.sandisk.com/Assets/File/OEM/WhitePapersAndBrochures/RSMC/WPaperWearLevelv1.0. pdf. [Accessed: November 29, 2006].

[51] M-Systems, "TrueFFS wear-leveling Mechanism", Technical note (TN-Doc-017). [Online]. Available:www.m-systems.com/NR/rdonlyres/FCC7D817-38A5-4D80-8471-7DA793EA255/0/TN017TrueFFSWearLevelingMechanism.pdf. [Accessed: November 29, 2006].

[52] HDDGURU, "ATA/ATAPI Command Set", [Online]. Available: http://hddguru.com/content/en/documentation/2006.01.27-ATA-ATAPI-8-rev2b/. [Accessed: November 29, 2006].

[53] Samsung Electronics, "Smartmedia Format Introduction (Software Considerations)", 1999. [Online]. Available: www.win.tue.nl/_aeb/linux/smartmedia/SmartMedia Format.pdf

[54] Automated reassembly of file fragmented images using greedy algorithms by Nasir Memon, Member, IEEE and Anandabrata Pal.

[55] Automatic Reassembly of Document Fragments via Context Based Statistical Models by Kulesh Shanmugasundaram kulesh@isis.poly.edu Nasir Memon memon@poly.edu Department of Computer and Information Science Polytechnic University Brooklyn, NY 11201.

[56] Automated reassembly of fragmented images Anandabrata Pal, Kulesh Shanmugasundaram, Nasir Memon Computer Science Department, Polytechnic University, Brooklyn, NY 110201.

[57] Carving contiguous and fragmented files with fast object validation Simson L. Garfinkela,b a Naval Postgraduate School, Monterey, CA, USA bCenter for research on Computation and Society, School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA

[58] Identification and recovery of JPEG files with missing fragments Husrev T. Sencar*, Nasir Memon TOBB University of Economics and Technology, Computer Engineering Department, Sogutozu Cad. No: 43, Ankara 06560, Turkey