Reverse Engineering Alternate Data Streams to Detect the Secret Communication

by

Khaula Khawer

A thesis submitted to the faculty of Information Security Department, Military College of Signals, National University of Sciences and Technology, Rawalpindi in partial fulfillment of the requirements for the degree of MS in Information Security

August 2018

Declaration

I hereby declare that no portion of work presented in this thesis has been submitted in support of another award or qualification either at this institution or elsewhere.

Dedication

“In the name of Allah, the most Beneficent, the most Merciful"

I dedicate this thesis to my parents, my siblings and my endearing nephew Ibrahim who supported and encouraged me at each step of the way.

Acknowledgments

All praises to Allah for the strengths and His blessings in completing this dissertation.

I would like to convey my gratitude to my supervisor, Maj Muhammad Faisal Amjad, PhD, and co-supervisor, Asst Prof Waseem Iqbal for their constant supervision and support. Their invaluable help of constructive comments and suggestions throughout the experimental and thesis works are major contributions to the success of this research. Also, I would thank my committee members; Lecturer Waleed Bin Shahid for his support and knowledge regarding this topic.

Last, but not the least, I am highly indebted to my parents, my cute and lovely nephew, my sisters, my brother and my bhabi. They have always been a source of inspiration for me. I would like to thank them all for their constant care, undying love and continuous support through my times of stress and excitement.

Abstract

With the growing security concerns of the digital world, the forensic investigators and law enforcement agencies need to thoroughly scrutinize the suspected system, leaving no stone unturned. Extracting and recovering the hidden data in a forensically sound manner helps a lot in getting the required information and conducting the forensic investigation with success. NTFS Alternate data streams provide an ideal way to conceal the data within them and hence terrorists as well as adversaries could employ this feature to hide their offensive plans and malicious activities in the ADS of the compromised systems as normal users are unaware of the ADS presence. The factors that further augment the value of the ADS for data hiding are that they require low level of expertise to create and manipulate, are not much prone to suspicion for hiding high level secrets, and are the integral part of the most widely used file system. To reverse engineer alternate data streams for efficient and effectual data retrieval, exploring diverse possible ways of data hiding is essential. The research follows this approach and successfully implements the data hiding concepts of ADS nesting, encoded fragmentation and also the defense-in-depth strategy by applying the compression, password encryption, encoding, fragmentation and ADS nesting concurrently. Also the data concealed through different stated techniques is effectively recovered, keeping integrity intact. A comprehensive route, encompassing all the mentioned techniques, is then proposed in the thesis to effectively analyze and retrieve the ADS and their contents.

Table of Contents

List of Figures

List of Tables

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS/MPhil thesis written by ~~Mr~~/MS **KhaulaKhawar,** Registration No. **00000117523**, of **Military College of Signals** has been vetted by undersigned, found complete in all respect as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial, fulfillment for award of MS/MPhil degree. It is further certified that necessary amendments as pointed out by GEC members of the student have been also incorporated in the said thesis.

Signature: _____

Name of Supervisor Maj (Retd) Muhammad Faisal Amjad, PhD

Date: _____

Signature (HoD): _____

Date: _____

Signature (Dean): _____

Date: _____

Table of Contents

Chapter 1

1. Introduction

We are living in a digital era today where everything is being digitalized including wars, attacks and defense. Security is the main concern of today's digital world. It is the need of the hour to make the digital data secure against all types of reconnaissance, unauthorized

access and damage. From personal to national level, one cannot have any privacy or security in today's world unless necessary security measures are taken to protect the digital information.

Information security is the set of security measures taken to protect the data, digital and non-digital both, from unauthorized access, usage, duplication or destruction. These security measures include both the physical and digital security procedures and strategies essential to avert, detect, document and defy the threats to the data whether in transit or at rest [31].

CIA (confidentiality, integrity, availability) triad is the building stone of information security. CIA makes it certain for the sensitive data (be it in transit, during processing or at rest) not to be exposed to any of the unauthorized party (confidentiality), not to be altered by any unauthorized party (integrity) and be available all the time to the sanctioned parties (availability) [31].

1.1 Data Hiding

Digital data hiding is to save or store the data in the places where it is not likely for the information to be stored. In such a case, where the data is stored in an unexpected place, it needs special skills to store and also to retrieve data from such places. The hidden data, that is covert, misplaced, lost, accidentally removed or so, could be considered as an unintentionally dark data or intentionally dark data. Dark data means the data that is difficult to be analyzed and accessed while the Light data can be easily analyzed. Many data hiding techniques make use of dark data in combination with the light data. In encryption, the dark message generates the light data while the dark data exists within the light data in watermarking. And the dark data coexist with the light data in the alternate data streams of NTFS. There could be unlimited variants for such combinations. [1]

Data Hiding has always been a major part of Computer Forensics as many cases have been solved because of hidden data retrieved by experts. There are numerous methods that can

be used in order to hide data from potential examination. Data hiding is one of the major anti-forensics techniques that will hinder the forensic investigation process.

[2], [4] An NTFS data hiding method is considered to be good, if viewed from the suspect's perspective, if it meets the below mentioned objectives:

a. The prospect that the hidden data would be overwritten is very low.

b. The concealed data is unnoticeable by the normal user.

c. Large amount of data could be concealed by employing the data hiding approach.

d. No error/bug is reported by the normal system check, such as by 'chkdsk' utility.

1.1.1 Data Hiding Categories

Data hiding is classified into following three main categories [30]:

1. **Out-of-Band:** The category in which the concealed information break the design of the format of the medium as it is the part of the medium that dwell outside the format specification of that medium.

Examples include slack space at the end of file/ partition, host protected area.

2. **In-Band:** This part of the medium is within the format specification of that medium; hence the hidden data must live within the boundary specification of the particular format.

Examples include alternate data streams (ADS), reserved but unallocated sectors.

3. **Application Layer:** It is a subset of In-Band data hiding category and in application layer, information is obscured in a higher level format specification.

Examples include steganography, hidden text within documents.

1.1.2 Data Hiding Techniques

Various data hiding techniques are available. These techniques are broadly divided into two categories:

1. *Non-Physical Data Hiding:* Data hiding through the digital methods or forms like cryptography, steganography and watermarking are included in this category. These three techniques differ in various aspects. Cryptography intends to protect the content of the

message from unauthorized access by rendering the contents into an unreadable form however the communication is open whereas steganography aims to hide the fact that the secret communication is being taking place. Watermarking, on the other hand, attempts to establish the copyright information to the message or data. Watermarking provides ownership, license, source and copyright information by adding metadata to the message [3], [42], [43] and [41].

2. *Physical Aspect of Data Hiding:* Physical aspect deals with utilizing the digital storage locations (file systems, hard disks etc.) for the concealment of information or data. The inbuilt loopholes in the system architectures lead to the physical aspects of data hiding [1]. Different data hiding techniques employed in this category includes host protected area, slack space, boot sector, bad sectors, deleted files, deleted/hidden partitions, alternate data streams etc.

1.1.3 Data Hiding in NTFS

Every object in NTFS is considered as a file and can possess any of the attributes assigned to the file. Some of the attributes can be used more than once in a single file. According to the file type, some attributes are essential and not others, but NTFS do not restrict the file to have unnecessary attributes as well. This liberty could be misused for data hiding and without being evident.

In NTFS file system, the hidden data analysis is categorized into three phases. In first phase, it is established if any hidden data exist or not by examining various data hiding techniques. The next phase deals with digging the concealed data out of the hidden place while the hidden files are recovered in the last phase [2]. There are various challenges involved in the analysis of hidden data that includes the data storage in a random order or without any structure, removal of the file signature etc. Moreover, steganographic or cryptographic methods will further hinder the data recovery process. [2]

1.2 Alternate Data Streams

Alternate data streams were introduced in 1993 by Microsoft to make the Windows compatible with the Macintosh's HFS (Hierarchical File System) [2], [35], [39]. In Hierarchical file system, a file is comprised of a data fork and a resource fork. The user's data is saved in the data fork while the metadata about the data fork is saved in the resource fork. So, the Microsoft launched the alternate data streams in NTFS file system for the supporting the HFS' resource fork information. The data fork information is stored in the primary files of NTFS.

The most outstanding feature of ADS is its ability of being attached to the main file stream without affecting its overall size and functionality. Hence, the security implications of this technology lie in the fact that detection of the amounts and types of data stored in additional $DATA attributes is usually difficult to detect [35], [40]. The opponents could employ this implication to communicate their information furtively.

1.2.1 Main features of ADS

Main features of ADS are listed as:

- Alternate data streams of NTFS provide ideal means of hiding large amount of data in an existing primary stream.
- In MFT, the contents of the $DATA attribute can be random and yet remain unsuspicious as it's the only attribute without a precise format.
- In contrast to other NTFS hiding techniques, alternate streams are very easy to create without much expertise at hand.
- Alternate streams, on being attached to a main stream, do not change its size nor do they alter its functionality on the whole.
- Alternate data streams are a fundamental feature of the NTFS that cannot be disabled [35], [39].

- NTFS Alternate data streams do not possess characteristics of their own rather they inherit all the access rights assigned to the default unnamed stream they are associated with.
- Any user having write privileges on a file or directory can use such files/ directories to hide secret data.

1.2.2 Basic Commands with ADS

Some basic commands used with alternate data streams are given below:

Windows Command Prompt commands

type dua.flv > HANDS.3gp:dua.flv

dir /r

"C:\Program Files\VideoLAN\VLC\vlc.exe" "f:\HANDS.3gp:dua.flv"

Windows Power shell Commands

$a = Get-Content f:\MainFile.txt:dua.flv -Encoding Byte -ReadCount 0

Set-Content f:\HANDS.3gp -Encoding Byte -Value $a

Get-Content f:\MainFile.txt -Stream HANDS.3gp -Encoding Byte -ReadCount 0 | Set-Content f:\HANDS.3gp -Encoding Byte

Get-Item f:\HANDS.3gp:dua.flv

Get-Item f:\HANDS.3gp:dua.flv | Measure-Object -property length –sum

Remove-Item f:\HANDS.3gp -stream dua.flv

Nesting of ADS, their usage as a covert medium for high level secrets and reverse engineering of ADS are the focus of this research by taking the audio, video and graphic files (compressed, uncompressed) as the data set. The research will identify ways of extracting stealthy information while preserving the evidence and catering the forensic challenges.

1.3 Justification for the Selection of the Topic

We are living in a cyber world. Cyberspace is ever more becoming a place of threat and danger, susceptible to hacks and cyber combat. The offensive use of cyber weapons

became evident now, leaving behind the idea of them only being defensive. This research will help our Military/Law enforcement agencies in monitoring and analysis of the intelligence shared between adversaries for the purpose of reconnaissance. It will contribute a lot in the forensic investigations while inspecting the NTFS drives, to look for the various dimensions adhere to alternate streams of data.

1.4 Relevance to National and Army Needs

Today's is a Digital world where everything is being digitalized including the wars, attacks and defense. The proposed methodology will help the Law enforcement agencies in two ways. Firstly, in using ADS to hide the high level secrets and secondly, in discovering the information or intelligence concealed in ADS by adversaries or terrorist organizations. It is the need of the hour to wade through all the possible means of communication, which could be adopted by the adversaries, to get hands on them. This research will benefit the military, while dealing with this data hiding technique, in extracting and analyzing the covert information that will be transformed into significant digital evidence.

1.5 Relevance to Forensic and Digital Industry

The devised strategy and the methodology followed in the research will help the forensic and digital industry, in large, for developing such a commercial tool that can fully unveil the secrets and information hidden inside the alternate data streams. It will also help the forensic investigators to seek out the various different dimensions while inspecting the ADS.

1.6 Advantages

The research will help

a) Military/Law enforcement agencies in monitoring and analysis of the intelligence shared between adversaries for the purpose of reconnaissance

b) Law enforcement agencies to beware of the challenges posed by secretly communicating through ADS

c) Forensic experts and investigators for achieving best results by catering different dimensions of ADS while dealing with NTFS drives

d) Researchers in conducting a thorough survey regarding the subject. Plus also help them in selecting a novel idea for their future researches.

e) Academia/Scholars to get out the more in-depth knowledge about alternate streams.

1.7 Areas of Application

a) Military
b) Law Enforcement Agencies
c) Forensic Industry
d) Digital Communication
e) Data Hiding Techniques
f) Anti-forensics Techniques
g) Reverse Engineering

1.8 Thesis Compilation

The research is organized into six chapters; the brief description of the rest of the dissertation is given below.

- **Chapter 2**: This chapter contains the literature review done to compile this thesis write up and all the possible notions and researches done on the topic.

- **Chapter 3**: In this chapter the proposed methodology has been discussed with a complete review and illustration of the various employed procedures in detail.

- **Chapter 4**: This chapter contains experimental results carried out and findings noted.

- **Chapter 5**: This is the concluding chapter that draws a brief conclusion of all the work done so far and future possibilities that can be carried out.

Chapter 2

2 Literature Survey

The section discusses the prior works done in the field of data hiding, especially in reference to NTFS data hiding techniques. Plus the published works related to the alternate data streams are also elaborated in the chapter. It also touches the reverse engineering concept and some tools and techniques used by analysts for reverse engineer a system, device or an object. The chapter gives a fair idea to the readers regarding how much work has already been done in the said domain and how is the current research different from them. The section discusses the prior works in a way that makes the readers understand the new strategy and approaches, the current study bring forward.

The preceding literature related to the data hiding techniques lay out various methods of hiding data in NTFS, both physical (related to disk structures and file systems) and non-physical techniques (like cryptography, watermarking, steganography). Plus different ways to use alternate data streams, as detailed in literature review, is also talked about. Main focus is on the potential vulnerabilities associated with the NTFS alternate data streams. The concept of reverse engineering and the methods used by the process are also outlined by the end of the chapter.

2.1. Data Hiding

In [1] and [45], the authors termed the ‘digital data hiding’ phenomenon as old as the computer system themselves. Data hiding is defined as hiding the digital information in the locations where they the data is most unexpected to reside in. The study is focused towards the intentionally hidden/ dark data and the data hiding schemes for Windows and Unix file systems. Due to the continuous enhancement of the sophistication related to forensic expertise on both sides of the law, i.e., by criminals and investigators, digital data hiding is turning out to be the ever more important topic.

Data hiding is the tactic that takes advantage of the fact that the forensic investigators have to analyze the today’s hard drives that have the capacity to store enormous amount of

digital data, in the limited amount of resource and time on hand [6]. Whereas the data hiding has the shortcoming that an adept and diligent investigator might discover the covert data due to the fact that the data resides on the evidential system or network even after being concealed [7].

Authors in [6] stated various techniques of hiding data in a way that it remains unnoticed despite being residing on the system. They mentioned relocation of data to a place that will be bypassed by the examiners. To make the data invisible is next mentioned in the research in which the fact that the stealth data existed on the system, is concealed. Steganography and alternate streaming are both the methods to make the digital data invisible. It is also mentioned in [41] that steganography could be employed to further make the encrypted data secure by making it invisible as the encrypted data draws the unnecessary attention to itself. Thirdly modifying the file extensions also is the method of hiding digital data. The second method of making the data invisible through streaming is employed in the current research and various ways of data covertness is explored.

The authors also highlighted the need for forensic investigators of keeping updated regarding the bugs and vulnerabilities of the counter forensic tools as well as of the in depth understanding of the varied tactics employed by the nefarious users.

It is described in [1] and [45] that the hidden data can be found at each of the level of nested data structure and/ or the geometric structure of the hard drive, for example, hard drive, file system, file, record, and field or a partition. The stealthy data could possibly be existed in the areas generated by these organizations on the secondary storage devices. Various digital disk warrens or data hiding techniques include host protected area, slack space, boot sector, bad sectors, deleted files, deleted/hidden partitions etc.

It is mentioned that getting the understanding and knowledge of the possible data hidden methods indirectly means of knowing to discover the concealed data. This very approach is

used in the current dissertation to reverse engineering the NTFS alternate streams by exploring diverse methods of data hiding through them.

The authors in [1] talked about the digital data hiding methods associated with the file systems while do not took the non-physical data hiding techniques such as compression, encoding, encryption, watermarking and steganography into account. Cryptography, steganography and watermarking are briefly discussed and compared in [1], [41] and [42], [43]. These non-physical techniques of concealing the information possess their own associated difficulties in analyzing them. On the contrary, all these non-physical procedures of data hiding are utilized in the current study.

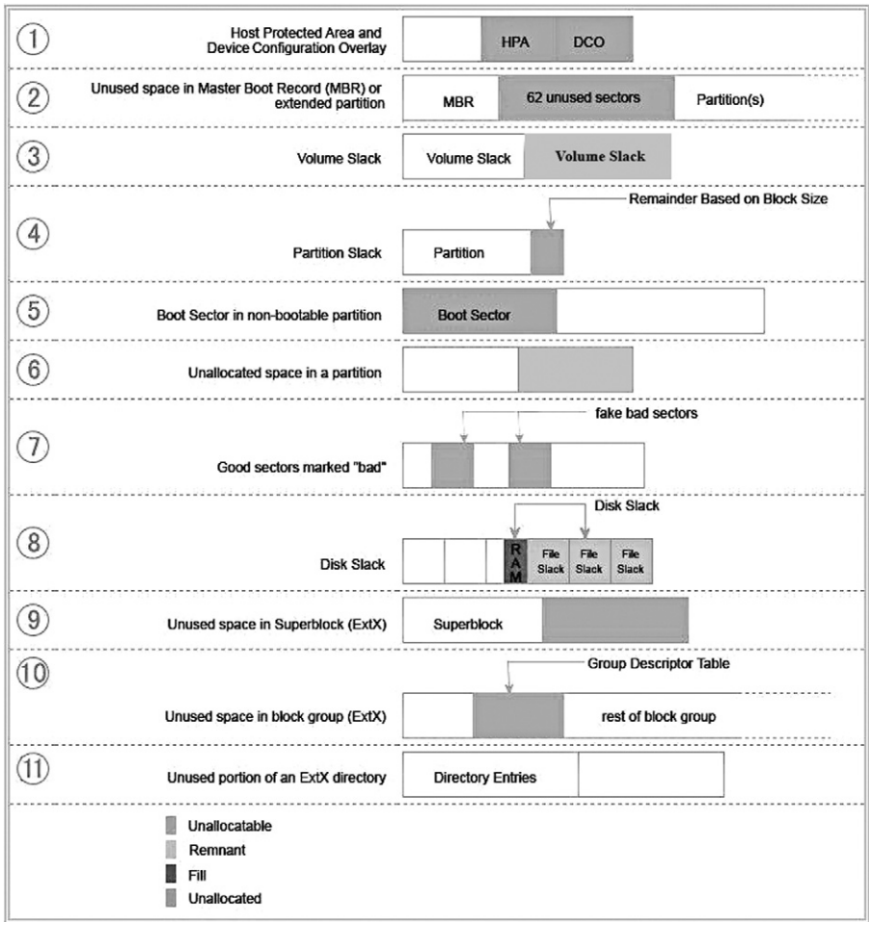


Fig. 2.1 – Digital disk warrens [1]

The paper also commented on the use of the commercially available forensic tool for discovering the hidden data and stated that different tools are effective in different ways and all the tools could not discover all the types of covert data concealed through every

method. Hence the use of more than one forensic tool is suggested so that the investigation could be termed as forensically sound.

2.2. Data Hiding in NTFS

2.2.1. NTFS Background

Everything in NTFS file system is considered a file including the metadata of the file system. On a disk volume, an NTFS structure is organized in four logical blocks [2] as depicted in the below figure:

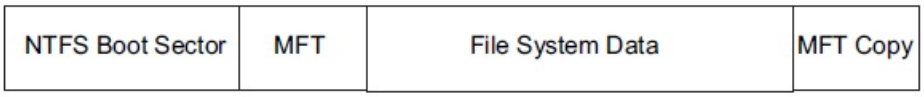


Fig. 2.2 – NTFS volume structure [2]

Master File Table, MFT is the base of NTFS. Every file in NTFS has one entry in the MFT at minimum and each entry in the MFT is called a file record. The MFT file record has a default size of 1024 bytes, out of which some are reserved for the header while other are used by its attributes [2], [44].

An attribute contents can be resident or non-resident. The resident contents are saved within the boundary of the MFT file record or entry whereas additional clusters are allocated to store the non-resident contents of an attribute [36], [44]. These added clusters are saved as the cluster chain/ run in the run list linked with an attribute [4]. Cluster is the smallest unit allocated on the disk space and is termed as ‘data unit’ in NTFS [2].

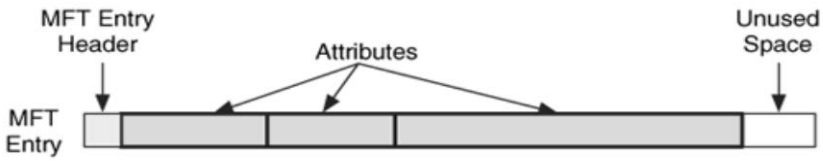


Fig. 2.3 – Structure of MFT entry [5]

2.2.2. NTFS Based Data Hiding Mechanisms

Various probable ways of data hiding in NTFS as well as the analysis methods for the discovery and retrieval of the covert data is discussed by the authors in [4]. The study is focused towards the criminal activities to hide data by employing the NTFS data hiding methods. Also the main aim of the criminals to hide their concealed data from forensic investigators is mentioned. Ineffective data hiding methods are also stated in the paper along with the information hiding techniques regarding volume slack, file slack, bad clusters, boot file, additional clusters, and alternate streams.

On the whole, there are two main phases of analyzing the data concealed within the NTFS. The first phase is concerned with the detection of the presence of the covert data. Occurrence of stealthy data is discovered by checking the system for inconsistencies and abnormalities and if found any; mark them for the analysis phase.

The second phase of hidden data analysis is to recover the covert information. The main challenge faced by the data recovery or retrieval is the storage of the data in the file system without any specified format or structure. Plus the removal of file signatures and the data hidden in random order will further add the complexity.

The hidden data analysis methods discussed in the paper require a lot of time in recovering the stealthy information without having the automated tools at hand. At the time of the research, only the tools for analyzing and searching the ADS were present but not for any other NTFS data hiding mechanism.

Just a few of the probable ways of data hiding in NTFS are stated in [4]. The new methodologies continue to emerge and evolve with the time and it all is relied upon the vision of the nefarious users. The flexible nature of NTFS makes the hidden data analysis in NTFS file system even more intricate. Due to the fact, there are numerous methods of hiding the secret information and their validity is not certain without a publication.

The paper [2] organized the effective mechanisms of data hiding into two main categories that are the slack space based NTFS data hiding techniques and the information hiding methods specific of the NTFS file system. The NTFS specific techniques for concealing the data are further divided into metadata files based methods and the data file based procedures.

The digital data hiding in $BadClus file, $DATA attribute and $Boot file is discussed in the metadata files based mechanisms. Also the risk associated with the manipulation of the metadata files, for concealment of information, is talked about in the research. No attribute offers much potential of data hiding apart from the $DATA attribute, as it's the only attribute that can be made non-resident with certainty.

The data files based data hiding methods examined three techniques of hiding the sensitive information in the $DATA attribute of MFT. These techniques include the data concealment in alternate streams of data file, in the ADS in the directories and the information hiding in the added clusters of the unnamed default $DATA stream.

The research concludes with categorizing the investigation of covert data in NTFS into three phases. In first phase, it is established if any hidden data exist or not by examining various data hiding techniques. The next phase deals with digging the concealed data out of the hidden place while the hidden files are recovered in the last phase [2]. There are various challenges involved in the analysis of hidden data that includes the data storage in a random order or without any structure, removal of the file signature etc. Moreover, steganographic or cryptographic methods will further hinder the data recovery process. [2]

2.3. Alternate Data Streams

Alternate data streams are a feature that is unique of NTFS. They are brought in to make the Macintosh's Hierarchical File System (HFS) compatible with Windows NT servers.

ADS were introduced in early 1990s to support the information stored in HFS' resource forks i.e., the metadata of the data fork/ files [35], [2], [39].

More than one $DATA attribute can reside in an MFT entry (file record) in NTFS. First $DATA attribute is the default unnamed stream. Any additional $DATA attribute is the ADS and every ADS should be named [2], [36], [34]. Alternate data streams could be used for storing metadata for files or directories but they may well be used for data hiding owing to the fact that most of the system utilities analyze the first default $DATA attribute of an MFT record while ignoring the additional unnamed streams [2], [40].

Moreover, almost all the attributes of MFT entry have an implied format and integrity of the system may have compromised by any exploitation of these attributes' contents which makes all such attributes unbefitting for information hiding. Only the $DATA attribute do not have any specific format and could have random contents. ADS/s is a fundamental feature of NTFS and can store a huge amount of data while still being unsuspicious. All these traits of ADS make them most appropriate for camouflaging the secret data.

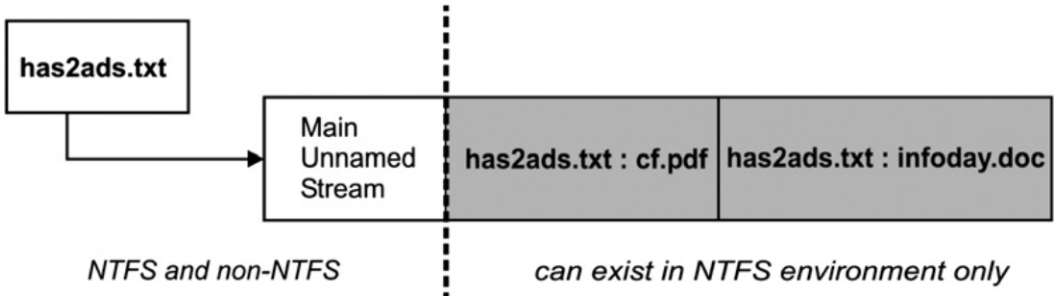


Fig. 2.4 – ADS visibility in NTFS and non-NTFS environments [2]

Two additional $DATA attributes are shown attached to the main unnamed stream (has2ads.txt) as ADS.

Palmgren in [8] stated the following points:

- Alternate data streams can be attached to a main stream file or to a directory.
- Multiple ADS/s can be added to a single directory or a file.

- The operation of the default primary file remains unaffected with the addition of any number of the alternate streams.
- Alternate data streams will be moved or copied to the location with the default data stream it is attached to, as far as the move or copy operation is within the NTFS file system.

Cook in [18] questioned some of the decisions made by Microsoft regarding the ADS implementation. The first thing the author debated about is the capability of the alternate data streams to have any size and of any type without any restriction. All this resulted in a lot of encouraged the hackers and criminals to use the ADS for hiding their illegal activities.

The second decision about which the author argued is the invisible nature of the alternate streams to the standard Windows commands and tools. Neither the size nor the functionality of the default stream is modified with the added ADS/s [10]. Palmgren [8] also mentioned that the only noticeable difference by adding alternate streams is the modification time and date of the default unnamed stream. Even the file hash of the default primary file remains unchanged.

The only limitation of the amount of data that could be concealed in alternate streams is the total size of the NTFS media. Furthermore, ADS are comparatively easy to create and add through Windows command prompt.

2.3.1 Auditing Alternate Data Streams

The author in [18] made suggestions regarding the auditing of the ADS/s. He stated that verifying the alternate streams as malevolent or benign is the first step after getting hands on all the alternate streams of a drive or directory. The alternate data streams used by the genuine applications or programs are normally of very small size, hence he termed it safe to overlook the ADS/s having size of 256 bytes or less while examining the system for ADS' presence.

Plus the authors mentioned that the file types of the default unnamed stream, having ADS/s is to be looked at because some files are likely to have a metadata file attached to it as ADS. For example, an image file is likely to have a thumbnail in ADS.

The author also described that it is extremely unsafe and would raise suspicions if an alternate streams is added to the root directory like C:\, D:\ drives etc.

Chung in [19] and Marc Ochsenmeier in [39] stated some of the legitimate uses of ADS that are to store the keywords associated with a file, associating sounds or fonts to a file, adding summary information of a file in ADS, providing a thumbnail of an image in an ADS, saving the document summary information with the file, having Mac OS' icon types in ADS and allowing the ADS to have the favicons to help recognizing a specific website.

2.3.2 Vulnerability Assessment of ADS

It is still unsure whether the NTFS alternate data streams are a threat or a feature. It has both pros and cons associated with it [3]. The main disadvantage is that it is often being overlooked by the standard windows commands and utilities.

Four major vulnerabilities are focused in the research to better understand the security implications of alternate data streams [9], [16]. Each of these vulnerabilities is discussed with respect to various studies in the existing literature.

2.3.2.1 Virus Attacks

Alternate data streams can also hide executable files (.exe, .vbs, .bat, .cmd) in them and the invisible nature of ADS would help in thwarting the detection of the concealed viruses [9]. The study described that most of the antivirus software dealers only scan the default unnamed stream of the file. It is suggested to load the ADS into memory before their execution in order to get them detected via the real time scanning of the file system.

Berghel and Brajkovska in [12] stated that the tree structure of ADSs has a maximum depth of 1 and hinted this limitation to be shortsightedness. This hint is taken into consideration in the current study and successfully concluded that nesting of ADS is feasible, once the ADS gets compressed. Berghel and Brajkovska further discuss the use of ADS for storing binary executables and the necessity of stealth to deter the detection of hidden malwares, in the form of executables. They further termed the planting of ADS with risky executables as "file-phishing" because it will execute the hidden malwares behind the innocent looking applications. The research also argued about the security implications of ADS and concluded with the ADS vulnerability to be overstated. It discusses that ADS can be used for hiding malwares or as a covert channel but could not do that much harm as many other IP level or lower levels techniques. The central reason that creates alarms with the name of ADS, according to the authors of [12], is the hidden nature of ADS. The authors conclude their study by stating that the actual potential of ADS could never be known owing to the negative claims associated with the ADS vulnerability.

The current research work, on the other hand, discovers much more potential of the alternate streams and makes them appear more vulnerable, especially when coupled with other digital data hiding mechanisms such as encryption, compression, steganography, encoding and fragmentation.

The article [13] discussed ADS in general and the related potential risks. Also the recommendations are given by the author to mitigate the impact of the threats ADSs have on a system. As ADS is an integral feature of NTFS coupled with the fact that the size and number of ADS/s added to the default visible file cannot be restricted as far as the user has access rights to the default stream, guarding a system from alternate streams misuse is almost impracticable in an NTFS environment. A method to monitor the changes in a file system is proposed in the article. Keeping a database of file checksums for tracking a file system for any modification is termed as the most effectual solution for this. The real menace linked with the alternate streams, according to the author, is that they are relatively

unknown but the threat could be reduced by applying and implementing the well-chosen defense practices.

Mahajan and others in [9] discussed the ways a nefarious user can adopt to compromise a victim's system by hiding their malicious codes and activities through alternate data streams. ADS can be used by hackers and criminals for execution of the covert communications, running FTP servers and illicit chat rooms on a system once it got compromised. The tools, backdoors and rootkits that the hacker could use for causing further harm after compromising a system also can be concealed within the alternate streams. An ADS tool is presented in the paper for the creation, detection, execution and deletion of the ADS/s. The methods for all these operations of ADS are also explained.

The proposed system [9] illustrated how a backdoor could be used to remotely access a victim's system in the future by hiding it in the alternate stream attached to a legitimate naive default stream.

msfpayload windows/meterpreter/reverse tcp LHOST=192.168.18.132 LPORT=4444 R I msfencode -e x86/shikata ga nai -c 1 -t exe -0 lroot/file name.exe

Metasploit frameworks' msfpayload is used for backdoor creation in the study as depicted in the above command.

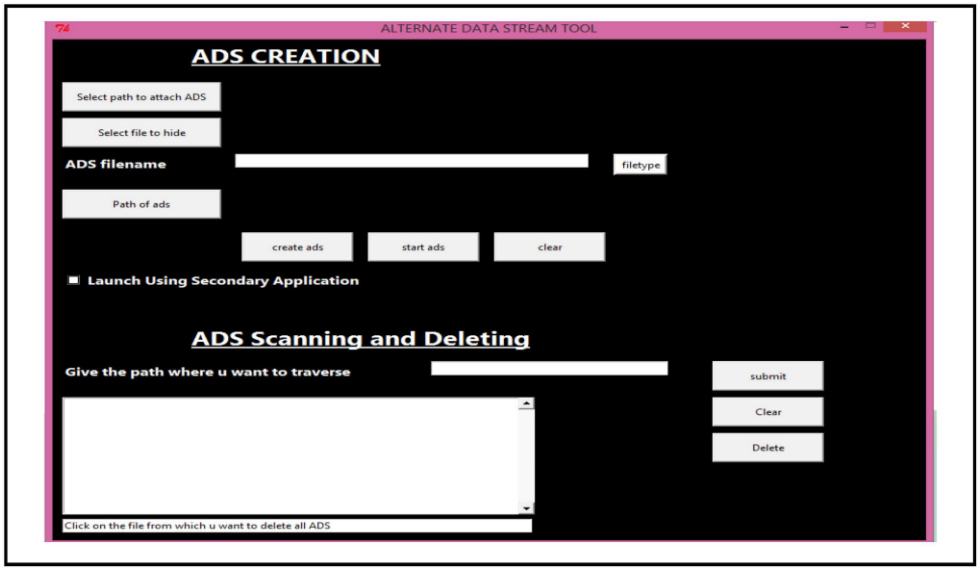


Fig. 2.5 – Alternate Data Streams (ADS) tool [9]

Only one time access is required by the attacker to hide the backdoor in ADS and to write a simple BAT script that will activate the concealed backdoor every time the victim's system starts up. In this way, the hacker always gets system level access of the compromised system on a system startup.

In [14], the authors came up with the very useful idea of the compressed ADS. The main focus is on documenting the compressed ADS, their attributes regarding data hiding and to explain its nefarious use once created. The paper also brought forward a forensically sound way to detect and manipulate the compressed ADS.

Compressed alternate streams have some new attributes, specific to their compressed form besides having inheriting all the previous ADS attributes. These additional attributes are listed below:

- Compressed ADS can b stored in non-NTFS environments (storage devices, file systems).
- Compressed alternate streams could be transferred over the internet via email, ftp, messengers etc.

The compressed ADS are employed in the current research in ADS nesting and other data hiding tactics. The current study also verifies the claims made by the authors in [14].

Two different computer forensic segments have been affected by the compressed ADS. Concealed files data storage and the likely evidence (ADS) in the file systems in combination with the characteristics of the application layer (compressed hidden files) is the first section. The other segment is the digital data hiding in order to protect the information from illegal intrusion.

The research [14] further presented a realistic case to highlight the threat posed by the compressed alternate streams. It is showed that attackers could exploit the compressed

ADS to spread viral software as the current antivirus and forensic tools cannot detect the virus in compressed ADS.

Ryan in [15] gave a detailed knowledge of the NTFS alternate data streams and also compared them with regular files and directories. After discussing quite a few malicious uses of ADS, the author then analyzes different Microsoft tools as well as third-party softwares and measures their efficiency against searching and manipulation of the alternate data streams. A set of Windows shell extensions are presented in the study to incorporate the use of ADS with the normal Windows search and therefore had brought the NTFS alternate streams out of the shadow into the forefront. Many probable data hiding techniques are presented in the current research work using ADS and it is stated that despite being brought the ADS into the light, much potential lies within them for concealing the secrets, even of extreme importance.

Damon Martin in [16] recommended the enabling of real-time virus scanning on systems to be the best existing defense. It is described that the system can examine a virus hidden within ADS through real-time scanning method. An antivirus on the system must shield the system against a virus attack that has a known signature. But the author termed the real-time virus scanning option as impractical considering the resources utilized by this scanning on majority of the production systems.

The author [18] stated that the uses of the NTFS alternate streams are shocking. ADS provide an ideal location to hide any information or piece of software. The tools used by the hackers or attackers to take over a system can be obscured within the alternate data streams for further manipulating and damaging the very system they are concealed in. This will greatly help the system breaker by lessening the required bandwidth for visiting the compromised system over and again.

The author described that it's a common practice by the crafty computer criminals to secrete the potential evidences of their nasty activities within NTFS ADS of an innocent's compromised system.

2.3.2.2 Backups

By studying the NTFS alternate data streams and their behavior, it is established the backup systems for file systems ignore the attached ADS files while taking backups and only backup the default unnamed streams. ADS being the NTFS specific feature are the main problem in backing them up. All the ADS will be removed once the NTFS file system backup is stored on a non-NTFS environment. This will hinder the forensic investigation in acquiring all the required likely evidences.

Bem and Huebner in [11] described that Windows backup utilities are only partially compatible with NTFS alternate data streams despite the fact that they are intended to work with the NTFS file formats. Discovering the alternate streams in the NTFS file system through the computer forensic tools are comparatively easy in contrast to their probing in their backups.

The authors organized the backup utilities in two main groups after studying the various available backup utilities. These categories are ADS aware and non-ADS aware. The second group is in which the ADS got lost during the backup. It is also discovered that different forensic tools act in different ways and differs in the information they provide about the alternate streams during backup and the restoring process and mentioned that they often lose the data in transit.

A new classification of the backup utilities or softwares is proposed in the study [11] keeping in view the behavior of ADS in backups and restoration process. The implications of the proposed classification towards the forensic examination of file system backups are also talked about. An investigator is required to restore the backups to examine the contained files as most of the backup utilities apply compression on the data while taking

backups and the ADSs treatment by backup software is of specific importance to the forensic investigators. There is a chance of losing the critical probable evidence by applying an incorrect or wrong approach for backup.

The authors grouped the proposed classification into five classes: class 0 to class 4. A basic test is described in the paper that could be adopted by anyone to test any backup utility against any of our class. In the end, a forensic methodology is suggested that would help the examiners and investigators to handle the backups carefully so that the data loss can be evaded.

Below fig. 5 shows the ADS-aware application that is fully aware of ADS presence in any environment. Hence, ADS can be restored or backed up to both NTFS and non-NTFS environments.

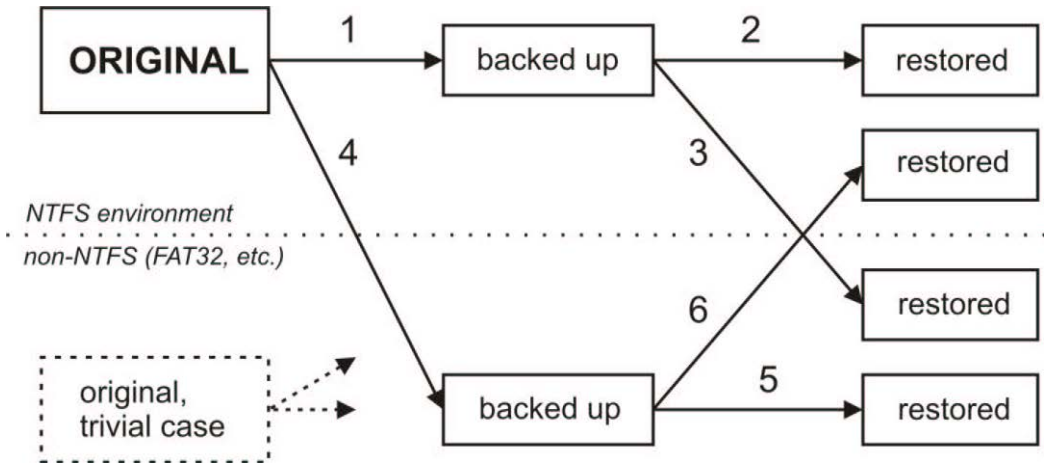


Fig. 2.6 – Backing up and restoring data [11]

It is mentioned in [16] that ADSs can be stored in any of the device or structure having NTFS file system let it be a hard disk, Jaz drive, Zip disk or an email attachment that is saved in an NTFS partition. And the data along with the hidden streams can be moved and copied between the NTFS volumes. It is also true in contrary. All the ADSs will be removed the moment they are moved or copied to any of a non-NTFS environment. It has

both plus and minus associated with this implication. To get rid of the alternate data streams, this implication offers a straightforward approach which is a plus. Conversely, the minus associated with this inference that the ADSs might be ignored by the backup software. There is also a probability that the backup data is saved and restored on a non-NTFS device or structure. In such a case, the alternate data streams are not backed up.

The forensic challenge that the complete backups of the compromised system may not be available is also stated in [16]. The significance of having the satisfactory backups of every NTFS device and structure is highlighted in the certification paper by explaining that tracking the modifications to the NTFS file systems is essential to have reliable backups so that the malicious activities hidden through ADSs may not be missed by the forensic investigators.

2.3.2.3 Denial of Service Attack

Denial of service attack can also be performed by exploiting the alternate data streams of NTFS. One type of DOS attack employing the ADS is to get the system drive or a partition filled with the alternate data streams by attaching a large number of them to a default stream. Consequently no space will be left on the drive to store the paging files and hence the server will get collapsed.

Attaching more than six thousand hidden alternate streams to a default visible stream is also an example of the denial of the service attack. The system stops responding whenever such a file is requested to be accessed to which more than 6000 ADSs have been attached and if the system does respond, the response will be excessively slow.

Ryan in [15] explained that a completely filled partition or a volume of a drive always causes instability of operating system and application. A system administrator is incapable to trace a file having extra large attached ADSs in the absence of a third-party ADS tool and it became more unfeasible in the case of an ADS attached to an exclusively locked file.

In such a case, ADSs cannot be found. The only solution left with is to mount the volume on a separate machine and then perform the ADSs scan.

2.3.2.4 Data Hiding

Hiding the digital data is the major apprehension related to alternate data streams. The real menace is the difficulty to detect the hidden attached ADS based on the fact that the size of the default visible file will remain unchanged. Network and system administrators should employ different available ADS tools to detect the concealed alternate streams.

The authors in [23] stated that preserving the secrecy of the NTFS alternate data streams became questionable with the effectual ADS detection and manipulation tools on hand. Also antiviruses exist in the market to sense the presence of malicious software within the ADS. This being said, the study brought forward a new approach of hiding data within the ADS without having fright of being caught by applying an external encoder to the alternate streams. The authors claimed that the fresh method will improve the data confidentiality to the extent that will bring the ADS back into action. The current dissertation validates the authors' claim of highly increasing the stealth by adding an external encoder to the ADS along with presenting the other ways of data hiding to further lessen the ADS detection prospects.

It is further said in the research that the tool developed in [23] will help the users to hide their sensitive information within the alternate streams by applying encoding to the data.

The proposed research will carry forward this approach of encoding of the hidden data and will add further stealth by combining different digital data hiding strategies [23].

ADS description, destructive nature of ADS and its execution techniques are discussed in the said chapter of the book [24]. It mentions various ways of hiding data in alternate streams that will be explored in the research in order to validate these possibilities. The book mentioned that in order to deter the detection of hidden data in ADS, the binary files

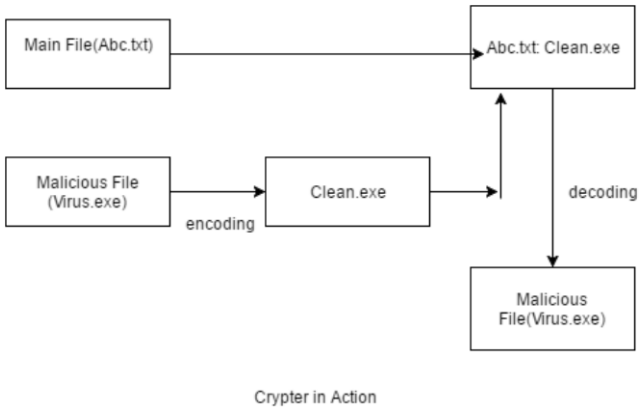


Fig. 2.7 – Encoding and decoding of an ADS file [23]

can be stored in multiple alternate streams only to combine when needed. This fragmentation of binary files is evaluated in the proposed research. Also the streams creation and manipulation without extension identifier is experimented. These methods are then used in combination of various other hiding strategies to further complicate matters.

In [22], the authors discussed that as the ADS data gets available on the task manager since the Windows 2000, it is not any more difficult to supervise these streams. It is further stated that using alternate data streams for data concealment is not worth it any longer with the presence of various tools and techniques for identifying and managing the ADS offered by both the Microsoft and the different researchers. But the proposed research will put forth the various ways that illustrate that the potential for data concealment within ADS is still high and the alternate data streams may contain much more sensitive data than is expected to be resided in them.

Sharma, Singh and Goel mentioned various NTFS data hiding methods in [20]. The study is more focused towards the alternate data streams of NTFS for data concealment. In NTFS, the first unnamed stream is the default one whereas all the additional streams, attached to the default stream, is the alternate data stream.

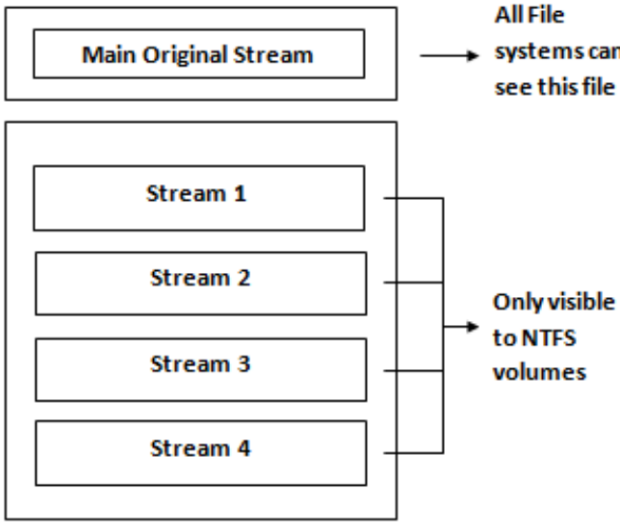


Fig. 2.8 – NTFS Support for Multiple Data Streams [20]

Figure 2.8 shows the attachment of four added ADSs to a single default stream and the visible file (default stream) remains unmodified both in its operation and size.

It is implied by the authors in [21] due to the potential vulnerabilities attached to ADSs, forensic investigators cannot ignore them while analyzing and examining a system.

"ADS Examiner Tool" is developed in [21] to detect the data concealed within the ADSs. The authors identified the suspected locations ADS can use for hiding the information and then based on these locations, five functions have been developed to search for the presence of alternate data streams in different locations. The data could also be mined out of the ADS/s. The tool also facilitates the forensic investigators by offering the support for the detection of ADSs in the deleted files and also in the disk images. The finding of the alternate streams in the disk image is handy in the post mortem forensic analysis of systems.

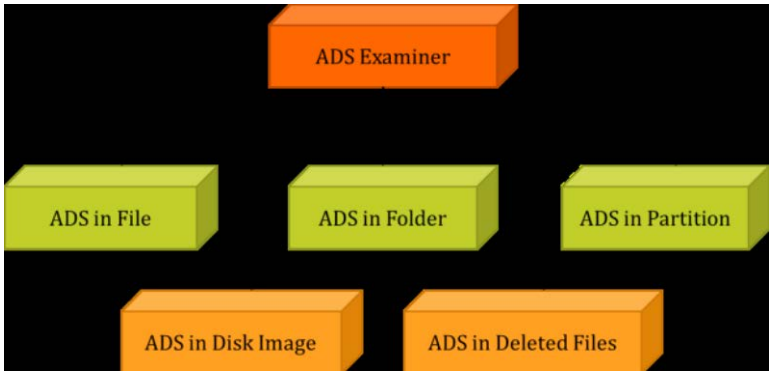


Fig. 2.9 – Block diagram of ADS Examiner tool [21]

Practical aspects of hiding various types of file are explained in [25]. Sample command line examples are mentioned in the article to help the readers start with creating ADSs and for hiding executables and video files within. This practical guide is very helpful to begin with the manipulation of ADS. The article also discussed some ADS tools, like LADS and Streams, to hunt for ADSs in the entire directory structures and hard disks.

2.4 Reverse Engineering

Reverse engineering is the process or set of processes employed to unravel or determine the ideas and procedures involved in the production by thoroughly investigating a product or process. In the context of software, reverse engineering could be used for the recovery of the source code, for fixing a bug, to identify malwares, for understanding the working of a programs and much more.

Authors in [27] mentioned that to alter or modify a system is never the aim of the reverse engineering. Rather it deals with reproducing the information by thoroughly examine the used methodology. The insight to the system by recognizing the involved segments and determining the relationships between various segments or components of a system is a prerequisite to regenerate the abstractions of the retrieved information. The same procedure is adopted in the current research to retrieve and recover the information concealed in the alternate data streams.

Yet another way to [28] define reverse engineering is that it is a procedure that analyzes the operations and framework of a system, device or an object to discover its technological principles.

The author mentions various motives behind reverse engineer a system. It could be to patch binary and change system behavior, finding vulnerabilities, discovering trade secrets, analyzing protocols, dodging protection mechanism, academic research or also could be the pure curiosity [28].

The study [26] stated that there exists no specific process to break through a program. Various methods are adopted by the crackers or reverse engineer to penetrate a system and the type of the process or device and the kind of security mechanisms applied to it provide the basis to employ a particular course of action. Some of the commonly used tools are also mentioned in the paper that are helpful in the said domain that include dissemblers, debuggers, Unpackers, Hex-Editors, File analyzers, File and Registry Monitors. WinHex is a Hex-editor used in the current study for analyzing the presence of NTFS alternate data streams.

Hex editor searches for the content of a binary files as well as executables. Hex editor can add new content in the binary form as well as may edit it legally. The hex editing tools are available in the market that aid in quickly getting hands on the areas of interest [29]. Such programs are easy to use and anyone with basic understanding of the programming could also utilize it. It further emphasizes that to reverse engineer a process, diverse tools and procedures regarding the technology should be employed.

Reverse engineering is termed as both a science and an art [17]. Different high level tasks, performed by the analysts or examiners while reverse engineering diverse kind of files, are categorized in the research. The categories include analyze, calculate, compare, filter, identify, locate, modify, report, view etc. All these are again sub-divided into various tasks. For example, the in calculate class, the sub tasks are binary, hex or decimal calculations,

encryption, decryption, encoding, compression, fragmentation, and to calculate the hashes. These tasks are also employed in the current study to reverse engineer the alternate data streams.

Literature related to the alternate data streams, digital data hiding and reverse engineering is thoroughly surveyed in the chapter to give a detailed insight of these concepts to the readers. It is also discussed that how a particular study relates to the current research and how a certain idea proposed by a specific publication is carried forward. After reviewing the former works on the said topics, the proposed research methodology is described in the next section of the dissertation.

Chapter 3

3 Proposed Research Methodology

The research study explored new dimensions for hiding digital secrets in NTFS alternate data streams and then devised a strategy to reverse engineering the ADS to get hands on the probable secrets concealed within. ADS had long being used for data hiding purpose but owing to the development of commercial tools that can easily detect ADSs presence, delete them and search for the anomalies resided within, ADS has since being considered not of much advantage in the field of data hiding. The data hiding potential of alternate data streams is explored in the research and it re-strengthens the notion of concealing high level secrets within ADS without the fear of them being revealed.

Furthermore, discovering ways to hide the digital data within ADS will offer new dimensions while reverse engineering the concealed data in ADS. The research devised a strategy for reverse engineering the alternate streams by taking, all the probable paths to hide data, into consideration.

Nesting of ADS, ADS usage as a covert medium for concealing high level secrets and the reverse engineering of ADS are the focus of this research ***by taking the "audio, video and graphic files (compressed, uncompressed)" as the data set***. The research will identify ways of extracting stealthy information while preserving the evidence and catering all the forensic challenges.

3.1 Adding Covertness to ADS

The research work proposed the following fresh ideas that will add the covertness to the data hidden in ADS:

- Nesting of ADS
- Fragmenting the binary files to add secrecy by adding the fragments in multiple streams

- Using ADS with steganography technique
- Adding defense in depth by combining various hiding techniques such as compression, encryption, encoding, fragmentation and ADS Nesting

Each of the above mentioned data hiding method will be discussed in detail in the chapter. The section put lights on the concept of the technique; the logic used while implementing a specific method and then analyzes the course of action, adopted for the implementation of a process.

3.1.1 ADS Nesting

The research introduced the notion of Nesting of ADS, i.e., hiding ADS within ADS. Prior to this work, ADS was considered to be of depth 1. The study adds depth to ADS by successfully implementing the nesting of ADS within ADS up to level 3. The proposed idea will hinder the forensic examination of ADS further by adding defense level to the secret data hidden behind ADS.

Command prompt command "**type**" is used for attaching ADS to a file whereas "**Get-Content**" function of Windows Power Shell is used to retrieve back the ADS as main file.

3.1.1.1 Procedure

Following course of action is taken for nesting of ADS and their retrieval:

1. Primary File is selected.
2. Another secondary file, with secret data, is selected.
3. Secondary file is added as ADS to the primary file.
4. Primary file with ADS is then compressed through WinRAR, with streams options checked.
5. Another Primary file is selected to hide the compressed file (containing ADS).
6. The compressed file (containing ADS) is then attached as ADS of the second primary file selected.
7. The uncompressed file with ADS (in 3) is also added as ADS of the second primary file.

8. Both the ADSs (compressed, uncompressed) are then retrieved back.

9. For the uncompressed ADS file, check the file through WinHex and also through the “dir /r” command.

10. ADS file, concealed within the uncompressed file, is not retained.

11. For the compressed ADS file, first step is to uncompress the file.

12. Then by traversing to the uncompressed directory, check the first primary file through WinHex and through the “dir /r” command.

13. ADS file with secret data, concealed within the compressed file, is retained successfully.

For nesting up to further levels, the second primary file with both ADS (compressed, uncompressed) has to be compressed and then added as alternate stream of another primary file and it goes on successfully. The level of depth to be added for ADS nesting depends upon the algorithm or approach one is using for data hiding.

3.1.1.2 Flowchart

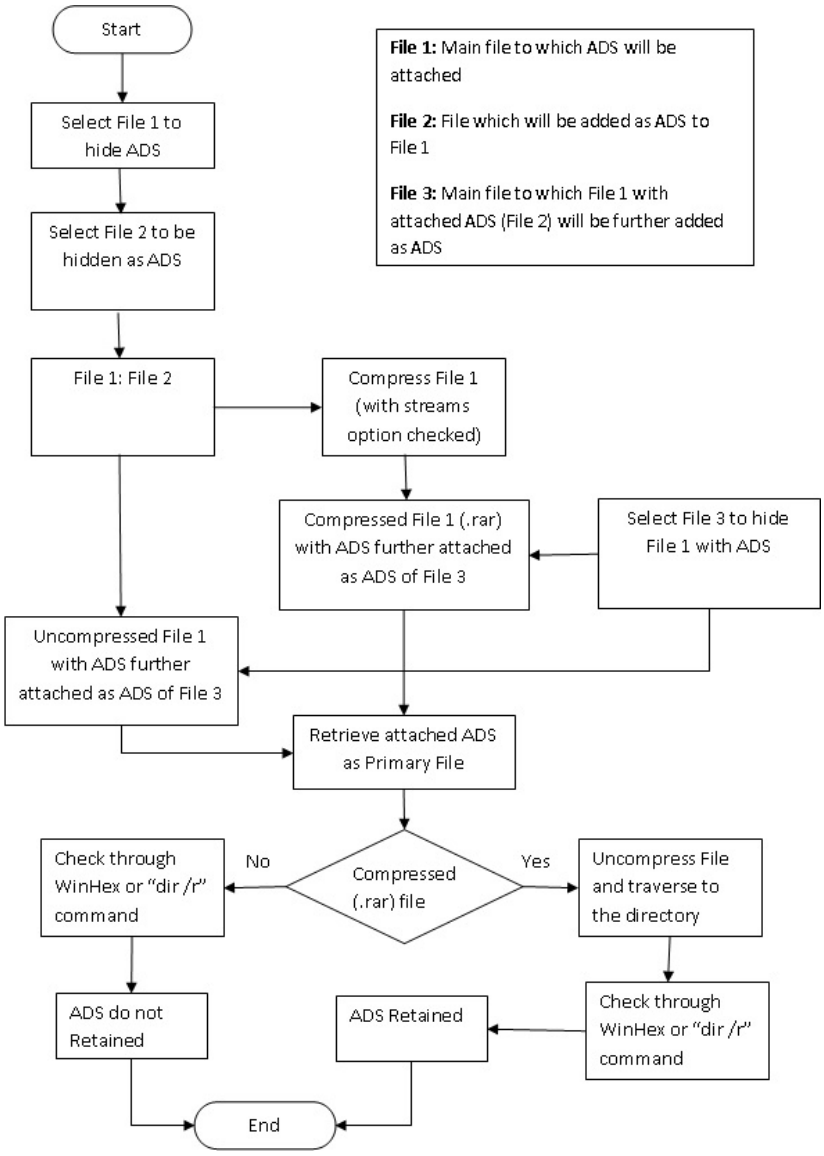


Fig. 3.1 Flow chart for ADS Nesting (ADS within ADS)

3.1.1.3 Analysis of the Procedure

ADS' nesting is only possible with the compressed file. Unless the file having ADS is not compressed, it couldn't be nested further.

Nested ADS successfully adds stealth to the ADS. The nested alternate streams can neither be detected by windows search/commands nor by any commercially available ADS tool.

Following commands are used for getting the contents of the alternate stream of depth level 1.

Get-Content f:\ HANDS.3gp -Stream dua.flv

Get-Content f:\ HANDS.3gp: dua.flv

While for the further nested ADS having depth level 2, 3 or so, there is no such command. For example, following command, with double colons for getting contents of the nested ADS, is not understandable by Windows Power Shell.

Get-Content f:\dua.flv:HANDS.3gp:Message.mp3

It is further observed that the compressed file having ADS, whether nested or not, doesn't show up through the "dir /r" command, nor through WinHex or commercial software. This compressed file is actually a file having no ADS of its own but the ADS attached to it once it was uncompressed.

Table 3.1: ADS Nesting Analysis

Instance	**Result**	**Finding**
ADS Nesting in uncompressed file **Message.mp3: (HANDS.3gp:dua.flv)**	Nested ADS (ADS within ADS) not retained	ADS Nesting not possible in uncompressed file (having ADS)
ADS Nesting in compressed file **Message.mp3: HANDS.rar** HANDS.rar has (HANDS.3gp:dua.flv)	Nested ADS (ADS within ADS) retained successfully	ADS Nesting possible only when the file (having ADS) is compressed.

3.1.2 Fragmentation of Binary File

Fragmentation is the process of dividing the information into various parts such that unless all the parts of a certain secret data are merged together, the information is not intelligible. There could be many ways to fragment a file and still further to hide them in multiple streams.

The research successfully employs the concept of fragmenting a binary file and putting them in multiple alternate data streams of a primary file. A power shell script is written for this purpose that takes a binary (audio/video/image/compressed) file as input, converts the file into Hex format, fragments the encoded file into four parts and then added these fragments as four different ADS to a primary file. The purpose behind the fragmentation is to lessen the probability of detection of the secret data concealed within ADS. The reverse path of combining the fragments of the binary file is also tested to be successful.

Windows Power Shell script is written to fragment a file after encoding it. "**Substring**" function associated with a variable is used to fragment a binary file. Each fragment is then saved in a separate variable and then "**Set-Content**" function is used to attach these fragments as ADS.

3.1.2.1 Procedure

Following steps are used to fragment a binary file to be attached to multiple alternate data streams of a primary file:

1. Binary file, to be fragmented, is selected.
2. Binary file's content is obtained in 'Byte Encoding'.
3. The Byte format is then converted into Hex format and stored in a file.
4. The Hex formatted file is then divided into four equally sized chunks or fragments.
5. A primary file is selected, to which the fragments of the binary file will be attached.
6. The fragments are then attached as four different alternate streams of the selected primary file.

To retrieve the fragmented binary file back into the merged form, the steps shown below are used:

1. The contents of the multiple ADS are got and merged in a single file.
2. Then this Hex formatted file is converted back into byte format.
3. Then the file contents are output in a file in 'Byte Encoding'.

3.1.2.2 Flowchart

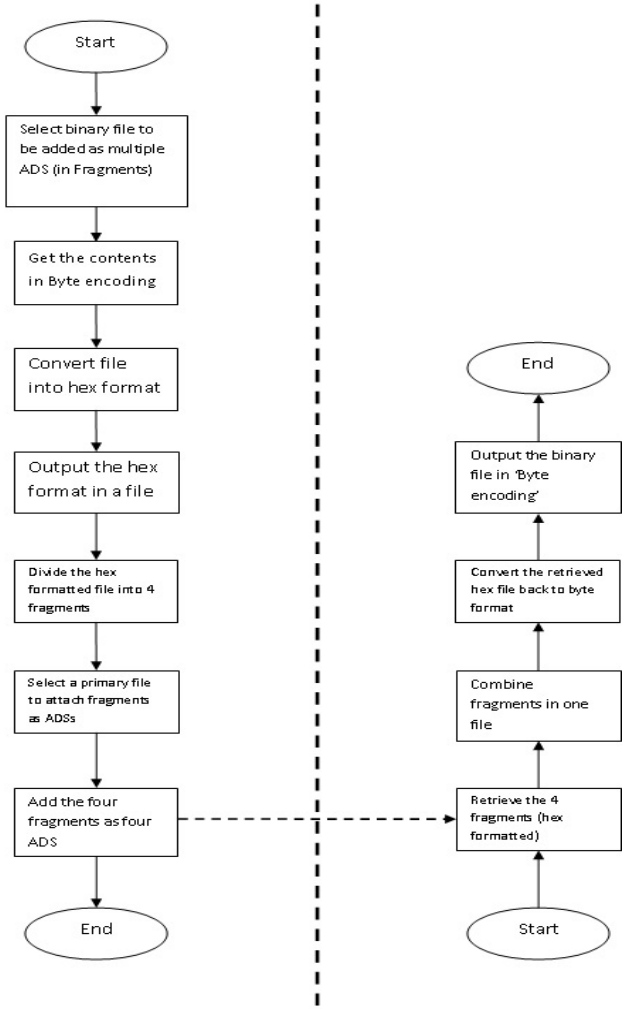


Fig. 3.2 Flow chart for Fragmentation

3.1.2.3 Analysis of the Procedure

A file is successfully fragmented into chunks and added as multiple ADS of a file but the below mentioned limitations and observations have been experienced while going through the process:

- The contents of the binary file (audio, video, compressed, image) can only got through the “Byte Encoding”. No other value of encoding parameter gets the contents of a binary file in a way that they can be manipulated or used later for examination.

- The binary file contents have to be further encoded, for example in a Hex format, to be used for experiments and evaluations. The reason is that once the binary file' content is got in a Byte Encoding and output in a file, the bytes then couldn't be retrieved from this saved file for further experimental use.
- Furthermore, if the binary file's contents in Byte encoding are manipulated on the run-time (i.e., without saving the contents in a file), the fragments are created but the integrity of both the fragmented and merged file is lost.
- Even with the encoding, when one of the Hex encoded fragments is retrieved individually, the fragments are retrieved and could be opened via an application but the integrity of the retrieved fragment is not maintained.
- The first Hex encoded fragment could not be opened or viewed through an application unless all the fragments are merged together.
- Unless all the fragments of the file are merged together, only then the binary file can be retrieved successfully with the integrity fully intact.

Hence, it is obvious from the observations stated above that the fragmentation of a file adds to the data hiding potential associated with alternate data streams. An adversary, if get hands on one or some of the fragments, could not be attained full information if succeed in getting any. Plus uniting the encoded fragments with some other data hiding techniques would further multifold the covertness to the ADSs.

Table 3.2: Fragmentation Analysis

Instance	**Result**	**Finding**
Getting Binary file contents using Byte value for Encoding parameter	Contents successfully retrieved and manipulated further	Only Byte Encoding successfully retrieves the binary file's contents
Saving the bytes in a file after getting binary file contents using Byte Encoding	Saved bytes in a file cannot be retrieved and used further	Contents get through Byte Encoding can only be used at runtime, bytes cannot be output in a file for future use

Hex Encoding of the bytes retrieved from a binary file	File successfully fragmented and fragments successfully retrieved with full integrity intact.	Bytes retrieved from a binary file can be used and manipulated successfully for further experiments when encoded in Hex Format.
Retrieving the Hex encoded individual fragments	Individual fragment is retrieved but their integrity is not maintained	Integrity of the retrieved fragment is lost
Retrieving the first Hex encoded fragment	First fragment of the binary file is not retrieved/ opened	First Hex-encoded fragment of the binary file could not be viewed unless all the fragments are merged
Merging all fragments of the binary file and then retrieving the merged file	Fragments of the binary file are merged and the merged file is successfully retrieved	The binary file can be retrieved and viewed successfully with the integrity retained only in case when all the fragments are merged together

3.1.3 ADS with Steganography

The study also employed the concept of using steganography along with alternate data streams. Secret data in a file is hidden behind an innocent file by applying steganography. For this purpose, steganography software is used. Then the innocent file with secret hidden file is further concealed as alternate stream to a primary file. The data is also effectively recovered back with full integrity retained.

3.1.3.1 Procedure

To create the test data to evaluate the ADS data hiding in conjunction with Steganography, below mentioned steps are followed:

1. Steganography software is opened.
2. Target file (to attach another file with secret data) is selected.
3. The file with secret data is selected, to be added to the target file selected above.
4. The file is then encrypted and password is given.
5. The file is then joined to the target file.
6. The encoded file (with hidden file) is then saved.
7. This file, with steganographic concealed file, is then attached as alternate stream of a primary file.

The data hidden by applying steganographic method besides ADS is retrieved back by following the below stated steps:

1. The encoded file attached as ADS is retrieved back as the main stream.
2. Steganographic application is launched.
3. The file, retrieved in the previous step, is then loaded in the steganography software.
4. The file/s hidden behind the encoded file is then selected.
5. The concealed file is then extracted and saved.

3.1.3.2 Analysis of the Procedure

By following the above stated procedure, the study claims the success of using ADS hiding process in union with the steganography.

Some of the observations noted during the course of action taken, are stated below:

- A secret file can be extracted out of the target file successfully if and only the password is given while retrieving a file. Otherwise, the file is extracted but one cannot have a view at the file's contents.

Table 3.3: ADS with Steganography

Instance	**Result**	**Finding**
Password is given while retrieving the file	File with secret data is extracted successfully	A secret file can be extracted out of the target file

	and secret data is visible	successfully if and only the password is given while retrieving a file.
Password is not given while retrieving the file	File with secret data is extracted but secret data is not visible	A secret file cannot be extracted out of the target file successfully if the password is not given while retrieving a file.

3.1.3.3 Flowchart

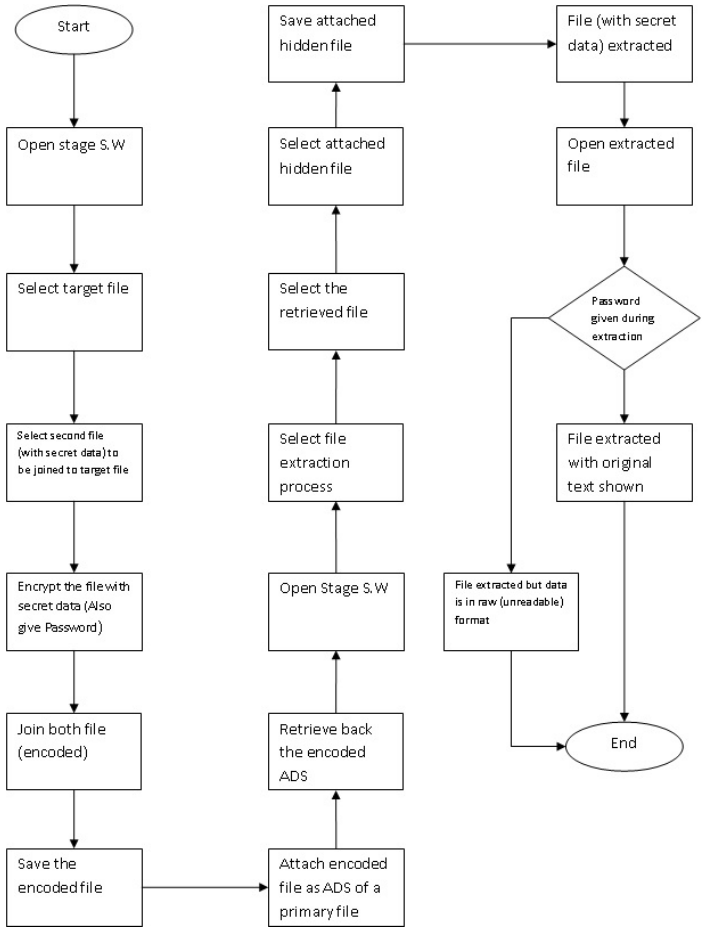


Fig. 3.3 Flow chart for ADS with Steganography

3.1.4 Exploring ADS Potential for Data Hiding

ADS likely potential for hiding digital data is explored in this section. Various security techniques in combination with each other are implemented in the dissertation and are proven successful after passing through the same power shell script. The ADSs are successfully preserved and retrieved while keeping the full confidentiality and integrity intact. Different digital data hiding methods are applied in conjunction of each other, for example, compression, encryption, ADS nesting, encoding and fragmentation. The variations of the combos can be unlimited.

3.1.4.1 Procedure

To create the test data for applying various techniques in combination, below mentioned steps are followed:

1. A binary File (video file) is selected.
2. Another binary file (video file), with secret data, is selected.
3. Secondary binary file is added as ADS to the primary binary file.
4. Primary file with ADS is then compressed through WinRAR, with streams options checked.
5. Another Primary binary file (audio file) is selected to hide the compressed file (containing ADS).
6. The compressed file (containing ADS) is then attached as ADS of the second primary file (audio file) selected.
7. Second primary (audio) file with ADS is then compressed through WinRAR, with streams options checked.
8. This compressed binary file with nested ADS is then fed into the power shell script.
9. The compressed binary file's content is obtained in 'Byte Encoding'.
10. The Byte format is then converted into Hex format and stored in a file.
11. The Hex formatted file is then divided into four equally sized chunks or fragments.
12. A primary (image) file is selected, to which the fragments of the binary file will be attached.

13. The fragments are then attached as four different alternate streams of the selected primary file.

To retrieve the compressed fragmented binary file back into the merged form, the following procedure is used:

1. The contents of the multiple ADS are got and merged in a single file.
2. Then this Hex formatted file is converted back into byte format.
3. Then the file contents are output in a file in ‘Byte Encoding’. The compressed file with nested ADS is acquired back in this step.
4. Next step is to uncompress the compressed file with nested ADS.
5. Decrypt the file (give password) while uncompressing the file.
6. Then by traversing to the uncompressed directory, check the uncompressed binary file (nested ADS of level 2) through WinHex, the “dir /r” command or through “ADS Manager”.
7. ADSs of depth level 2, concealed within the compressed file, are retained successfully.
8. The ADSs are then retrieved back as the primary files to confirm that the integrity is intact. For this purpose, the first compressed ADS is uncompressed to begin with.
9. If required, enter the password to uncompress the ADS.
10. Then by traversing to the uncompressed directory, check the uncompressed binary file (having ADS of level 1) through WinHex, the “dir /r” command or through “ADS Manager”.
11. ADSs of level 1 are retained successfully.

3.1.4.2 Flowchart

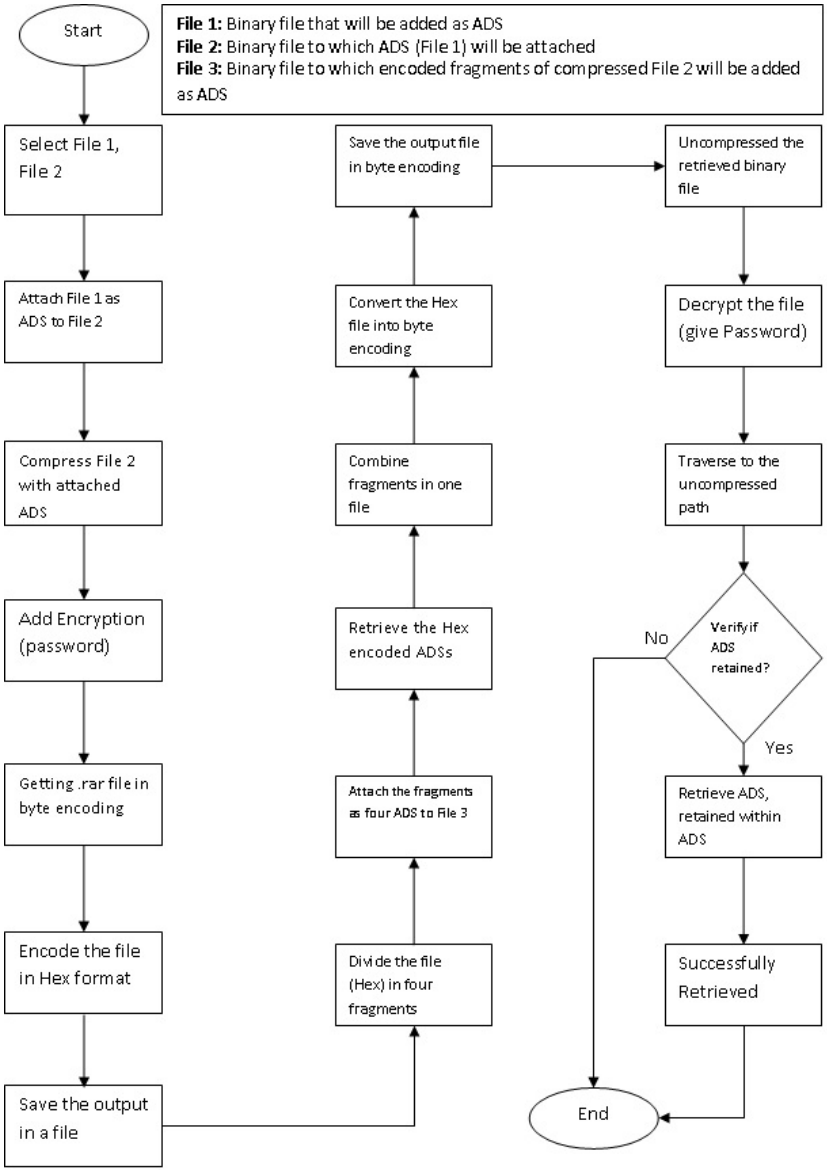


Fig. 3.4 Flow chart for Defense in Depth Technique

3.1.4.3 Analysis of the Procedure

The data have been successfully hidden inside the alternate data streams by using the approach of employing various data hiding techniques together. The files have also been effectively retrieved back and integrity is verified to be maintained. Following findings have been observed during the execution of the above mentioned steps:

- Time, required for the data hiding through the stated technique, increases with the increase in file size.
- The value assigned to the “ReadCount” parameter of “Get-Content” command greatly affects the execution time of algorithm, implemented for encoding and fragmenting of the primary files and ADSs.

ReadCount parameter has significant affect on the time the Get-Content command takes for fetching or retrieving the contents of a file. The total execution/ operation time reduces as the value of ReadCount increases [32], [33]. This can create a noticeable difference while dealing with very large sized files. Hence, one has to select the value of ReadCount intelligently by taking the file size into consideration.

Get-Content f:\ H.jpg -Encoding Byte -ReadCount 512 | Set-Content f:\ R_H.jpg -Encoding Byte

The ReadCount parameter is highlighted in Red along with its value to give a fair understanding of how this parameter is used with Get-Content command. The value here ‘512’ means that the command will get 512 bytes at a time while retrieving the contents of “H.jpg” file.

- The first chunk of data is retrieved slowly with the high value of ReadCount whereas retrieved quickly with low ReadCount value but overall retrieval time reduces when high value of the parameter is selected [32], [33].
- At maximum, about 125MB data has been retrieved successfully at once. This observation has two implications:
    - One can use zero value for ReadCount parameter only while dealing with the files less than or equal to 125MB, because it will give error of exceeding supported range with the larger files. Zero value of the parameter means to get all the contents at once.
    - One can give the value of at most ‘131,072,000’ (i.e., 125MB) to the ReadCount parameter as the value greater than this will give error.

Table 3.4: Defense in Depth Technique

Instance	Result	Finding
Getting Binary file contents of size less than or equal to 125 MB using zero value for ReadCount parameter	Contents successfully retrieved	About 125MB data can be retrieved successfully at once
Getting Binary file contents of size greater than 125 MB using zero value for ReadCount parameter	Contents do not retrieved and gives error of 'exceeding supported range'	At maximum, about 125MB data has been retrieved successfully at once

3.1.4.4 Exploring Diverse Grouping of Data Hiding Techniques

This section is intended to discover the probable options for data hiding when multiple data hiding techniques are employed in union. The data hiding methods used in the study are

- Compression
- Encryption (Password Protection)
- ADS Nesting
- ADS with Steganography
- Encoding
- Fragmentation

There are numerous possible combinations of all the above stated techniques. One arrangement of using various methods in union is described in this section. Whereas these approaches can be grouped in many different ways, for example, a binary file may first be encoded and fragmented at first and then attached as ADSs of a primary file. Then these encoded and fragmented ADSs are compressed and encrypted.

Another combination can be to attach a binary file as ADS, compress this file and attach another secret binary file with this compressed file through steganography software. Then attach this file (with steganographic hidden file) as ADS of another secret binary file. Then

compress this file and encrypt it with password. Then encode and fragment this compressed encrypted file and attach as multiple ADSs of s primary file.

Furthermore, one probable grouping could be to fragment an encoded secret binary file. Then add one of the fragments as steganographic file to another file. Then add this file (with steganographic hidden file fragment) as ADS of the second fragment of the first binary file. Then this fragment with ADS is compressed and password encrypted. The third and fourth fragments of the first binary files are then added as ADSs of this compressed encrypted file.

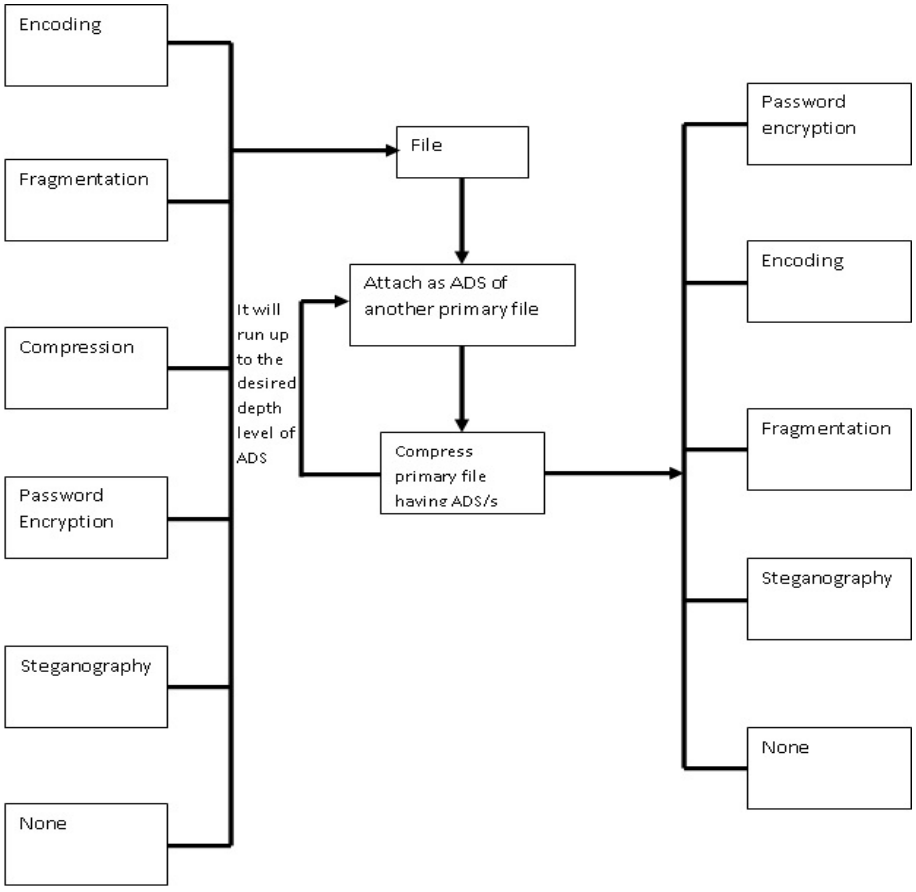


Fig. 3.5 Exploring Options for Employing Data Hiding Methods

Moreover, yet another likely arrangement could be to add the fragments of a binary file, each in a next depth level of ADS. To elaborate, it is to have four fragments nested up to depth level 4, having first fragment attached as ADS of a primary file. Then compressed

this file with ADS and attach this compressed file as ADS of the next fragment. Again compress this file and attach as ADS of the third fragment. The process is repeated once more to add the compressed file (having ADS of depth level 2) as alternate stream of the fourth fragment of the original binary file. Password encryption could also be added along with compression.

These techniques could be arranged in many different ways. The figure below will help understand the possibilities one has at hand while dealing with all these methods in conjunction. It demonstrate the prospect of utilizing the various data hiding approaches before adding the alternate data stream to a primary stream as well as employing these techniques after ADSs are attached with the primary stream. The probability of using data hiding methods with both the alternate data stream/s and primary streams (having ADS) is hence depicted through the figure below.

3.2 Reverse Engineering the ADSs

ADS Potential for concealing the secrets of extreme interests is explored in a variety of dimensions in the dissertation that will eventually facilitate the intelligence agencies and forensic investigators to get hands at the secrets of the adversaries. So, the research devises a path to unravel the potential camouflaged data behind ADS, keeping in view its usage as a covert medium.

Focal objective of the study is to unveil all the hidden alternate streams with potential covert data and to reveal the secrets concealed within, with as much success as possible while keeping the forensic challenges in consideration. The victory not only lies in retrieving or unveiling the stealthy data within ADS but also to preserve the evidence (ADS here) in its original form while digging them out of the multiple hiding techniques into the limelight for examination and investigation. If the alternate data streams got damaged or tampered during the retrieval process, the concealed data could not be termed as forensically sound and hence cannot be fully relied upon.

Given that the available literature on the NTFS alternate data streams could be read by anyone and tried at to exploit the various dimensions associated with ADS including data hiding, the probability of applying the derived forms of the existing methods is all the same. Adversaries and terrorists could also achieve such innovations and derivations that would thwart the well known discovery or revelation techniques both through tools and commands. Keeping all this into perspective, a detailed analysis of the seized systems of terrorists and/ or adversaries is crucial to unveil the likely concealed secrets.

3.2.1 Procedure Applied

A pathway is devised in the study that will take the following steps when alternate streams are found in an adversary's system to reverse engineer those streams by taking the various combinations of data hiding schemes into consideration. The following course of action is suggested when one has no prior knowledge of the techniques applied to the ADSs or the procedure via which the retrieved ADSs have gone through.

1. The system is probed for the presence of alternate data streams.
2. All ADSs are retrieved back to the system as primary files.
3. ADSs are analyzed for file types.
4. If file types are recognized, the files are opened via appropriate application/ software.
5. If the retrieved ADSs are encoded, they are scrutinized for the encoding scheme applied.
6. The decoding process is applied once the employed encoding technique is comprehended.
7. If more than one encoded ADSs are found of equal size and employing same encoding format, the likely fragments are combined into one file.
8. The combined fragments are then decoded back and the decoded file is output in byte encoding.
9. File type is then checked through WinHex.
10. The file is then opened through the correct application.
11. If the file extension is ".rar" (compressed file), then the file is uncompressed.

12. If password is asked for while uncompressing the file, various methods are used to recover the password or to get the hidden data. Otherwise skip this step.

13. Then by traversing to the uncompressed directory, the directory is further checked for the existence of alternate data streams in this directory.

14. If the nested alternate data streams are found in the directory, then the process will be repeated from step 2.

15. The process will be repeated until all the compressed ADSs are retrieved and investigated for the ADSs concealed within the file when uncompressed.

3.2.2 Analysis of the Procedure

Reverse engineering the alternate data streams have been explored in the research with all the employed hiding techniques. This implied that all the ADSs hidden by applying various data hiding methods have been successfully retrieved back while preserving the evidence and catering all the forensic challenges. Having said that, some of the observations have been noted as follows:

- Various techniques for adding covertness to the ADS have been implemented in the thesis, only to give the intended readers a fair understanding of the data hiding potential that lies within the NTFS' alternate data streams. It will eventually help the law enforcement agencies, military intelligence and forensic investigators to know the various possibilities and where to look for the concealed secrets.
- To reverse engineer the ADSs is not an easy task while being totally ignorant of the algorithm or procedure applied for hiding the data within.
- It is observed that when the file having ADS is compressed, the alternate streams associated with the compressed file is not shown through WinHex, commercial ADS software/s and nor through 'dir /r' command. This observation leads to the notion that besides examining the alternate streams found on the system, all the compressed files also required to be thoroughly checked and probed.

3.2.3 Flowchart

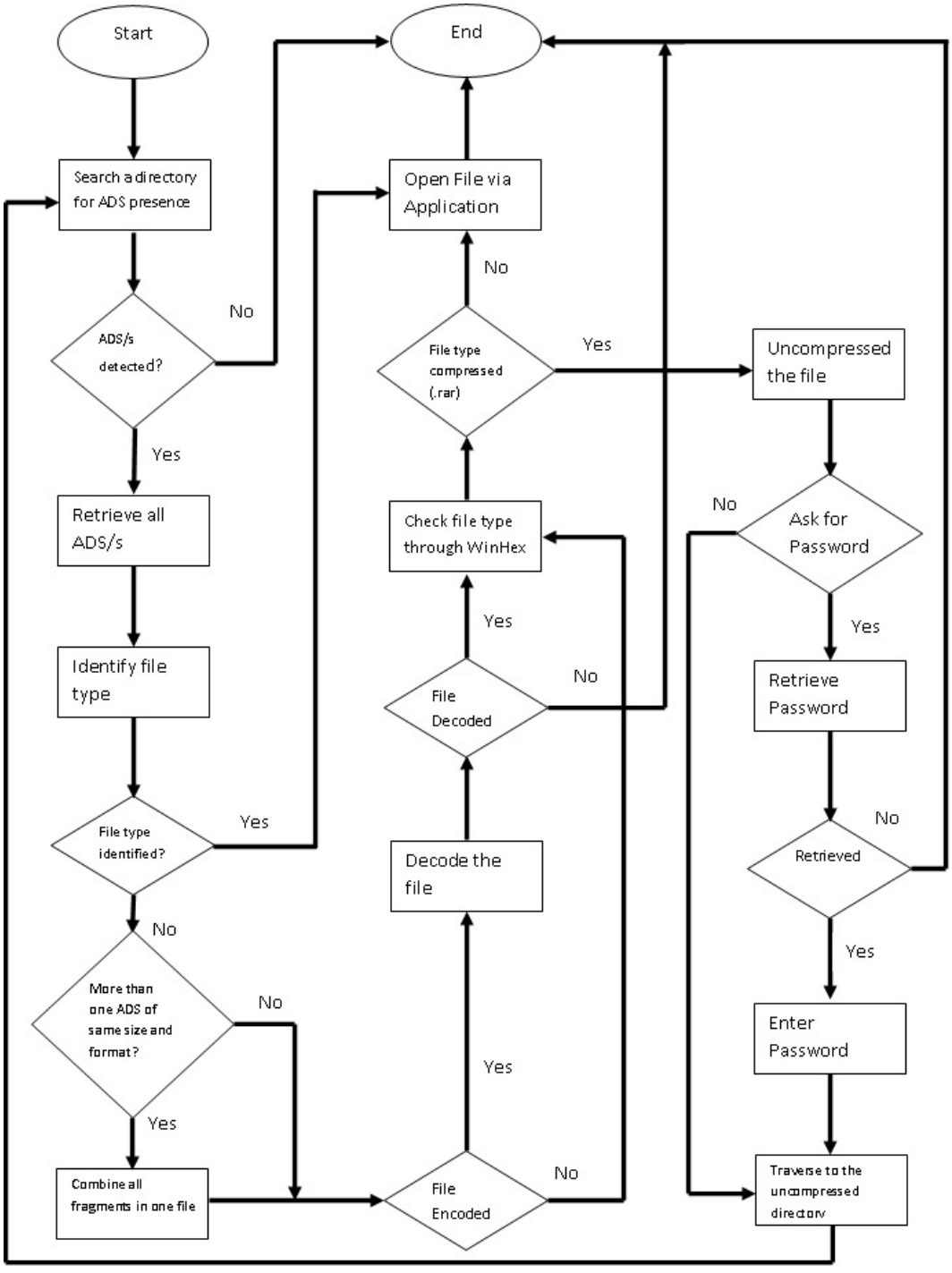


Fig. 3.6 Flow chart for Reverse Engineering ADS

3.2.4 Devising a Pathway to Reverse Engineer ADS

Below is displayed a generalized route worked out in the research to assist the forensic examiners and intelligence agencies while trying to retrieve and recover the alternate data streams.

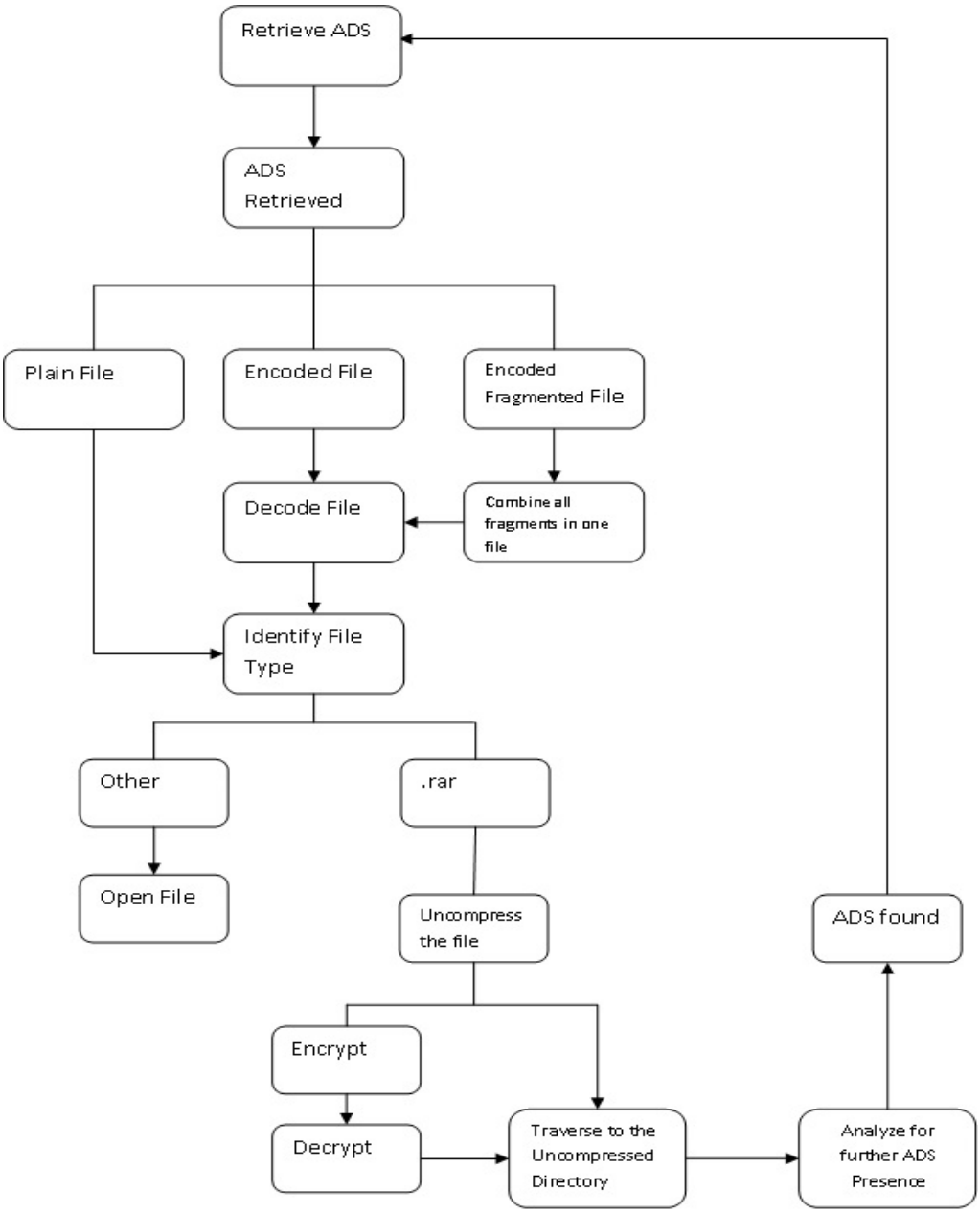


Fig. 3.7 Pathway to Reverse Engineer ADS

3.3 Forensic Challenges

The hidden nature of the NTFS alternate data streams is itself a challenge. Moreover the diverse data hiding approaches, implemented and examined in the dissertation, make the task of detecting and retrieving the ADSs competently even more complicated.

As the study is focused towards the discovery of the concealed secrets (of chief concerns) rather than towards the malicious use of ADS, therefore the forensic challenges are enlisted by taking this perspective of alternate streams into account.

- **Compressed ADS:** The compressed alternate streams present the biggest challenge for forensic investigators while reverse engineering the ADSs to detect the covert data inside. The reasons for this are
    - ADS' nesting is possible only with compressed files.
    - Compressed ADS could be further password encrypted.
    - Compressed ADS could be further encoded and/or fragmented.
- **Encryption:** The encryption scheme applied and/or the password protection given to the alternate streams hinder the examination procedure in regular whereas lead to the failure of data retrieval in extreme cases.
- **Steganography Technique:** The applied steganographic scheme also exposes a challenge during data extraction from the ADSs.
- **Encoding Scheme:** The encoding format used for hiding data in ADSs is also matter of concern for forensic examination.
- **Fragmentation*:*** Fragmentation also renders a great challenge while carrying out the task to reverse engineer an alternate data stream.
    - Number of Fragments of a file
    - The manner in which the fragments are concealed within multiple ADSs
- **Unknown Algorithm:** The approach or algorithm employed in hiding the digital data within ADSs when multiple data hiding techniques are used in union, also pose a challenge. Without exactly knowing the steps of the process used for hiding the data inside ADS, successful recovery of data seems impossible.

- **Backup Software:** The possibility that the systems might not be completely backed up is a main concern for forensic work [16]. There are two associated risks with this
    - The alternate data streams are not backed up properly
    - The backup data is restored on a non-NTFS volume that would result in the loss of the ADSs.

Chapter 4

4. Experimental Results and Observations

The experimental effects and findings are listed in this chapter, against various approaches and techniques that are examined throughout the study. The section is drafted in way that focused towards whether the data integrity is retained during the course of retrieving or digging out the ADS against their usage to conceal secret data. The results are measured against the specified parameters to ensure that the procedure meets the forensic standards by successfully preserving the evidence, ADS in the case.

The observations established during the thesis are also outlined in the section. These findings are noted down during the testing of the different approaches employed in the research. In the end, some recommendations are stated for more efficient usage and detection of ADS, based on the experiments and evaluations.

4.1 Test Environment

A Fujitsu Lifebook AH532/G21 laptop machine is employed for testing purpose with 64-bit Microsoft Windows 10 operating system installed. System's specs include Intel Core i7 processor with the speed of 2.20 GHz and 6GB installed RAM.

Following file types are examined in the dissertation against the varied data hiding approaches. These file types are analyzed before their addition as ADS and after digging them out of ADS to make sure that the data retrieved from alternate streams is forensically sound and is to be trusted at.

- Compressed (.rar)
- Acrobat Reader (.pdf)
- Audio (.mp3)
- Video (.flv, .3gp)
- Image (.jpg, .png)

4.2 Software Requirements

- **Windows PowerShell 5.0 (Administrator Mode):** A task-based command-line shell used in the study for manipulating with ADS, Retrieving ADS contents, retrieving ADS as a primary file, encoding and fragmentation of ADS.
- **Command Prompt (Administrator Mode):** Windows command shell used in the thesis for creating ADS and scanning a drive or directory through “dir /r” command.
- **WinHex** is a universal hex editor used in the field of digital forensics and data recovery. It is employed in the study to examine the NTFS files, drives and directories for the presence of alternate data streams.
- **WinRar** is a file archiver utility for Windows employed for compressing the files having ADS for ADS nesting and adding passwords to the file.
- **MD5 and SHA checksum utility** is a freeware tool that generates hash to verify the file integrity and is used for the same purpose in the research.
- **ADS Manager** is a free utility for adding, removing, searching, renaming and editing the alternate data streams associated with the main stream files [37].
- **Xiao Stenography** is free steganography software for hiding a text file behind an image or audio file. The program also offers encryption and password protection. Xiao steganography is used for employing ADS data hiding in combination with the steganography technique.

4.3 Results

The results are compiled category wise by taking one approach at a time. A comparison is shown of a binary file before it is attached as ADS and after its retrieval against the following parameters:

- File Hash
- File Size
- File Duration (in case of audio/video)

3.1.3 ADS Nesting

a. ADS of Depth level 2

The testing procedure involves the below mentioned files:

- dua.flv – file to be added as ADS
- HANDS.3gp – primary file to add ADS
- HANDS.rar – compressed file (with ADS) to be further added as ADS of another primary file
- Message.mp3 – second primary file to add compressed file (having ADS) as ADS

All the above files are measured against the parameters both at the start of the procedure and after their retrieval following their use as ADS.

Results of original file against the said parameters are displayed in **grey** while for retrieved file, they are shown in **white**.

Table 4.1: ADS Nesting (Depth Level 2 ADS) Test Results

	File Type/ Name with Extension	**File Hash (MD5)**	**File Size (KB)**	**File Duration (seconds)**
Original File	Video/dua.flv	81BA2B3648FAE571F98C00891E9D9E8B	1084	12
Retrieved File	Video/R_dua.flv	81BA2B3648FAE571F98C00891E9D9E8B	1084	12
Original File	Video/HANDS.3gp	4005CC3D79D1B8853D6560E831A33CC7	263	37
Retrieved File	Video/ R_HANDS.3gp	4005CC3D79D1B8853D6560E831A33CC7	263	37
Original File	Compressed/ HANDS.rar	9141B85E8ABCB1747FEC71A20116E5C8	1300	-
Retrieved File	Compressed/ R_HANDS.rar	9141B85E8ABCB1747FEC71A20116E5C8	1300	-

b. ADS of Depth level 3

The testing procedure involves the below mentioned files:

- dua.flv – file to be added as ADS
- HANDS.3gp – primary file to add ADS
- HANDS.rar – compressed file (with ADS) to be further added as ADS of another primary file

- Message.mp3 – second primary file to add compressed file (having ADS) as ADS
- Message.rar – the second primary file is again compressed with ADSs
- Home.jpg – third primary file to add compressed second primary file

All the above files are measured against the parameters both at the start of the procedure and after their retrieval following their use as ADS. Results of original file against the said parameters are displayed in **grey** whereas for retrieved file, they are shown in **white**.

Table 4.2: ADS Nesting (Depth Level 3 ADS) Test Results

	File Type/ Name with Extension	**File Hash (MD5)**	**File Size (KB)**	**File Duration (seconds)**
Original File	Video/dua.flv	81BA2B3648FAE571F98C00891E9D9E8B	1084	12
Retrieved File	Video/R_dua.flv	81BA2B3648FAE571F98C00891E9D9E8B	1084	12
Original File	Video/HANDS.3gp	4005CC3D79D1B8853D6560E831A33CC7	263	37
Retrieved File	Video/ R_HANDS.3gp	4005CC3D79D1B8853D6560E831A33CC7	263	37
Original File	Compressed/ HANDS.rar	9141B85E8ABCB1747FEC71A20116E5C8	1300	-
Retrieved File	Compressed/ R_HANDS.rar	9141B85E8ABCB1747FEC71A20116E5C8	1300	-
Original File	Audio/ Message.mp3	B97210DA6C3B4886ED718740987BFC69	1791	114
Retrieved File	Audio/ R_Message.mp3	B97210DA6C3B4886ED718740987BFC69	1791	114
Original File	Compressed/ Message.rar	469FE5C4698E1228804E64C7E2F96F75	4762	-
Retrieved File	Compressed/ R_Message.rar	469FE5C4698E1228804E64C7E2F96F75	4762	-

c. ADS of Depth level 4

The testing for further nesting, of depth level 4 ADS, the procedure involves the below mentioned files along with all the files mentioned in part b.

- Home.rar – the third primary file is again compressed with ADSs
- Flowers.png – fourth primary file to add compressed third primary file

Results of original file against the said parameters are displayed in **grey** whereas for retrieved file, they are shown in **white**.

Table 4.3: ADS Nesting (Depth Level 4 ADS) Test Results

	File Type/ Name with Extension	**File Hash (MD5)**	**File Size (KB)**	**File Duration (seconds)**
Original File	Video/dua.flv	81BA2B3648FAE571F98C00891E9D9E8B	1084	12
Retrieved File	Video/R_dua.flv	81BA2B3648FAE571F98C00891E9D9E8B	1084	12
Original File	Video/HANDS.3gp	4005CC3D79D1B8853D6560E831A33CC7	263	37
Retrieved File	Video/ R_HANDS.3gp	4005CC3D79D1B8853D6560E831A33CC7	263	37
Original File	Compressed/ HANDS.rar	9141B85E8ABCB1747FEC71A20116E5C8	1300	-
Retrieved File	Compressed/ R_HANDS.rar	9141B85E8ABCB1747FEC71A20116E5C8	1300	-
Original File	Audio/ Message.mp3	B97210DA6C3B4886ED718740987BFC69	1791	114
Retrieved File	Audio/ R_Message.mp3	B97210DA6C3B4886ED718740987BFC69	1791	114
Original File	Compressed/ Message.rar	469FE5C4698E1228804E64C7E2F96F75	4762	-
Retrieved File	Compressed/ R_Message.rar	469FE5C4698E1228804E64C7E2F96F75	4762	-
Original File	Image/ Home.jpg	B337243B4A4067FF43AA6D3765E350F0	144	-
Retrieved File	Image/ R_Home.jpg	B337243B4A4067FF43AA6D3765E350F0	144	-
Original File	Compressed/ Home.rar	B1377A88545E15F76407E949670BBB6A	4911	-
Retrieved File	Compressed/ R_Home.rar	B1377A88545E15F76407E949670BBB6A	4911	-

4.3.2 Fragmentation

Following set of files has been tested against the script used for encoding and fragmentation.

- Compressed (HANDS.rar, HANDS_CBig.rar)

- Video (HANDS.3gp, HTB_HANDSBig.3gp)
- Audio (Message.mp3, HTB_MBig.mp3)
- Pdf (J.pdf, HTB_JBig.pdf)
- Image (D.jpg, HTB_DBig.jpg)
- Image (F.png, HTB_FBig.png)

All these files are examined against the following parameters both at the start of the procedure and after their retrieval from ADSs following combining the fragments into one file and decoding that file. Results of original file against the said parameters are displayed in **grey** whereas for retrieved file, they are shown in **white**.

Table 4.4: Fragmentation (.rar) Test Results

	File Type/ Name with Extension	File Hash (MD5)	File Size (KB)	File Duration (seconds)
Original File	Compressed/ HANDS.rar	9141B85E8ABCB1747FEC71A20116E5C8	1300	-
Retrieved File	Compressed/ HANDS_CBig.rar	9141B85E8ABCB1747FEC71A20116E5C8	1300	-

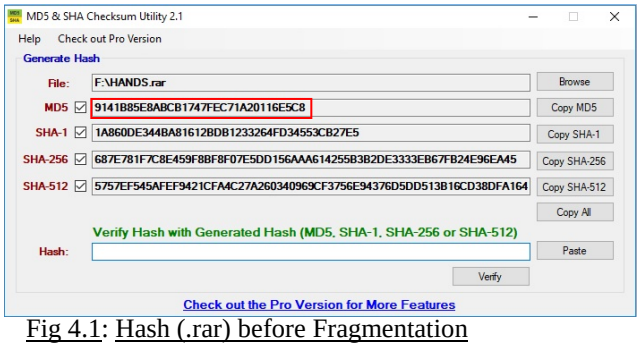


Fig 4.1: Hash (.rar) before Fragmentation

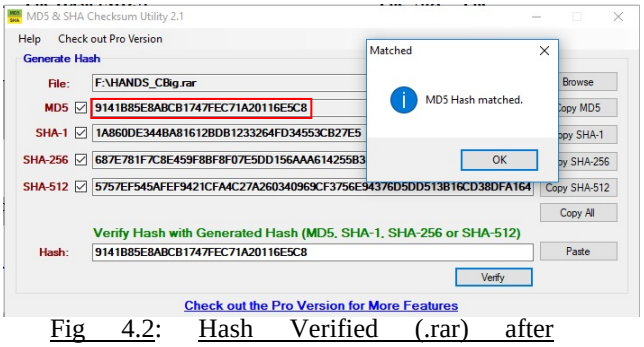


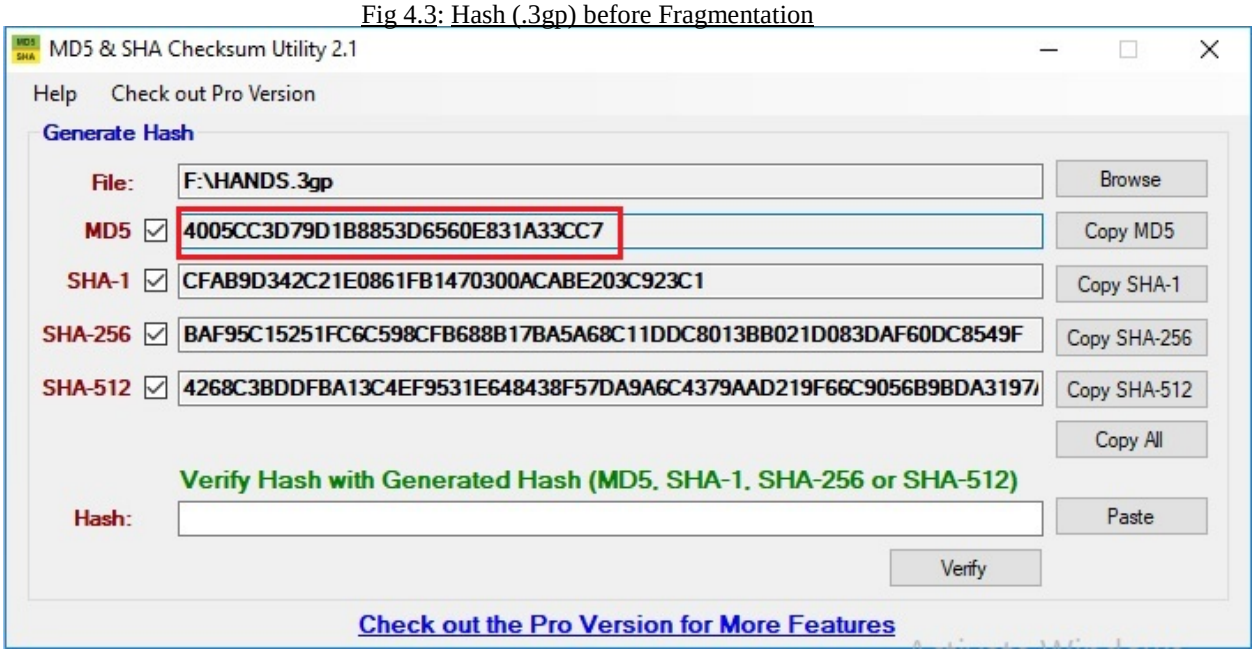
Fig 4.2: Hash Verified (.rar) after Fragmentation

Table 4.5: Fragmentation (.3gp) Test Results

	File Type/ Name with Extension	File Hash (MD5)	File Size (KB)	File Duration (seconds)

Original File	Video/HANDS.3gp	4005CC3D79D1B8853D6560E831A33CC7	263	37
Retrieved File	Video/ HTB_HANDSBig.3gp	4005CC3D79D1B8853D6560E831A33CC7	263	37

Fig 4.3: Hash (.3gp) before Fragmentation



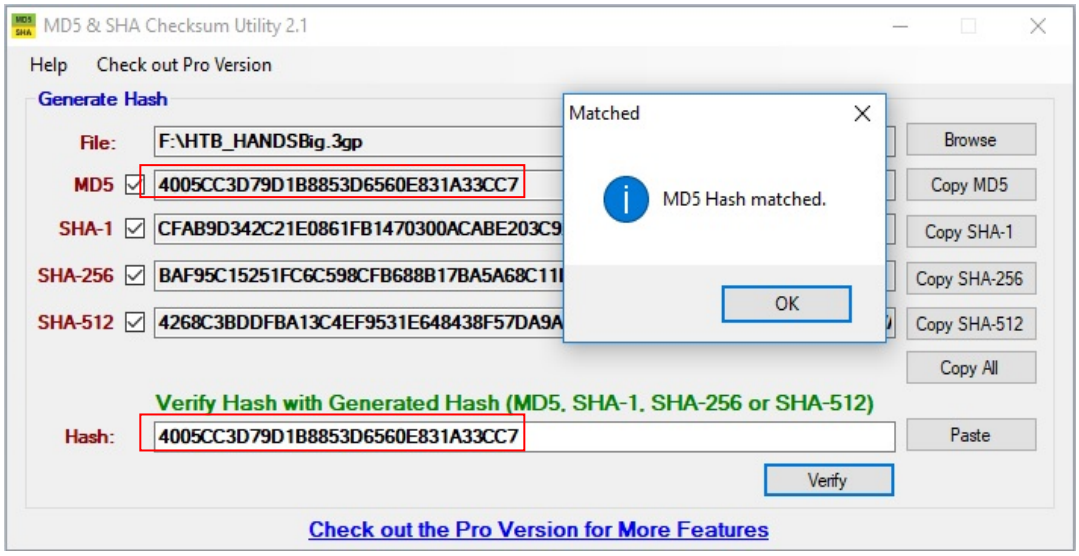


Fig 4.4: Hash Verified (.3gp) after Fragmentation

Table 4.6: Fragmentation (.mp3) Test Results

	File Type/ Name	**File Hash (MD5)**	**File Size**	**File Duration**

	with Extension		(KB)	(seconds)
Original File	Audio/ Message.mp3	B97210DA6C3B4886ED718740987BFC69	1791	114
Retrieved File	Audio/ HTB_MBig.mp3	B97210DA6C3B4886ED718740987BFC69	1791	114

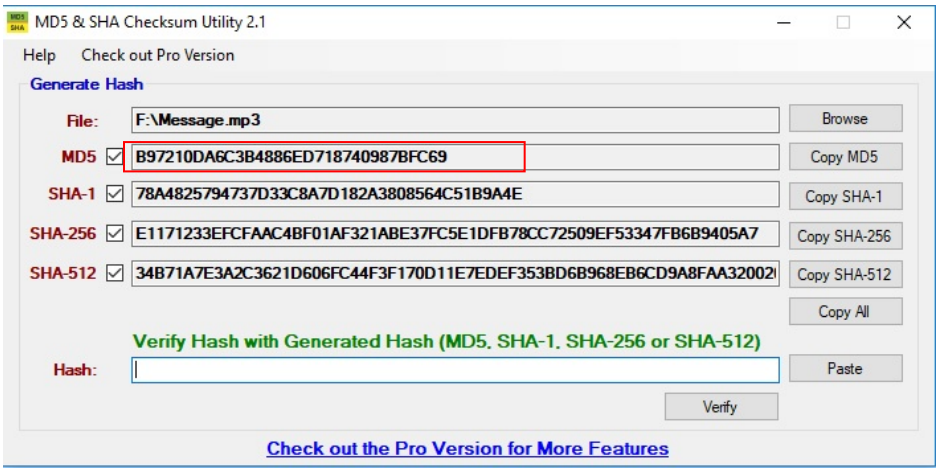


Fig 4.5: Hash (.mp3) before Fragmentation

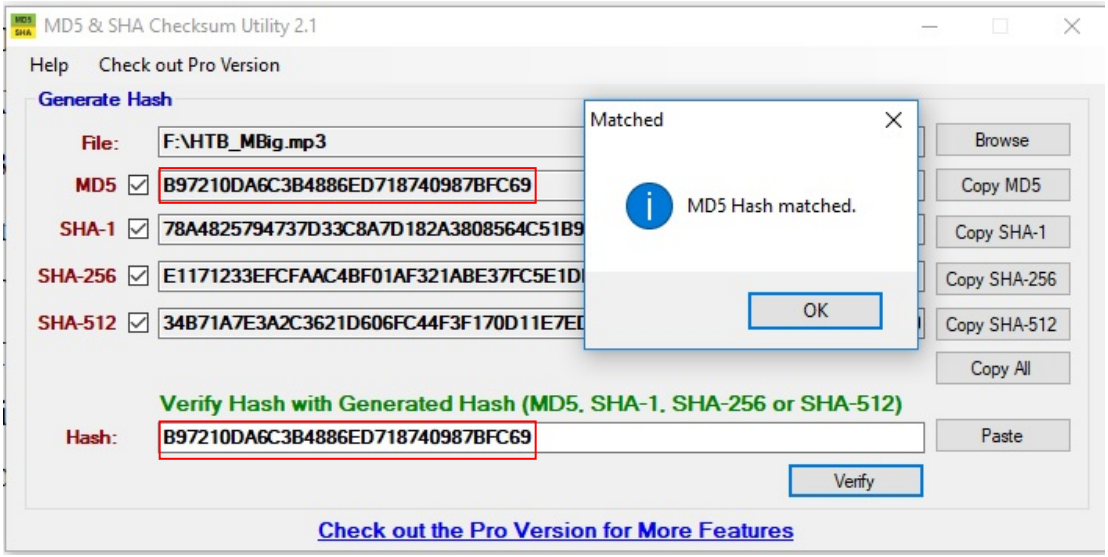


Fig 4.6: Hash Verified (.mp3) after Fragmentation

Table 4.7: Fragmentation (.pdf) Test Results

	File Type/ Name with Extension	File Hash (MD5)	File Size (KB)	File Duration (seconds)
Original File	Pdf/ J.pdf	5213043513D0CAA408871C2D0EBC1D23	1366	-

Retrieved File	Pdf/ HTB_JBig.pdf	5213043513D0CAA408871C2D0EBC1D23	1366	-

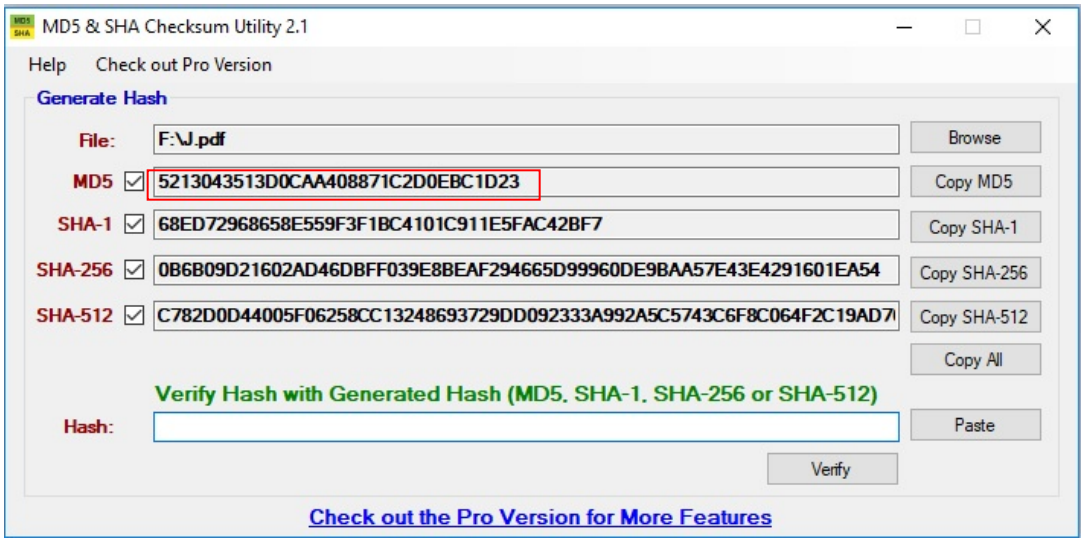


Fig 4.7: Hash (.pdf) before Fragmentation

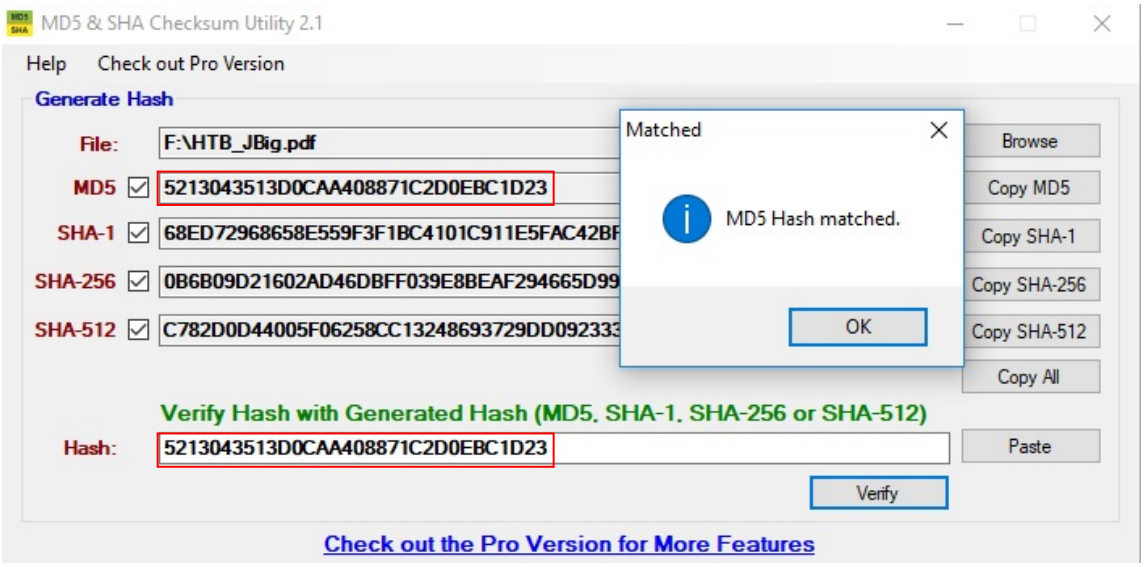


Fig 4.8: Hash Verified (.pdf) after Fragmentation

Table 4.8: Fragmentation (.jpg) Test Results

	File Type/ Name with Extension	**File Hash (MD5)**	**File Size (KB)**	**File Duration (seconds)**
Original File	Image/ D.jpg	AE308055BEBEEFAF3A7B8FA002533DBE	499	-
Retrieved File	Image/ HTB_DBig.jpg	AE308055BEBEEFAF3A7B8FA002533DBE	499	-

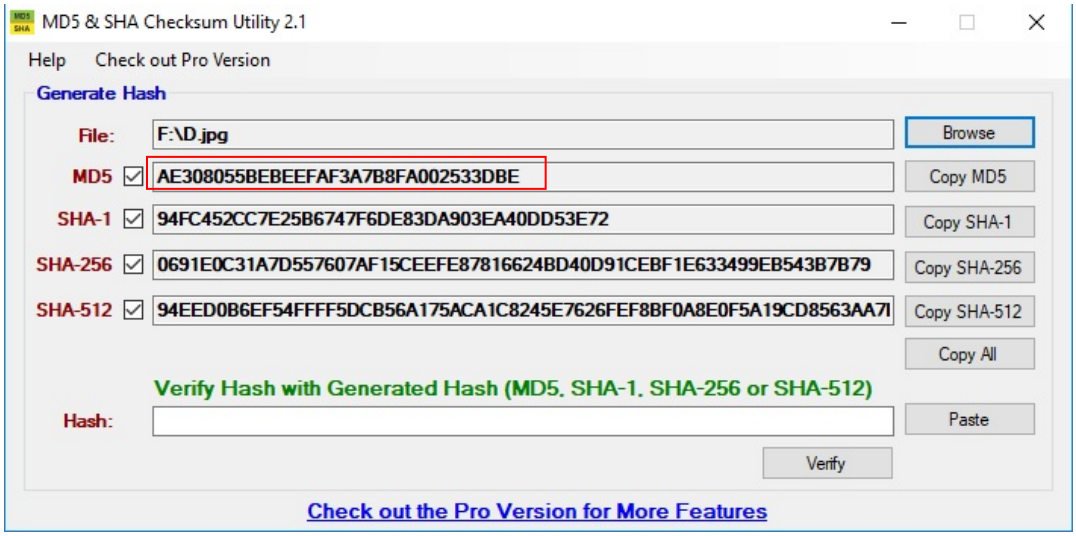


Fig 4.9: Hash (.jpg) before Fragmentation

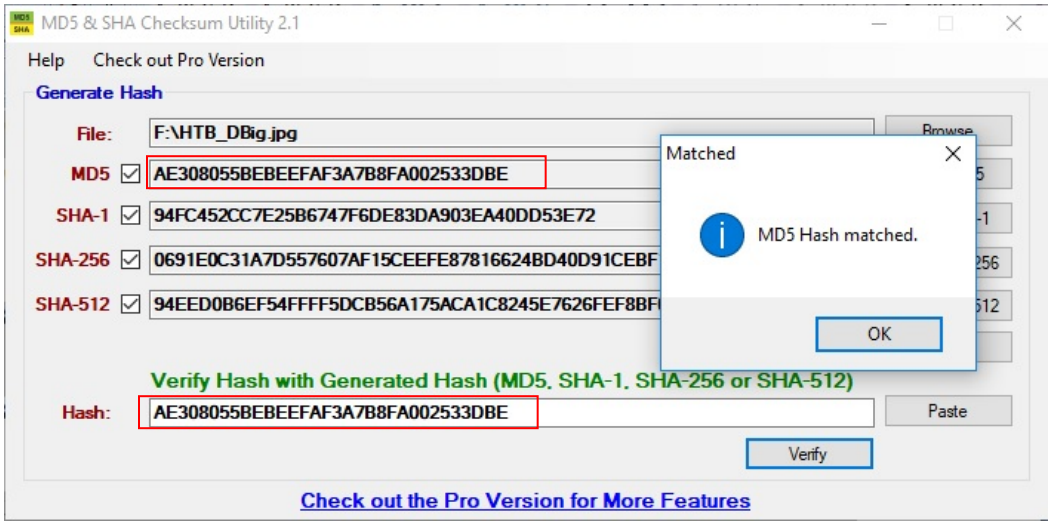


Fig 4.10: Hash Verified (.jpg) after Fragmentation

Table 4.9: Fragmentation (.png) Test Results

	File Type/ Name with Extension	**File Hash (MD5)**	**File Size (KB)**	**File Duration (seconds)**
Original File	Image/ F.png	8CEC53DAB120021CF425FB265D31DCB1	341	-
Retrieved File	Image/ HTB_FBig.png	12D581DA9AF0CFB5BD06A6919A03973D	341	-

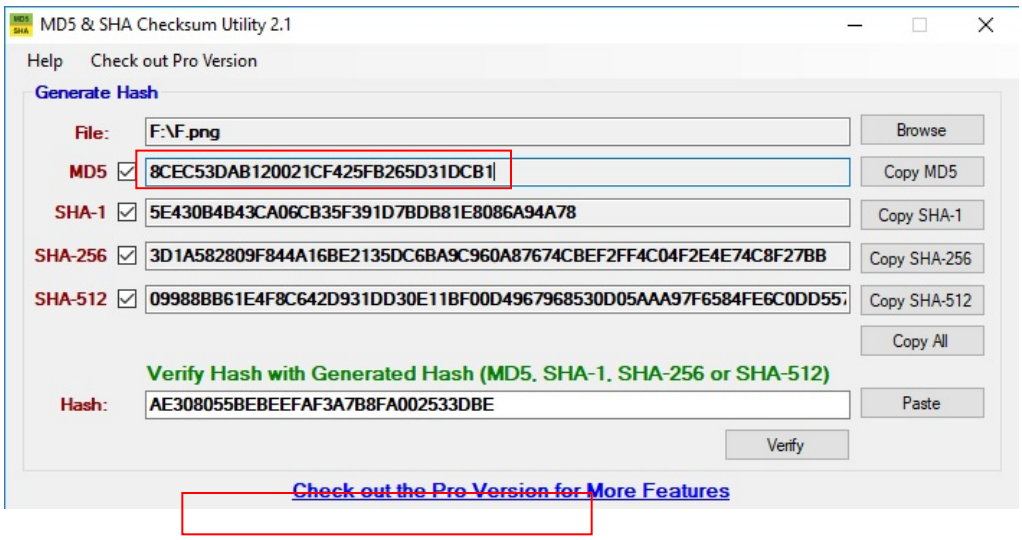


Fig 4.11: Hash (.png) before Fragmentation

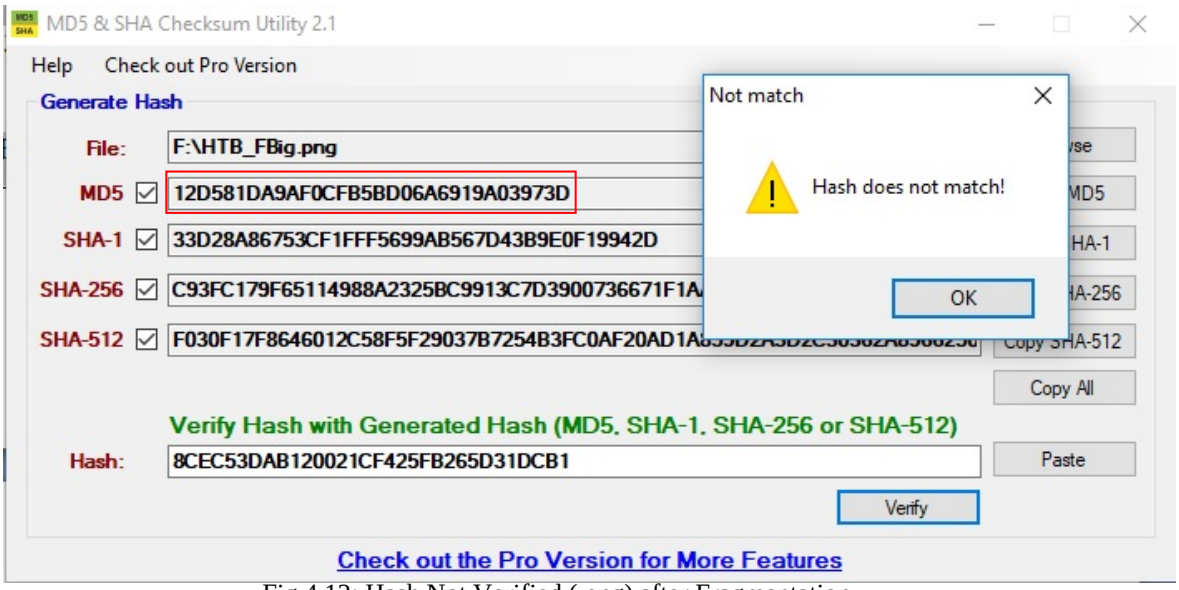


Fig 4.12: Hash Not Verified (.png) after Fragmentation

Fig 4.13: Original Image (.png) before Encoded Fragmentation Fig 4.14: Retrieved Image (.png) after Encoded Fragmentation

4.3.3 Combination of compression, encryption, ADS nesting, encoding and fragmentation

To test the combination of various techniques, a **test scenario** is created in which a video file (**dua.flv**) is added as ADS of another video file (**HANDS.3gp**). The HANDS.3gp is then **compressed** with streams options checked. The password **encryption** is also added while compressing the primary stream (HANDS.3gp). The compressed file (**HANDS.rar**) is then **nested** as alternate stream of another primary audio file (Message.mp3) to add the depth level to ADS. The nested ADS, of depth level 1, is then retrieved as R_H.rar. This retrieved ADS (**R_H.rar**) is then fed to the power shell script for **encoding** and **fragmentation (R_H)** and then to reverse script for combining fragments in one file (**H_Big**) and decoding. The resulted file is output as HTB_HBig.rar. The decoded file (**HTB_HBig.rar**) is then decompressed by decrypting the file (giving password). Then by traversing to the uncompressed directory, further presence of nested ADS is checked. And as in this case, there is the nested alternate stream of depth level 2 existed (dua.flv), this nested stream is then retrieved and verified whether the integrity is maintained or not. Results of original file against the said parameters are displayed in **grey** whereas for retrieved file, they are shown in **white**.

First of all, **dua.flv** is checked against the following parameters both at the start of the procedure before adding it (dua.flv) as ADS and then when it is retrieved back at the end of procedure by traversing to the uncompressed folder (**ADS Nesting**).

Table 4.10: Nested ADS Test Results

	File Type/ Name with Extension	**File Hash (MD5)**	**File Size (KB)**	**File Duration (seconds)**
Original File	Video/dua.flv	81BA2B3648FAE571F98C00891E9D9E8B	1084	12

Retrieved File	Video/R_dua.flv (of depth level 2)	81BA2B3648FAE571F98C00891E9D9E8B	1084	12
Original File	Compressed/ HANDS.rar	9141B85E8ABCB1747FEC71A20116E5C8	1300	-
Retrieved File	Compressed/ R_H.rar (of depth level 1)	9141B85E8ABCB1747FEC71A20116E5C8	1300	-

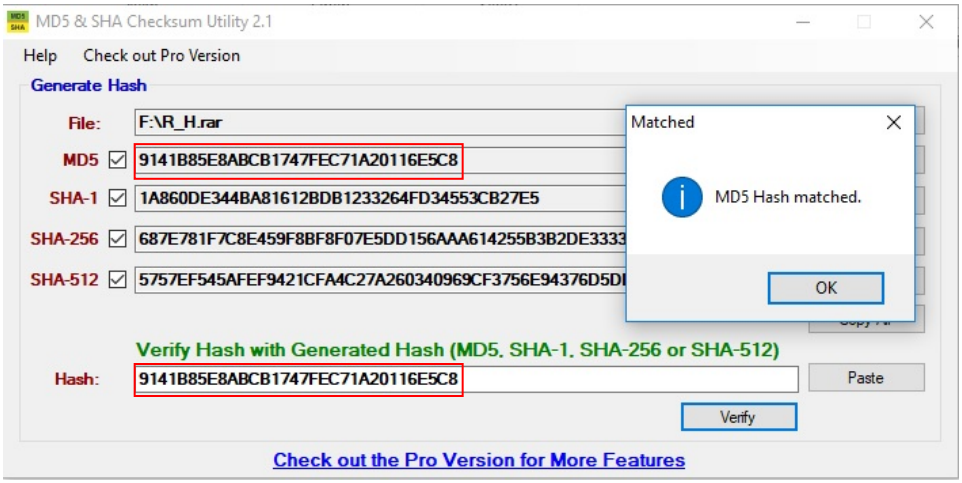


Fig 4.15: Nested ADS (Depth Level 1) Hash Verified

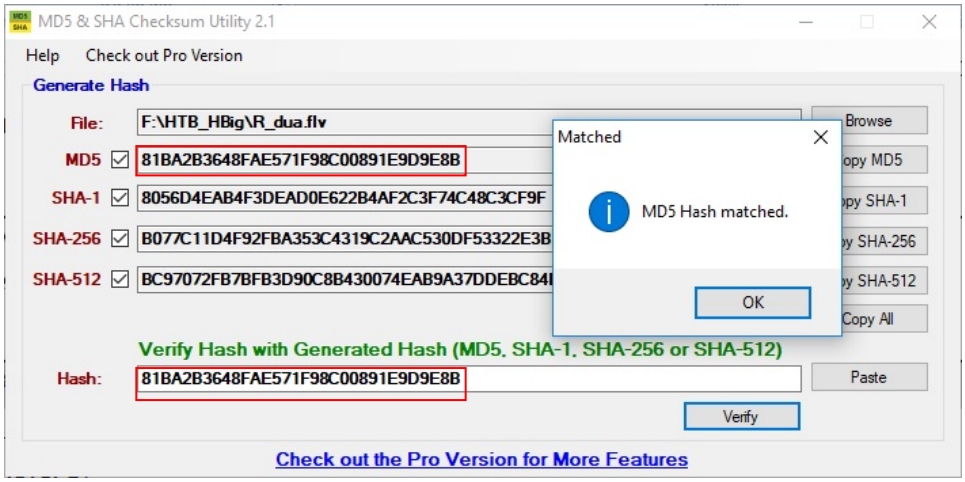


Fig 4.16: Nested ADS (Depth Level 2) Hash Verified

The compressed file '**R_H.rar'** is checked against the following parameters before encoding applied to it and also when the file is decoded back through the script, after fragmentation and defragmentation '**HTB_HBig.rar'.** Hence both these files (R_H.rar, HTB_HBig.rar) are measured in the following table to verify that the evidence is preserved throughout the employed encoding procedure (**Encoding**).

Table 4.11: Encoding Test Results

	File Type/ Name with Extension	File Hash (MD5)	File Size (KB)	File Duration (seconds)
Original File	Compressed/ R_H.rar	9141B85E8ABCB1747FEC71A20116E5C8	1300	-
Retrieved File	Compressed/ HTB_HBig.rar	9141B85E8ABCB1747FEC71A20116E5C8	1300	-

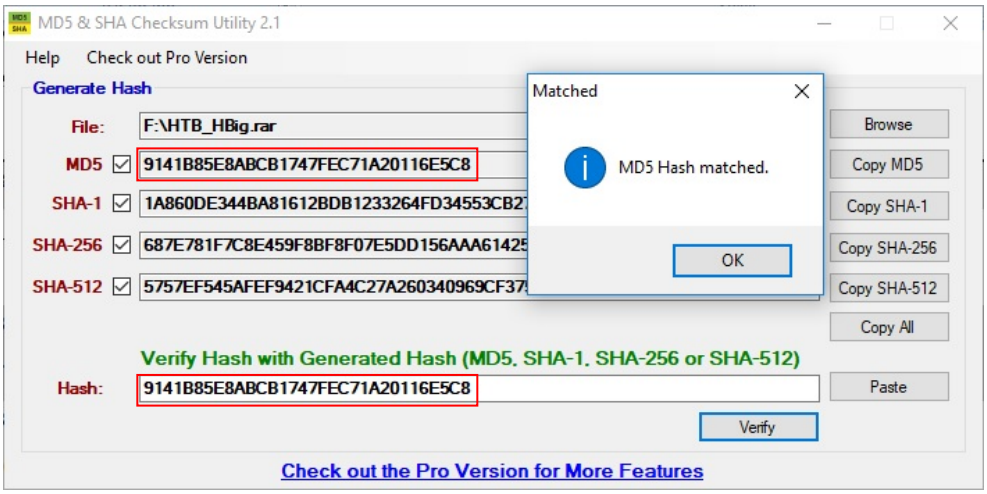


Fig 4.17: Encoding Hash Verified

Then encoded file '**R_H'** is checked against the following parameters before fragmenting it and then when the file is fragmented and gathered back in one file again '**H_Big'.** Hence both these files (R_H, H_Big) are measured in the following table to verify the forensic soundness of the fragmentation process (**Fragmentation**).

Table 4.12: Fragmentation Test Results

	File Type/ Name with Extension	File Hash (MD5)	File Size (KB)	File Duration (seconds)
Original File	File/ R_H	8748EAE4465A0D34437E31C9BFDAD513	2600	-
Retrieved File	File/ H_Big	8748EAE4465A0D34437E31C9BFDAD513	2600	-

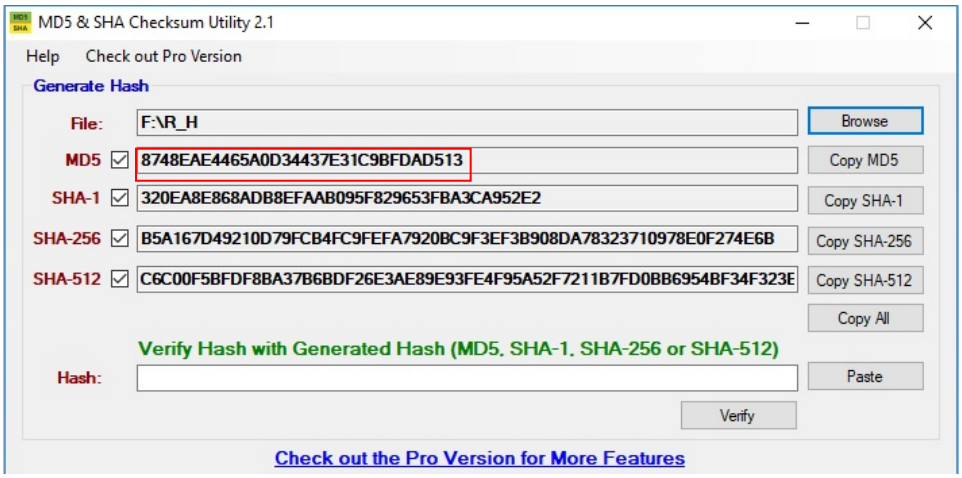


Fig 4.18: File Hash before Fragmentation

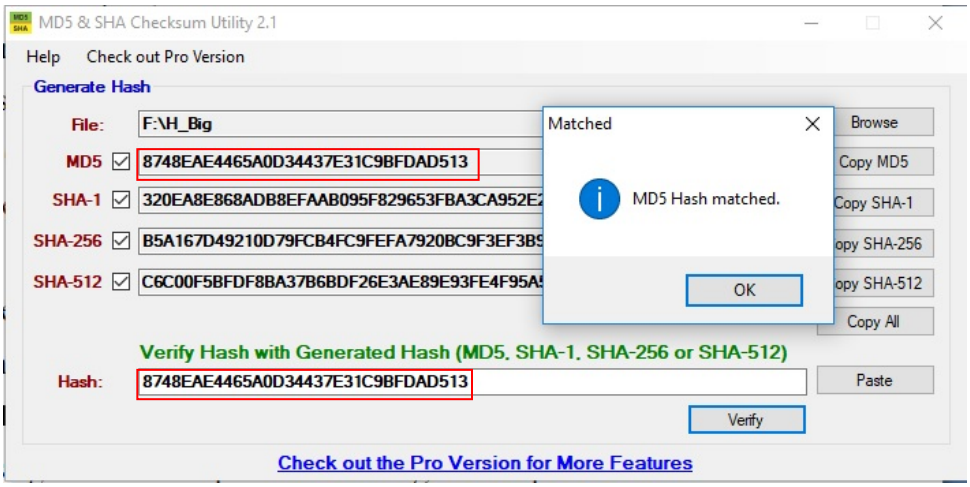


Fig 4.19: Fragmentation (before and after) Hash Verified

The first primary file with ADS '**HANDS.3gp**' is checked against the following parameters before compression and password encryption and then again when the file is compressed, encrypted, encoded and fragmented and again defragmented and decoded, uncompressed and decrypted. After that by traversing to '**HTB_HBig'** folder, **HANDS.3gp** (in this folder)

is checked**.** Hence both these files (HANDS.3gp, HANDS.3gp) are measured in the following table to verify the reliability of the compression and encryption process (**Compression/Password Encryption**).

Table 4.13: Compression Test Results

	File Type/ Name with Extension	**File Hash (MD5)**	**File Size (KB)**	**File Duration (seconds)**
Original File	Video/ HANDS.3gp	4005CC3D79D1B8853D6560E831A33CC7	263	37
Retrieved File	Video/ HANDS.3gp	4005CC3D79D1B8853D6560E831A33CC7	263	37

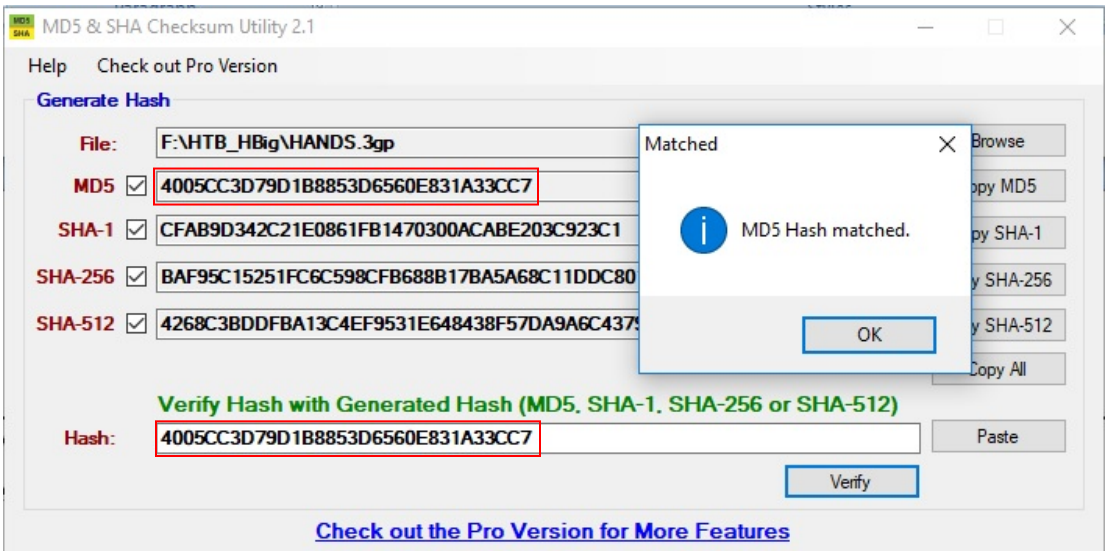


Fig 4.20: Compression (before and after) Hash Verified

4.4 Findings and Observations

Following are the some of the findings observed during the course of dissertation:

- As ADS, archived/ compressed files are studied to be most suspicious or risky, of all file types. The reason is the probability that the compressed ADS may have further nested ADS, of any depth level, having applied any of the data hiding technique that could possess the potential secrets hunted for.

- About 125MB data could be successfully retrieved from the ADS/s at once. Having value of ‘ReadCount’ larger than this, results in error of exceeding the supported range.
- Value of ReadCount parameter has considerable influence on the execution time of the Get-Content command, used for retrieving the contents of a file. With a large ReadCount value selected, the total execution time got shrunk. This can create a noticeable difference while dealing with very large sized files. Hence, one has to select the value of ReadCount intelligently by taking the file size into consideration.
- PNG files do not retrieve successfully after applying encoded fragmentation. Whereas if only encoding is applied, the .png file is retrieved with complete success.
- Main stream files, having alternate streams attached, may do cluster chaining. ADS files, themselves, also may have chained clusters.
- Main stream files with alternate streams attached may also become fragmented.
- Compressed files having ADS/s, or even further nested ADS, can be transferred over the internet through email clients.
- Archived/ compressed files containing ADS/s, or even further nested ADS, can be uploaded and downloaded via ftp clients.
- Alternate data streams can also be detected in the deleted main stream files through WinHex.
- Compressed files are normally checked against the malwares hidden inside the attached alternate streams and while the compressed files having ADS with other secrets and sensitive data are not inspected.

4.5 Recommendations

4.5.1 Data Hiding Perspective

Based on the test results and examination of procedures used for various approaches in the study, some recommendations are proposed for higher efficiency of the processes and to achieve better performance while hiding data in the NTFS alternate data streams.

- ADS should be of small size to evade suspicion. When the compressed file with ADS/s is uncompressed, the visible file is only the primary file while attached ADS/s is hidden from the end user. Whereas the end user can see the difference in the size of compressed file and uncompressed apparent primary file that can raise doubt. If the ADS/s is of small size, rather of minimal size, then one may think it benign or innocent and ignore the negligible size difference.
- Primary file (having ADS with secret data) should be of small size and innocent looking. When compressed, file with smaller size is easy to transfer between networks and is less prone to suspicion.
- It is better to compress the file having alternate data streams and then delete the uncompressed one because the ADS/s, attached to primary file that is compressed, are listed neither through the “dir /r” command nor through open source software “ADS Manager”. Even WinHex lists only one data attribute, no additional data attributes for hidden streams (attached to primary file) is visible when compressed file is viewed in WinHex.

4.5.2 Reverse Engineering Perspective

The recommendations stated above for hiding data could be adopted for reverse engineering the ADS as well, in the following contexts:

- All the main stream archived/ compressed files on the suspected system/ drive must be investigated for the presence of alternate streams within them by uncompressing these archived files.
- Small sized alternate data streams should not be left unchecked, especially when ADS are in compressed form. These alternate streams should be examined further for ADS nesting.
- All mainstream/ primary files on the suspected system/ drive should be inspected for the attached ADS, irrespective of their size.

Chapter 5

5. Conclusion and Future Endeavors

5.1 Conclusion

The leading concern linked to the digital information is to make it secure against all types of unauthorized access, modification, theft, deletion and destruction. In today's digital world, all the probable ways that could affect a system's defense must be examined. Data hiding is one such technique to compromise a system's protection by hiding the information in the places where it could be easily overlooked.

NTFS Alternate data streams provide an ideal way to conceal the data within them and hence could be used by terrorists as well as adversaries to hide their malicious activities and furtive plans. The factors that further enhance the worth of the ADS for data hiding are that they require low level of skill to create and manipulate and are less prone to suspicion for hiding high level secrets. Moreover, since NTFS is the most widely used file system, information hiding employing alternate streams could not be ignored.

Major contribution of the current study is that various fresh ways to add the covertness to the ADS are surveyed and implemented so that it cannot be detected by the current available ADS and forensic tools. The research successfully implements the concept of ADS nesting, encoded fragmentation and also the defense in depth strategy by applying the compression, password encryption, encoding, fragmentation and ADS nesting simultaneously. A comprehensive route is also proposed in the thesis to effectively analyze and retrieve the ADS and their contents.

Only by discovering all the probable digital information hiding tactics, one can employ the appropriate techniques to recover the knowledge through reverse engineering. Hence by seeing through the data hiding techniques presented in the study, the readers can get a considerate knowledge of the data hiding potential that lies within the NTFS' alternate data streams. This calls for the forensic investigators, law enforcement agencies to carry out the extensive analysis of the captured systems. It should become a policy to thoroughly scrutinize and inspect the ADS against various possibilities of data hiding tactics.

5.2 Future Endeavors

Future scope in the said domain is to design a data hiding tool by considering the approaches described in the research work. ADS and ADS nesting in combination of steganography of multimedia files can be explored further in a separate study. The current endeavor also recommends the ways to develop a detection tool to discover alternate streams concealed through different hiding techniques.

To check the presence of ADS/s in the deleted file/s and carving of such deleted files (that have attached ADS/s) can also be a promising future work direction.

References

[1] Hal Berghel, David Hoelzer, Michael Sthultz, Chapter 1 Data Hiding Tactics for Windows and Unix File Systems, *Advances in Computers, Elsevier*, Volume 74, 2008, Pages 1-17.

[2] Ewa Huebner, Derek Bem, Cheong Kai Wee, Data hiding in the NTFS file system, *Digital Investigation*, Volume 3, Issue 4, 2006, pp. 211-226.

[3] K. Duraiswamy , R. Uma Rani, Security through obscurity, 2005.

[4] Cheong Kai Wee, Analysis of hidden data in NTFS file system, 2006.

[5] Brian Carrier, File System Forensic Analysis, Addison-Wesley Professional, 2005.

[6] A. Jain and G. S. Chhabra, "Anti-forensics techniques: An analytical review," *2014 Seventh International Conference on Contemporary Computing (IC3)*, Noida, India, 2014, pp. 412-418.

[7] Noemi Kuncik and Andrew Harbison, "Counter forensics techniques – a brief overview", *Digital Forensics magazine – Grant Thornton*, 2010.

[8] K. Palmgren. (2006). Alternate Data Streams – What's Hiding in Your Windows NTFS. *White Paper, Global Knowledge Instructor, CISSP, Security+, TICSA*

[9] R. Mahajan, M. Singh and S. Miglani, "ADS: Protecting NTFS from hacking," *International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014)*, Jaipur, 2014, pp. 1-4.

[10] Piper, Scott & Davis, Mark & Shenoi, Sujeet, "Countering Hostile Forensic Techniques" in *International Federation for Information Processing, Volume 222. Advances in Digital Forensics II, (Boston: Springer),* 2006, pp. 79-90.

[11] Bem, D., & Huebner, E. Z, "Alternate data streams in forensic investigations of file systems backups", *International Conference on Computer Science and Information Systems*, Athens, Greece, 2006, pp. 443 – 454.

[12] H. Berghel and N. Brajkovska, "Wading into alternate data streams", *Communications of the ACM*, vol. 47, no. 4, p. 21, 2004.

[13] M. Broomfield, NTFS Alternate Data Streams: focused hacking. *Network Security*. 2006(8), 2006, pp. 7–9.

[14] A. I. Martini, A. Zaharis and C. Ilioudis, "Detecting and Manipulating Compressed Alternate Data Streams in a Forensics Investigation," *2008 Third International Annual Workshop on Digital Forensics and Incident Analysis*, Malaga, Spain, 2008, pp. 53-59.

[15] Ryan L. Means. Alternate Data Streams: Out of the Shadows and into the Light. *SANS Institute*, 2003. As part of GIAC practical repository.

[16] Damon Martin. Windows, NTFS and Alternate Data Streams. Published in *SANS Institute* 2000 – 2002. As part of GIAC practical repository.

[17] Gregory Conti, Erik Dean, Matthew Sinda, and Benjamin Sangster. “Visual Reverse Engineering of Binary and Data Files”. In: *Goodall J.R., Conti G., Ma KL. (eds) Visualization for Computer Security. Lecture Notes in Computer Science*, vol 5210. Springer, Berlin, Heidelberg, pp. 1-17, 2008.

[18] Cook R. Alternate Data Streams: Threat or Menace? 2005. Retrieved from http://www.informit.com/articles/printerfriendly.asp?p=413685

[19] John Marlin, "Alternate Data Streams in NTFS", *Blogs.technet.microsoft.com*, 2013. [Online]. Available: https://blogs.technet.microsoft.com/askcore/2013/03/24/alternate-data-streams-in-ntfs/.

[20] Tejpal Sharma, Ranjeet Singh, Gaurav Goel Analysis of Data Hiding With ADS in NTFS. *International Journal of Computer Science & Communication Networks*,Vol 6(2),45-48, 2016.

[21] S. Mahant and B. Meshram, ADS Examiner: Tool for NTFS Alternate Data Streams Forensics Analysis. *International Journal of Engineering Research & Technology (IJERT)*, Vol 1(4), 2012.

[22] F. Hsu, M. Wu, S. Ou and S. Wang, "Data concealments with high privacy in new technology file system", 2015. [Online]. Available: https://www.semanticscholar.org/paper/Data-concealments-with-high-privacy-in-new-file-Hsu-Wu/f71ab11e3ce4d0243f67cf72e60d5a8d1c589d89.

[23] R. Mahajan, "Stealth ADS: Enhanced framework for Alternate Data Streams," *2016 International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, Jaipur, 2016, pp. 1-5.

[24] John R. Vacca, K Rudolph. “Understanding Information-Hiding Techniques” in *System Forensics, Investigation, and Response*, 1st ed., USA. 2011. pp. 151-153

[25] "Practical Guide to Alternative Data Streams in NTFS", *Irongeek.com*, 2016. [Online]. Available: https://www.irongeek.com/i.php?page=security/altds.

[26] C. Humphris, M. Reddy, J. Hansen, H. Hutchings, M. James, N. Paiotta and G. Lukosek, "Software Security and Reverse Engineering | Source Code | Computer Program", *Scribd*, 2018. [Online]. Available: https://www.scribd.com/document/59779886/Software-Security-and-Reverse-Engineering.

[27] E.J. Chikofsky and J.H. Cross, "Reverse Engineering and Design Recovery-A Taxonomy", IEEE Software, Jan. 1990, 13-17.

[28] I. Raz, "Introduction to Reverse Engineering", *Cs.tau.ac.il*, 2011. [Online]. Available: https://www.cs.tau.ac.il/~tromer/courses/infosec11/lecture9.pdf.

[29] M. Popa, “Binary Code Disassembly for Reverse Engineering”, *Journal of Mobile, Embedded and Distributed Systems*, vol. IV, no. 4, 2012.

[30] I. Thompson and M. Monroe, “FragFS: An advanced Data Hiding Technique”. 2006.

[31] M. Rouse, "What is information security (infosec)? - Definition from WhatIs.com", *SearchSecurity*, 2016. [Online]. Available: https://searchsecurity.techtarget.com/definition/information-security-infosec.

[32] "Get-Content (Microsoft.PowerShell.Management)", Docs.microsoft.com. [Online]. Available: https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.management/get-content?view=powershell-6.

[33] “Get-Content”, [Online]. Available: https://ss64.com/ps/get-content.html

[34] P. Arntz, “Introduction to Alternate Data Streams”, 2016. [Online]. Available: https://blog.malwarebytes.com/101/2015/07/introduction-to-alternate-data-streams/

[35] “Alternate Data Streams for Managers”. Published in *SANS Institute* 2000 – 2002. As part of GIAC practical repository.

[36] Kurtis E. Kroeckel, Windows Alternate Data Streams, 2002. Published in *SANS Institute*, 2003. As part of GIAC practical repository.

[37] D. Brant, “ADS Manager”, 1980–2018. [Online]. Available: https://dmitrybrant.com/adsmanager

[38] N. Johnson, “Alternate Data Streams and the NTFS file system”, 2009. [Online]. Available: https://www.slideshare.net/nephijohnson/alternate-data-streams

[39] M. Ochsenmeier, “Windows Alternate Data Streams”, 2012. [Online]. Available: https://www.winitor.com/pdf/NtfsAlternateDataStreams.pdf

[40] L. Heddings, “How to Hide Data in a Secret Text File Compartment”, 2016. [Online]. Available: https://www.howtogeek.com/howto/windows-vista/stupid-geek-tricks-hide-data-in-a-secret-text-file-compartment/

[41] S. Garfinkel, "Anti-forensics: Techniques, detection and countermeasures," in *2nd International Conference on i-Warfare and Security*, 2007.

[42] M. Warkentin, E. Bekkering, and M.B. Schmidt, “Steganography: Forensic, Security, and Legal Issues,” *Journal of Digital Forensics, Security and Law*,3(2), 2008, pp. 17–34.

[43] Andreas Grytting Furuseth. “Steganography” in *Digital Forensics: Methods and tools for retrieval and analysis of security credentials and hidden data*, Trondheim, Norway. 2005. pp. 33.

[44] M. Alazab, S. Venkatraman, and P. Watters, “Digital forensic techniques for static analysis of NTFS images”, Proceedings of *ICIT2009, Fourth International Conference on Information Technology, IEEE Xplore*. 2009.

[45] H. Pieterse, M. Olivier, Data hiding techniques for database environments, in: *Advances in Digital Forensics VIII, Springer*, 2012, pp. 289–301.