Integration Testing with Steganographic Tools

Saugata Dutta
Research Scholar, Computer Science, OPJS University, Rajasthan, India
Dr. Om Prakash
Professor, OPJS University, Churu, Rajasthan, India

Abstract:
Steganography is the art of hiding information. It is the art of secret writing. Steganography helps in eliminating the useful and confidential information from common public or users. It is the art of concealing file, message, image or video within another file. The Cover file is the file which carries the message file. This paper discusses about hiding different format of files in different steganography tools and test the integrity of the message files inside the related application at the receiving end. This paper discuss about the experiment held using different steganography techniques with steganography tools. The objective of the experiment is to check the integrity of the message files of different formats hidden with different steganography tools and decoding at the receiving end. The experiment throws light on the condition of the file received, integrity and execution with the related application at the receiving end. The paper then shows experiment results used with image, audio, video, dual layer steganography and steganography file system followed by the conclusion whether the experiment result can reject the null hypothesis or the experiment holds no significant changes.

1. Introduction

Steganography is the practice of concealing a file, message, image, or video within another file, message, image, or video. The word steganography combines the Greek words steganos, meaning "covered, concealed, or protected", and graphein meaning "writing". The first recorded use of the term was in 1499 by Johannes Trithemius in his “Steganographia”, a treatise on cryptography and steganography, disguised as a book on magic. Generally, the hidden messages appear to be (or be part of) something else: images, articles, shopping lists, or some other cover text. For example, the hidden message may be in invisible ink between the visible lines of a private letter [1]. There are several ways to hide data, including data injection and data substitution. In data injection the secret message is directly embedded in the host medium. The problem with embedding is that it usually makes the host file larger; therefore, the alteration is easier to detect. In substitution, however, the normal data is replaced or substituted with the secret data. This usually results in very little size changes for the host file. However, depending on the type of host file and/or the amount of hidden data, the substitution method can degrade the quality of the original host file. Generation technique generates a container file based on the covert data. There is no original container file. However, it is time consuming and complex to develop. The DCT is "a technique for expressing a waveform as a weighted sum of cosines”. In a JPEG file, the image is made up of DCT coefficient. When a file is steganographically embedded into a JPEG image, the relation of these coefficients is altered. Instead of actual bits in the image being changed as in LSB steganography, it is the relation of the coefficients to one another that is altered. In addition to DCT, images can be processed with fast Fourier transform (FFT). FFT is "an algorithm for computing the Fourier transform of a set of discrete data values". The FFT expresses a finite set of data points in terms of its component frequencies. It also solves the identical inverse problem of reconstructing a signal from the frequency data.

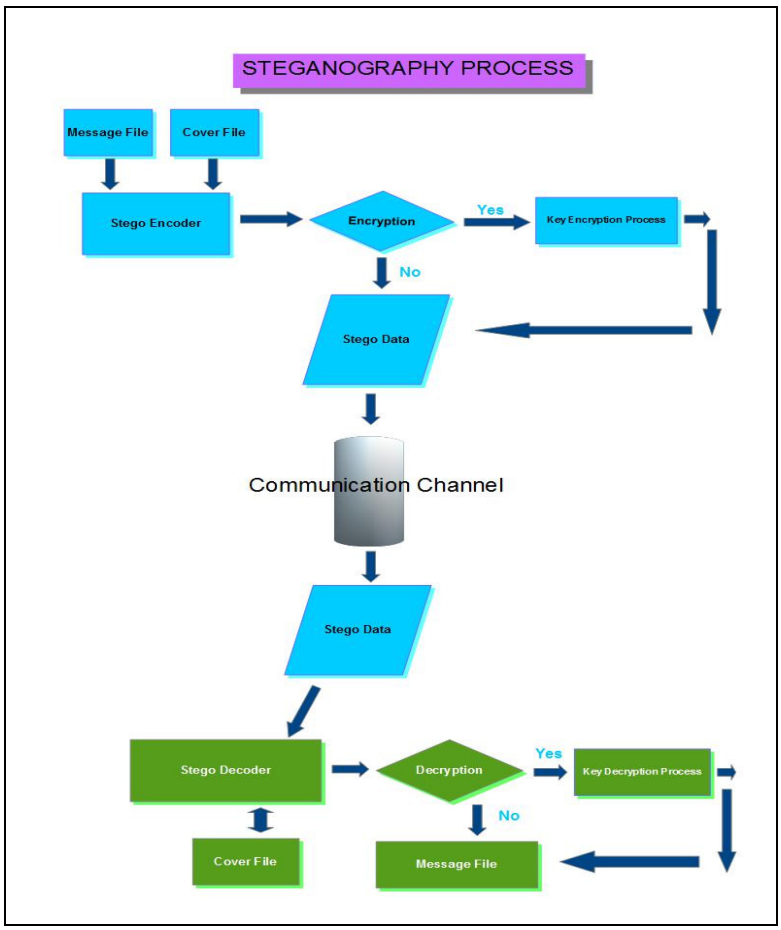


Figure 1: Basic steganography process

Steganography for audio has grown as a new technology that involves different algorithms such as echo encoding and phase encoding that are different from the algorithms used for image steganography. As audio techniques have been developed for audio streaming on the internet for radio station, incorporated into social networking and communication applications like Skype, Google hangouts and also in VOIP (Voice over IP) communications. In Video Steganography it can be used in video files, because as we know, AVI files are created out of bitmaps, combined into one piece, which are played in correct order and with appropriate time gap. A Single file is broken into frames which can be saved as BMP files. If algorithm for hiding data in digital pictures, we can hide our message in bitmap obtained in this way, and then save it into new AVI file. In steganographic file system, there is a storage mechanism designed to give the user a very high level of protection against being compelled to disclose its contents. It will deliver a file to any user who knows its name and password; but an attacker who does not possess this information and cannot guess it, can gain no information about whether the file is present, even given complete access to all the hardware and software. We provide two independent constructions, which make slightly different assumptions. Practical uses for this technology, for instance, one can store password information on an image file on your hard drive or Web page, some legal documents, some plans to be shared which may be hidden and documented. Applications where encryption not appropriate (or legal), Stego can be used for covert data transmissions. Although this technology was used mainly for military operations, it is now gaining popularity in the commercial marketplace. As with every technology there are illegal uses for Stego as well. It was reported that terrorists use this technology to hide and send their attack plans. The Purpose of our present research is to check the integrity of the message file of almost all possible formats after being attached with the cover file and hidden and encrypted with steganographic tools and sent over the network with different methods. Once the file reaches the destination, the file is then attached with steganographic tools to detach the file from the cover file after decryption. The original file at destination end is then attached with the relevant application. The size of file may be of any size.

Our Purpose is to see whether the file works in the same manner without any loss and can be integrated to the application as before it was unhidden and unencrypted. Files will be attached to different formats of cover file like Image, Video, Audio and TCP/IP packets. We will also check the condition of the cover files (image, video and audio files) with steganographic tools compared to the original. In another part of the research, message file(s) will be kept in the Stego File systems to test the file integration with the application and also will perform dual layer steganography and encrypt for better security and then the integrity testing with the application will be tested.

2. Review Literature

Qilin Qi proposed that steganographic message can be inserted in multimedia cover signals such as audio, image and video. There is a method called discrete spring transform which is used to remove the potential dangerous hidden information keeping the digital data in high visual quality. The Proposed transform causes the numerical value to be changed dramatically and the hidden information is not able to be recovered while at the same time the visual image quality is maintained [2]

Manisha Saini and Gaurav Saini explored tools being used for primarily data hiding and encryption. There are some common tools being used for Images, audio and video files. It also shows the virtual disk to be hidden in WAV (Audio) files. StegFs which is a steganographic file system for Linux. Spam Mimic is a popular steganographic tool that allows user to hide information in SPAM messages [3].

Abbas Cheddad, Joan Condell, Kevin Curran and Paul Mc Kevitt explored the emerging techniques DCT (Discrete Cosine Transform), DWT (Discrete Wave Transform) and adaptive steganography are not to prone to attacks, especially when the message is small. This is because they alter the coefficients in the transform domain, thus image distortion kept to minimum. Generally, these methods tend to have a lower payload compared to spatial domain algorithms [4].

Ketki Thakre and Nehal Chitaliya worked on dual image steganography where the proposed method embeds data in two cover images using four bit LSB technique. The secret data is hidden in binary form in two cover images due to which double protection has been provided to confidential data. The proposed scheme shows that it can be a good alternative for secure communication where two level of security is obtained [5].

Kaustubh Choudhary showed how an innocent looking digital image hides a deadly terrorist plan. It analyses the methods and reasons the terrorist is relying on it. Even a simple steganographic algorithm can hide data so efficiently that steganalysis become very difficult on various accords of time, computation power and money. The Real world terrorism uses more complex and innumerable steganographic algorithms [6].

Govinda Borse, Kailash Patel and Vijay Anand explored data hiding techniques were being used from ancient ages to recent times. In ancient times data hiding techniques used Wax Tablets, Invisible ink, Microdots, Null Ciphers and Girolama Cardano as simple as piece of paper with holes cut in it. In recent times the data hiding techniques has changed from the past in images, music, videos, storage, and VOIP and TCP/IP packets [7].

Erin Michaud explored some of the more common methods of data hiding using wide spread file formats and easily available tools as an introduction to the primary concepts of steganography. This exploration served as the starting points to the exploration of more complex steganographic techniques, for e.g. the use of network packets and unused hard disk space as cover medium or the more complex methodologies used on our standard image and audio files [8].

Ms. Anshu Sharma and Dr. Deepti Sharma proposed a system where it used LSB Technique for embedding the Text message and the three techniques i.e. AES, DES and Blow fish for encrypting and Decrypting the Information before sending it to the client. This showed how various techniques can be combined to provide more security by combining cryptography with steganographic techniques [9].

Matus Jokay and Martin Kosdy designed and implemented the steganographic layer based on the jpeg files that can be used directly as the virtual disk or in a connection with existing disk encryption solutions as a hidden storage medium [10].

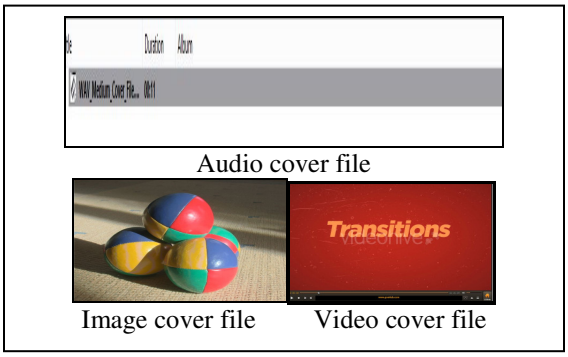


Figure 2: Audio, image and video cover file

Barret Miller developed a program that used the interface id portion of the source address fields of IPv6 packet as a covert channel in which to transmit secret data. The program can both transmit and receive secret data in the source addresses of IPv6 packets in two different and important ways. In the first way the program can be configured to work is to embed the message by changing the MAC address of the originating host to the secret message. In the second way the program can embed messages is by explicitly creating IPv6 packets containing the message in the source address field, which are then injected into the network [11].

3. Experiment

The Objective of the research is to test the integrity of the message file(s) once when hidden and encoded in a steganography tool and passed through the network and decoded at the receiver's end with the same steganography tool and finally attached in the related application. Various Steganography tools are analyzed where eleven best steganography tools are selected for usage. These tools

include image steganography, audio steganography, video steganography and usage of steganography file system. The next step is to identify the various file formats to be used for the message file(s) in the experiment. The identification includes not only the common file formats but also some of the rare and uncommon file formats are chosen. The sample message files are one hundred and three different types of file formats. The objective of the research is to have a full proof of checking the integrity and execution of the message file with different permutation and combination of steganography techniques such that the experiment can be concluded with a logical and significant evidence. The next step is to choose the cover file. In three broad categories (Image, Video and Audio) cover files are selected of different attributes. For Image steganography JPG and BMP formats are selected of different sizes. The dimensions include (800 x 600, 1920 x 1080, 3384 x 2256, 4692 x 3214, 5152 x 3439, 5494 x 5839, 7360 x 4192). Video Steganography cover files includes of MP4 formats with frame width mostly 960 and frame height 540 and frame rate 29/Sec while some are 15/Sec. In some instances of video steganography multiple cover files are being used. Sound steganography cover files includes WAV format files with 2 channel and 1411 Kbps of bit rate. SHA-1 Hash generator being used to generate checksum and verify the integrity by using verification file (. sha) which produces a 160-bit (20 byte) hash value. Hasher software being used to generate hash value for sample message file. The Second Layer of integrity check being used with file comparing utility Winmerge.

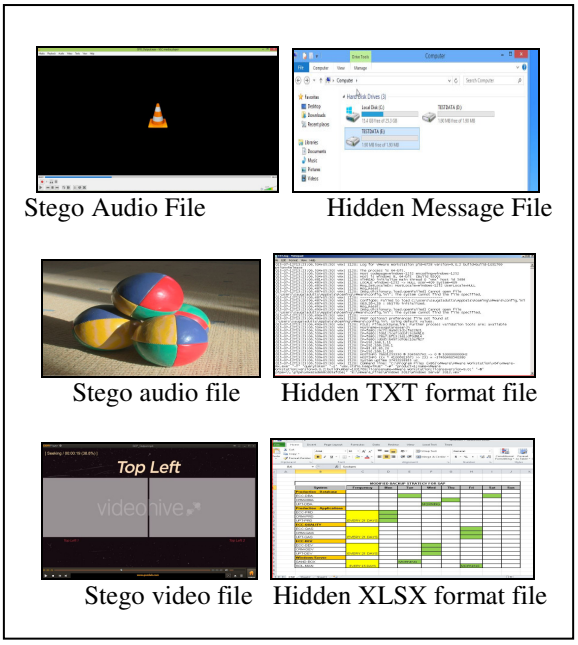


Figure 3: Stego audio, image and video encoded files with hidden sample message files

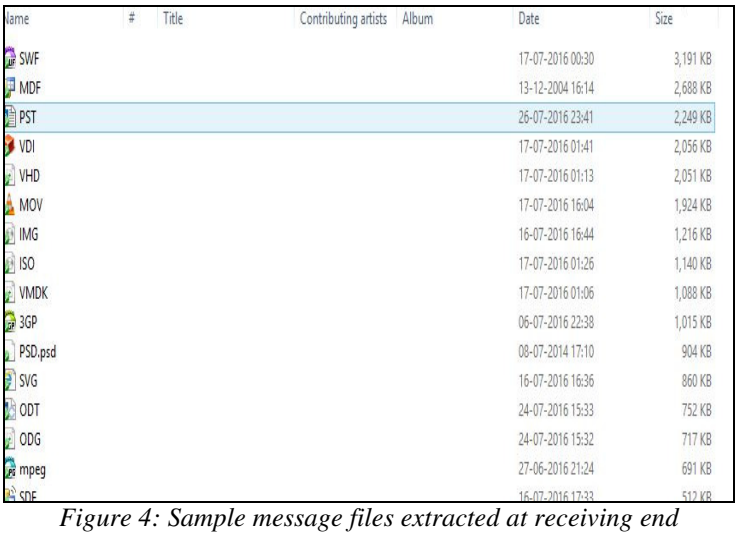


Figure 4: Sample message files extracted at receiving end

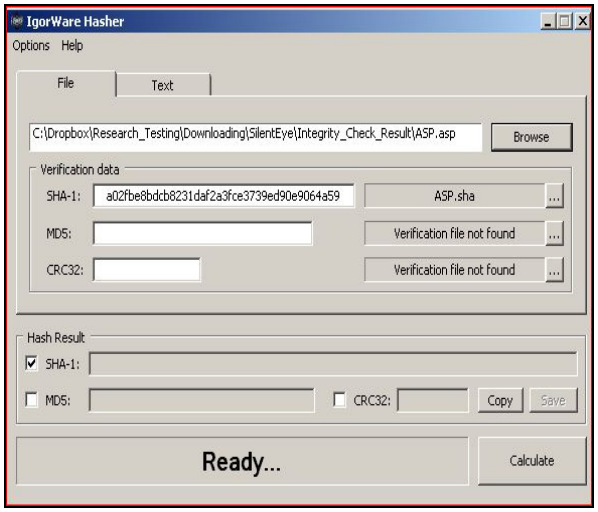


Figure 5: Integrity checking process of the message file with SHA-1 at the receiving end

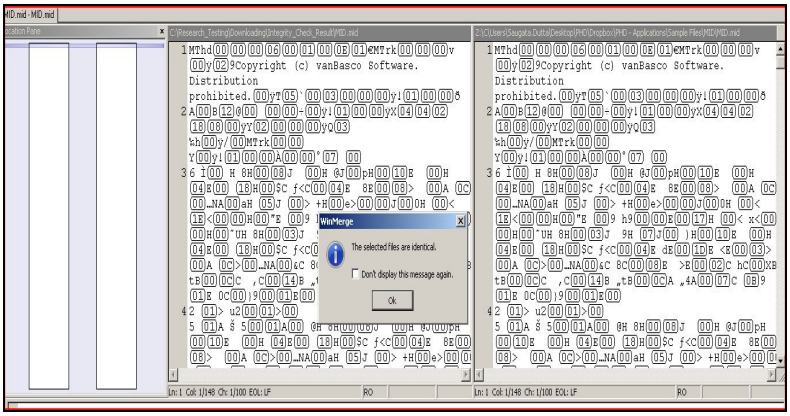


Figure 6: File comparison with the extracted message file against the non- encoded sample message file

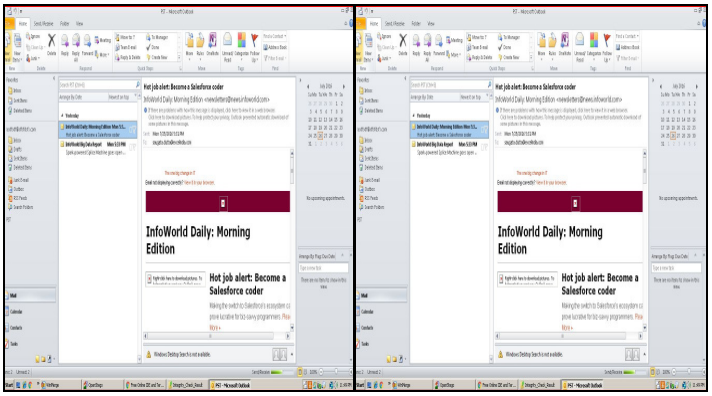


Figure 7: Message file execution with the related application comparing at the sending (before encoding) and receiving end (after decoding and integrity check)

The technique of steganography process includes image steganography, audio steganography, video steganography, dual layer video steganography and dual layer audio steganography and usage of steganography file system. The Experiment Process starts with hashing the sample files at first. These hash files are then kept separately in a secured location. For image, video and audio steganography each and every sample message files of different formats are hidden with each respective steganography softwares being used in the experiment process. They are in turn encrypted with AES-256 encryption and in some cases encryption is not applied. The stego output file format for all steganography process is usually the same format of the cover file. The output stego file along with the concerned hash file is then sent over the network and uploaded in cloud drive. The same output stego file is then downloaded at the receiver end and identical steganography software being used to decode the stego file. Once the stego file is decoded at the receiving end and message file is extracted where it is isolated from the cover file along with the decryption process (if encrypted) is then kept in a separate location. The extracted message file undergoes 3 layers of integrity check. The first layer of check

is the SHA-1 hash value check with hasher software. Once the message file passes the check, the second check starts with the file compare utility Winmerge. The original sample message file before the steganography process is downloaded and compared with the extracted message file. The last step includes the execution of the extracted message file with the related application.

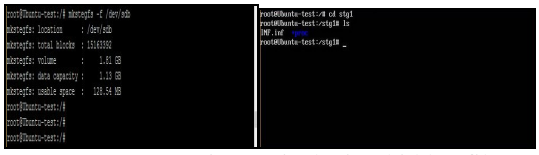
The dual layer image and video steganography process includes two layer encoding and decoding process and checking the integrity of the extracted message file. The first step is to encode the sample message file with a cover file including encryption. The stego output file is then re-encoded with a cover file and applied encryption. The output stego file is then transmitted to the cloud storage along with the respective hash file of the sample message file encoded. The output stego file and respective hash file is then downloaded and decoded and decrypted with the identical steganography software where the output stego file is then isolated from the cover file.

Layer 1 stego image file

Layer2 stego image file

Extracted png format message file

Figure 8: Dual layer steganography



StegFs Preparation Displaying hidden file

Figure 9: StegFs file system

The same file is then re-decoded and the message file is then decrypted and extracted while isolated from the second layer of cover file. The final output message file is then passed through the hash integrity test and eventually the file comparing utility with the original sample file. Once the process is finished the message file is then executed in the related application. The file should execute and work in the same way as before the steganography process.

The Stego File system experiment process starts with implementation of StegFs software in Linux operating system. The StegFs software is installed and implemented as an ext2 file system and mounted as a mount point. The sample message files are copied and kept in the Stego file system. The message files become invisible when remounted the StegFs file system and can only be called and opened with the file name. The name of the copied sample message files should be known which is a pre-requisite. The sample message file is encrypted and it is called with the following pattern "<filename>@<Password>". The Stego file system is then used for some input / output operations. The extraction process finally starts by calling the filename along with the encrypted password for the files which are encrypted. The Message files are then copied from the StegFs, the extracted message files are then pass through the integrity check with the respective hash file by using hasher. Once the test is passed the message file is then compared with the original sample file through the file comparison utility Winmerge. Finally, after the step is completed the extracted message file is then executed in the related application. The entire experiment results are recorded to conclude.

4. Tools and Samples Used

The sample message files used for this experiment are of one hundred and their different types of file formats.

Message files	Size	Message files	Size
.CAB	28 KB	.SVG	804 KB
.7z	790 B	.SCAD	1 KB
.APK	68 KB	.STL	312 KB
.BKF	24 KB	.SFX	52 KB
.ZIP	1 KB	.CMD	2 KB
.RAR	923 B	.INF	584 B
.RPM	65 KB	.REG	2 KB
.TAR	21 KB	.SEP	3 KB
.ISO	1.1 MB	.SCR	10 KB
.IMG	1.2 MB	.PNF	7 KB
.DXF	378 KB	.CHM	64 KB
.DWF	135 KB	.ACCDB	316 KB
.DWG	185 KB	.DBF	808 B
.MDB	232 kb	.ODS	45kb
.MDF	2.6 MB	.ODG	717 kb
.SQL	370 B	.PS1	2 KB
.SDF	512 KB	.SH	48 B
.BSON	1 KB	.PPK	1 KB
.AI	242 KB	.CRT	2 KB
.CDR	3 KB	.AMR	55 KB
.PSD	904 KB	.MP3	193 KB
.PDF	2 KB	.WMA	157 KB
.WLMP	7 KB	.WAV	108 KB
.CSV	538 B	.MID	38 KB
.DOC	27 KB	C	58 B
.DOCX	13 KB	.BAT	179 B
.LOG	41 KB	.JAVA	142 B
.OTF	3 KB	.PHP	270 B
.TTF	3 KB	.PY	234 B
.GPX	7 KB	.JS	88 B
.KML	35 KB	.SXC	11 KB
.BMP	32 KB	.XLS	27 KB
.GIF	33 KB	.XLSX	11 KB
.ICNS	38 KB	.3GP	1 MB
.ICO	1 KB	.AVI	225 KB
.PNG	3 KB	.FLV	139 KB
.JPEG / .JPG	34 KB	.MOV	1.9 MB
.LNK	1 KB	.MPG	690 KB
.URL	133 B	.MP4	205 KB
.COM	1 KB	.SWF	3.1 MB
.JAR	982 B	WMV	91 KB
.WAR	1 KB	VHD	2 MB
.TPL	4 KB	VMDK	1.1 MB
.MSG	11 KB	VDI	2 MB
.EML	193 B	HTML	78 B
.PST	265 KB	ASP	103 B
.PPS	74 KB	ASPX	6 KB
.PPT	40 KB	JSP	77 B
.PPTX	53 KB	.BAK	768 B
.ODP	58 KB	.DAT	741 B
.ODT	752 KB	.TXT	225 KB
.TMP	808B		

Table 1: Sample message file formats

There are various steganography utilities, for the purpose of experiment eleven different steganography utilities are used.

Tool Name	Details
Open Stego (V0.6.1)	OpenStego is a free steganography tool. It provides two main functionalities that is data hiding and digital watermarking. It supports file types for cover are: BMP, GIF, JPEG, JPG, PNG, and WBMP. It supports password-based encryption of data for additional layer of security. DES algorithm is used for data encryption, along with MD5 hashing to derive the DES key from the password provided.
Silent Eye(V0.4.1)	SilentEye is an easy to use cross platform steganography program. It supports various common image and audio formats including BMP, JPG, PNG, GIF, TIF, and WAV. The default encoding image format can be configured between JPG or BMP and similarly for audio encoding the default format is WAV only. The software also has an option for password encoding.
Xiao Steganography(V2.6.1)	Xiao Steganography is a simple to use free software to hide secret files in BMP images or in WAV files with encryption support. The Supported encryption algorithms are RC4, Triple DES, DES, Triple DES 112, RC2 and hashing SHA, MD4, MD2, MD5, algorithms.
DeepSound (V2.0)	DeepSound is a steganography tool and audio converter that hides secret data into audio files. The application also enables you to extract secret files directly from audio files or audio CD tracks. DeepSound also support encrypting secret files using AES-256(Advanced Encryption Standard) to improve data protection. The application additionally contains an easy to use Audio Converter Module that can encode several audio formats (FLAC, MP3, WMA, WAV, and APE) to others (FLAC, MP3, WAV, APE).
Clotho (V2.4)	Clotho is a powerful but easy to use tool for Windows to hide important or sensitive files into images, audio, executable or in other various types of files. The program gives you the possibility to specify the file where the sensitive data should be hidden, namely MP3, WAV, MID, OGG, JPEG, PNG, GIF, BMP, MP4, FLV, AVI, ZIP, RAR, EXE, MSI, DLL, or other file formats. Password encryption is also possible.
OpenPuff (V4.0)	Openpuff is a powerful Image, audio and video steganography too which has features like HW seeded random number generator (CSPRNG), Deniable steganography, Carrier chains (up to 256Mb of hidden data), Carrier bits selection level, Modern multi-cryptography (16 algorithms), Multi-layered data obfuscation (3 passwords), X-squared steganalysis resistance. It supports format like Images (BMP, JPG, PCX, PNG, TGA), Audio support (AIFF, MP3, NEXT/SUN, WAV), Video support (3GP, MP4, MPG, VOB) and Flash-Adobe support (FLV, SWF, PDF)
StegoPNG (v11.22d)	Stego PNG allows to hide a file of any type in a PNG or a BMP image file. The steganography software inserts data in the image file in a random way depending on a key (which is also used to encrypt the data), and changes other bits of the image file to compensate for the modifications induced by the hidden data so as to avoid modifying statistical properties of the image file. It is thus more secure than most steganography programs.
SteganoG (V1.21.0)	SteganoG store confidential data of any kind in a bitmap file that the image appears to be unchanged. A powerful compression and an adjustable image quality will also allow to save relatively large amounts of data. For data security, RC4 encryption method, Blowfish, TEA, Twofish and Skipjack.
SteganPEG (V1.0)	SteganPEG is an application of Steganography to JPEG images. The software can hide personal, private, or sensitive files in any JPEG image without changing its quality and size considerably. The main reason for choosing the JPEG image format is that it is a compressed image format that is very popular all over the world today.
OurSecret (V2.5.5)	Our Secret (formerly Steganography) hides text files or files such as video, audio, image and others in file. It is designed to hide and send sensitive files or messages. This allows to encrypt sensitive information, while at the same time hiding it in a file will evade suspicious. The program gives options to hide any file format with password encryption.
StegFs	StegFs is a steganographic file system in userspace which uses the FUSE library. Steganographic file systems are one step ahead of (or beyond) traditional encrypted file systems as they aim to grant the user plausible deniability of the files within

Table 2: Steganography tools used

IgorWare Hasher is a free SHA-1, MD5 and CRC32 hash generator for Windows, both 64bit and 32bit version is available. IgorWare Hasher is used to generate checksum for single file and verify its integrity by using verification file (.sha, .md5 and .sfv) generated by Total Commander, with support for UTF-8 verification files. Verification files will be loaded automatically if found.

Figure 10: Hasher Utility

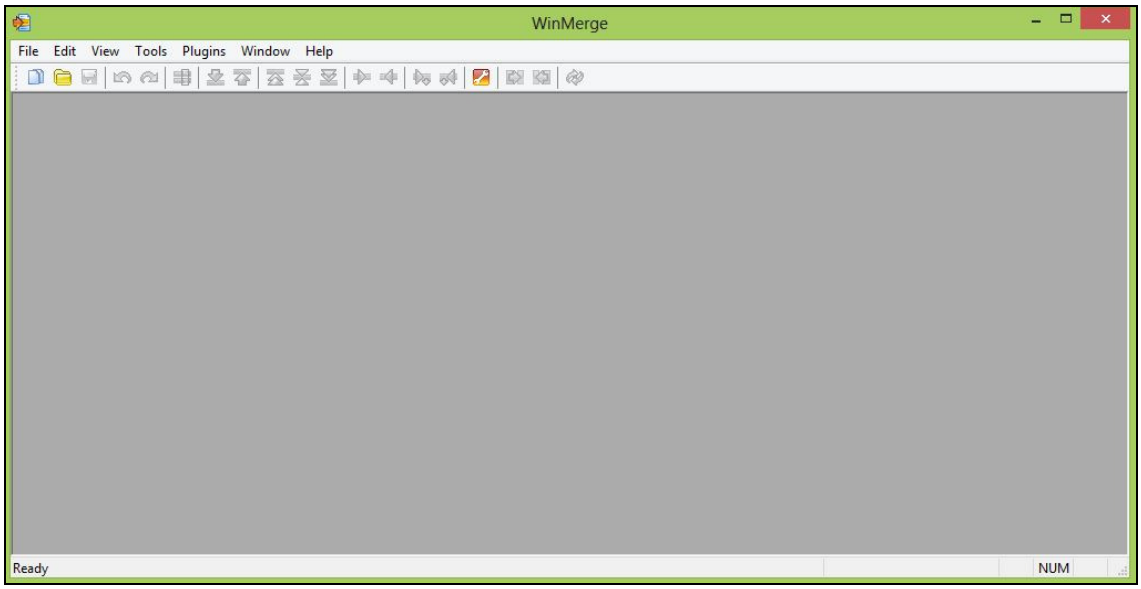


Figure 11: Winmerge utility

WinMerge is a free software tool for data comparison and merging files. It is useful for determining what has changed between versions and contents which in this experiment is being used to test the source and destination message files after the process of steganography.

5. Experiment Result and Analysis

The experiment result for all the software tools showed the capacity to hide files and decode at the receiving end, however there are some results where the process has failed. Broadly the result is categorized into three categories. The First category where the normal hiding techniques has been applied with cover image, audio and video files with one hundred and three different formats of message files and eight different sample steganography tools. The second category justifies the dual layer steganography techniques applied in one hundred and three different formats of message files and two different steganography tools in testing the integrity and thirdly the usage of steganography file system to test the integrity of one hundred and three different sample message file formats.

In the first category, the average success rate is 87% and failure rate is 13%. Openstego where image steganography has been applied yield a success rate of 100% with no failures at the receiving end. File format mov and swf files failed to encode at the receiving end which is beyond the scope of the experiment. Silent eye with sound steganography technique yield a success rate of 98% and failure rate of 2%. Clotho which uses video steganography techniques experienced a success rate of 100% of integrity testing including file comparison and file execution at the receiving end. Openpuff uses video steganography techniques also resulted a success rate of 100% of integrity testing including file comparison and file execution at the receiving end. The cover video file size for Openpuff used are very high as compared to Clotho because of the file hiding ratio is too low as compared with other utilities. Deepsound a sound steganography utility resulted success rate of 98% and failure rate of 2%. The message file format svg and swf were failed to decode.

XIAO, a sound steganography tool resulted a success rate of 100% along with StegoPNG, an image steganography tool which resulted the same 100% success rate. SteganoG on the other hand has a very low success rate of only 3% and failed rate of 97%. These files experienced corruption when extracted at the receiving end.

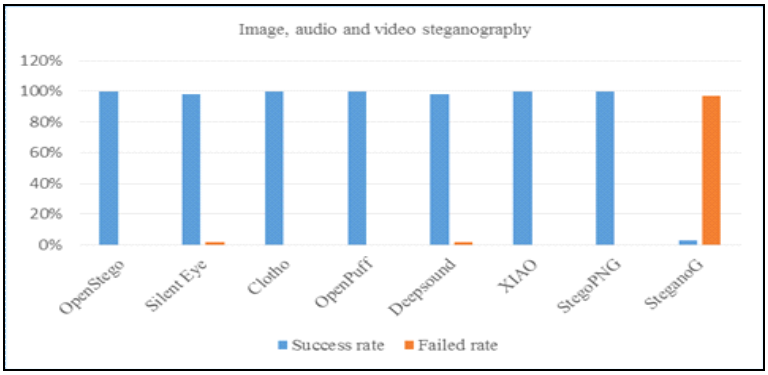


Figure 12: File integrity result of image audio and video steganography technique

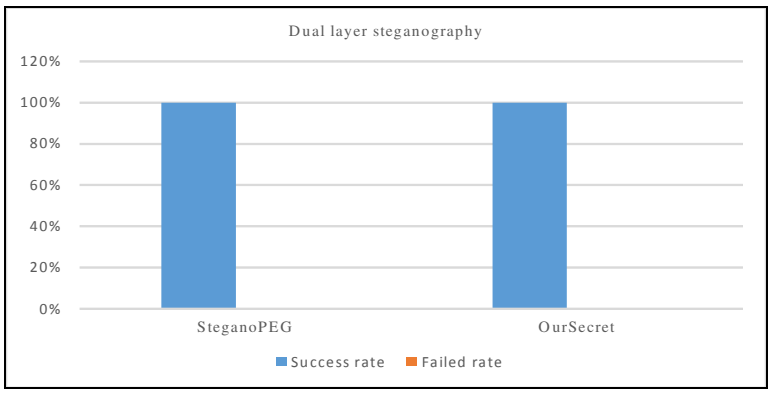


Figure 13: File integrity result of dual layer steganography technique

The second category of experiment result throws light on dual steganography techniques. SteganoPEG, an image steganography tool is being used to repeat the steganography process twice. The cover files were JPEG format in both the layers. This techniques of file integrity testing resulted success rate of 100%. The image cover file size in the second layer was high as compared to the first layer to accommodate both the message and cover file. Our secret, a video steganography tool being used for dual steganography techniques and resulted a success rate of 100% as well. The modus operandi for both the dual steganography implementation was same.

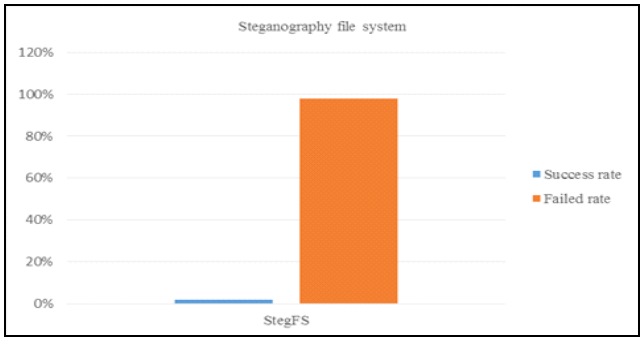


Figure 14: File integrity result of steganographic file system technique

The third category is implementation of StegFs, a steganographic file system. Sample message files were hidden in the file system which is basically the acid property of StegFs. The success rate of extracting the file after some file operations done on the file system is significantly poor. The success rate of only 2% and failure rate of 98% was alarming. The integrity testing of files extracted failed all stages of experimentation stages.

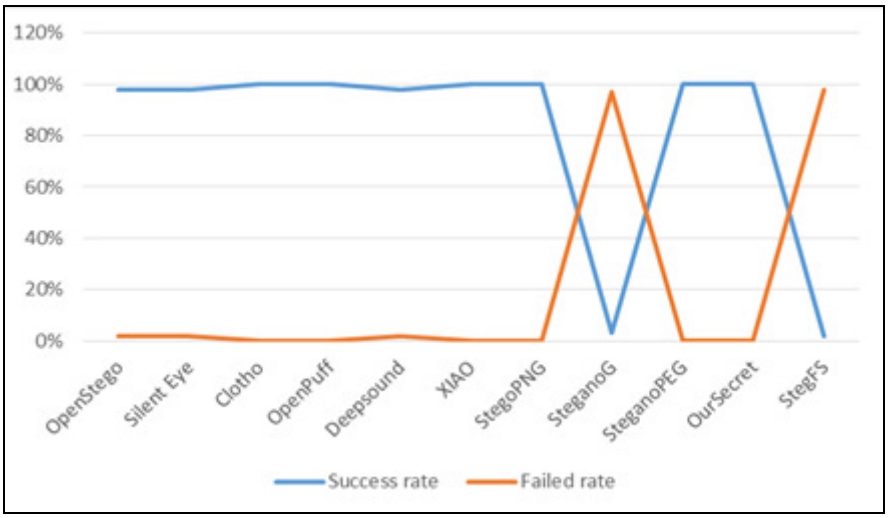


Figure 15: Overall file integrity result with applied steganography techniques

At the end of the experiment the results are clear and more prominent and proved that the result showed no significant changes. To statistically prove the same two statistical test has been carried out with the sample data file formats along with the eleven samples steganography utilities used in the experiment.

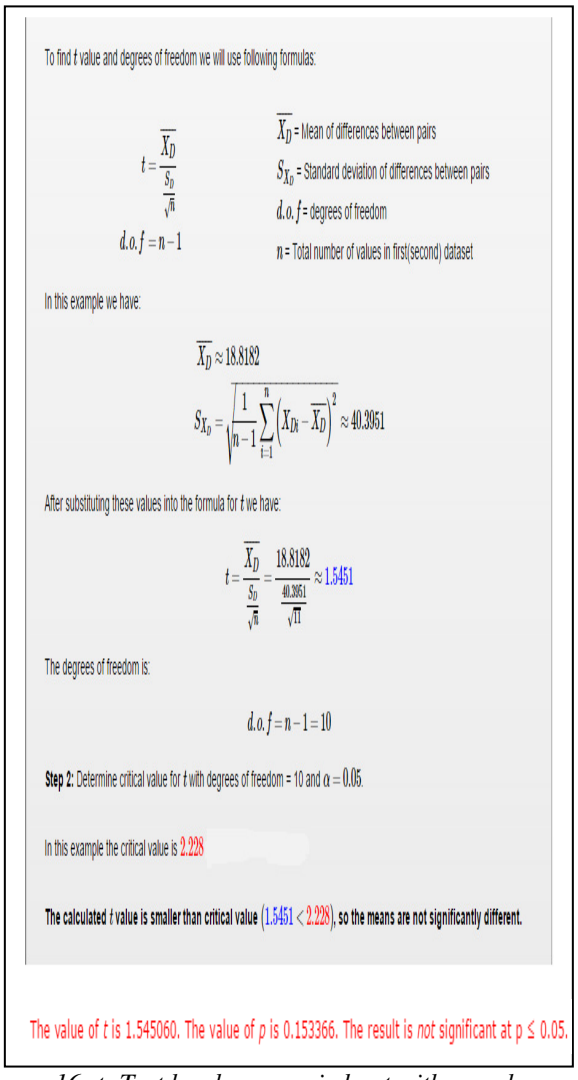


Figure 16: t- Test has been carried out with sample results

The result of the critical value is 2.228. The result showed that the calculated t value is smaller than the critical value (1.5451<2.228) which shows that means are not significantly different. The value of p with the associated t value generated 0.153366 which proves that there are no significant changes.
Next statistical test which is carried out is ANOVA test which shows the analysis of variance and determines that any of the differences between the means are statistically significance comparing the p value to the significance level to assess the null hypothesis.

The ANOVA result shows that the f-ratio value is 2.38721 and the related p value extracted from the statistical analysis is .138009 which shows the result is not significant.

Summary						
Groups	Count	Sum	Average	Variance		
Observed	11	926	84.181818	1631.763636		
Expected	11	1133	103	0		
ANOVA						
Source of variation	SS	df	MS	F	P-value	F crit
Between groups	1947.6818	1	1947.6818	2.387210715	0.1380086	4.3512435
Within groups	16317.636	20	815.88182			
Total	18265.318	21				

Table 3

6. Conclusion

The experiment, analysis and results shows that null hypothesis gets failed to reject. There are no significant changes in the result. Files of any format when attached with steganography tools with or without encryption and sent over to the receiving end has a clear chance of integrity of the files to be retained. The same holds for the dual layer of steganography where the file undergoes double steganography process and the integrity still is retained. The experiment also concludes even for large message files like ISO, VHD etc. the integrity of the files remained intact with some quality changes in the cover file. The steganography file system however failed in integrity testing and also showed that files of any format is stored along with some IO operations in the Stego file system has lesser or no chances of integrity and recovery.

7. Acknowledgements

I am thankful to my project guide who has supported me enormously in the experiment and also my sincere thanks to professors and friends for the encouragement. This work was supported in part by a grant from the National Science Foundation.

8. References

i. http://en.wikipedia.org/wiki/Steganography
ii. Qi, Qilin., “A Study on counter measures against steganography: An Active Warden Approach”, The Graduate College at the University of Nebraska, United States, Degree of Master of Science, 2013, pp. 63-65.
iii. Saini, Manisha. and Saini, Gaurav., “Steganography and tools used for steganography”, International journal of scientific and engineering research, January 2014, pp. 1693-1697.
iv. Cheddad, Abbas., Condell, Joan., Curran, Kevin., and Kevitt, Mc. Paul., “Digital image steganography: Survey and analysis of current methods”, Signal Processing, 2010, pp. 727-752.
v. Thakre, Ketki., and Chitaliya, Neha., “Dual Image Steganography for communicating high security information”, International journal of engineering and innovative technology, July 2014, pp. 7-12.
vi. Choudhary, Kaustubh., “Image Steganography and Global Terrorism”, Global Security Studies, 2012, pp. 115-135.
vii. Borse, Govinda., Anand, Vijay., and Patel, Kailash., “Exploring an ancient art of hiding information from past to the future”, International journal of engineering and innovative technology, 2013, pp. 192-194.
viii. Michaud, Erin., “Current Steganography Tools and Methods”, SANS institute, April 2003.
ix. Sharma, Anshu., and Sharma, Deepti., “Research on Analysis of Different Image Steganography Techniques”, International Journal of Engineering Sciences and Research Technology, June 2014, pp. 570-577.
x. Jokay, Matus., and Kosdy, Martin., “Steganographic file system based on JPEG files”, Tatra Mountains Mathematical Publications, 2013, pp. 65-83.
xi. Miller, Barret., “Steganography in IPV6”, Bachelor of Science in Computer Science, University of Arkansas, Fayetteville, United States, 2008.