Development/Implementation of Custom Steganography Program

Instructor: Jeremy Rasmussen

University Of South Florida

April 7, 2015

CIS4364.901S15

Cy Scott
Brandon Hunt
Justin Wetzel
Ised Gongora

Abstract

“Steganography is the art and science of writing hidden message in such a way that one part from the sender and intended recipient even realizes there is a hidden message.”

There are often cases in steganography, where the message itself may not be difficult to decode, but most people would not detect presence of the message. While cryptography provides privacy, steganography is intended to provide secrecy. Computer programs can encrypt a message using cryptography, and hide the encryption within an image using steganography.
This project determines a way to hide a file such as audio, video, digital image etc.

Motivation

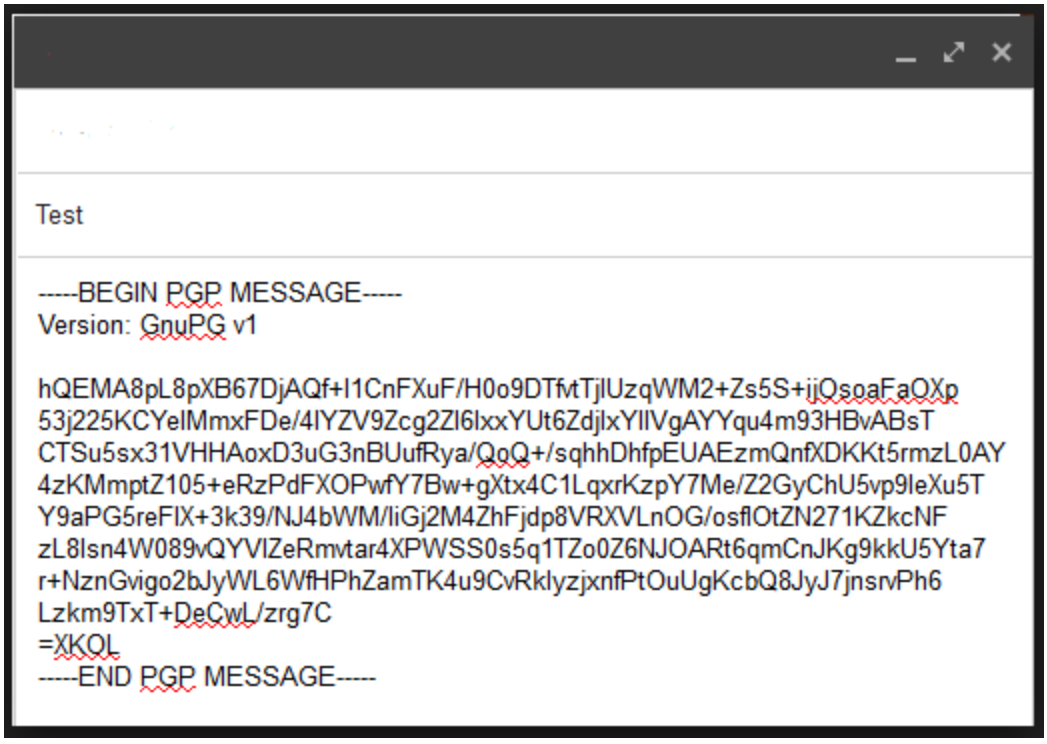
The primary reason for selecting steganography among the list of possible project topics was due to the unfamiliarity of the word that twigged an interest in the subject.

Another motivation was after reading an article “Veiled Messages of Terror May Lurk in Cyberspace”… “The idea of steganography is to take advantage of the fact that digital files, like photographs or music files can be slightly altered and still look the same to the human eye or sound the same to the human ear.

“The only way to spot such an alteration is with computer programs that can notice statistical deviations from the expected patterns of data in the image or music. Those who are starting to look for such deviations say that their programs are as yet imperfect but that, nonetheless, some are finding widespread use of steganography on the Internet. For National Security reasons some of these experts do not want to reveal exactly what they find, and where…. Quite an alarming number of image appear to have steganography in them…” (said one expert who has looked for them, Chet Hosmer, the president and chief executive of WetStone Technologies in Cortland, N.Y)

Introduction

History of steganography dates back to 440 B.C. Steganography derives from the Greek word, “Steganos” and was in Greece, where the first document occurrence of steganography took place, people used to write on wax-covered tablets. The word steganography is considered similar to cryptography or watermarking.
The main objective of steganography is to avoid drawing attention to the transmission of hidden information. Steganography relies on hiding covert message in unsuspected multimedia data and is generally used in secret communication between acknowledged parties.
In case of confidential information that should be sent, provision of security is an inevitable issue which cannot have any compromise.



The message above is a sentence in English that is encrypted using Pretty Good Privacy (PGP), most likely the most commonly used e-mail encryption software today. The advantage of steganography over cryptology is that message does not attract attention to themselves, to recipes, or messengers. Whereas the goal of cryptography is to make data unreadable by a third party, the goal of steganography is to hide from a third party.

Technology Used

- FFMPEG (with the .exe extension for Microsoft Windows).
- Microsoft Visual Studio 2013.
- C# is an objective oriented programming language developed by Microsoft as part of the .NET, which was later approved as a standard by The International Organization for Standardization (ISO) and The European Computer Manufacturers Association (ECMA). Anders Hejlsberg leads as developer of the C# language.
- In this project all the framework has been designed in C#

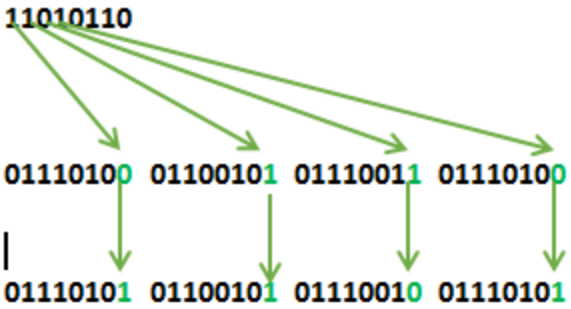
Steganographic Techniques

- **Substitution**
- **Transform**
- **Domain**
- **Generation**
- **Injection**

Substitution

The file size of the container object is preserved. It modifies pre-existing data of the container file and limits the steganographic capacity of the container file.

Least Significant Bit (LSB) manipulation



Transform Domain

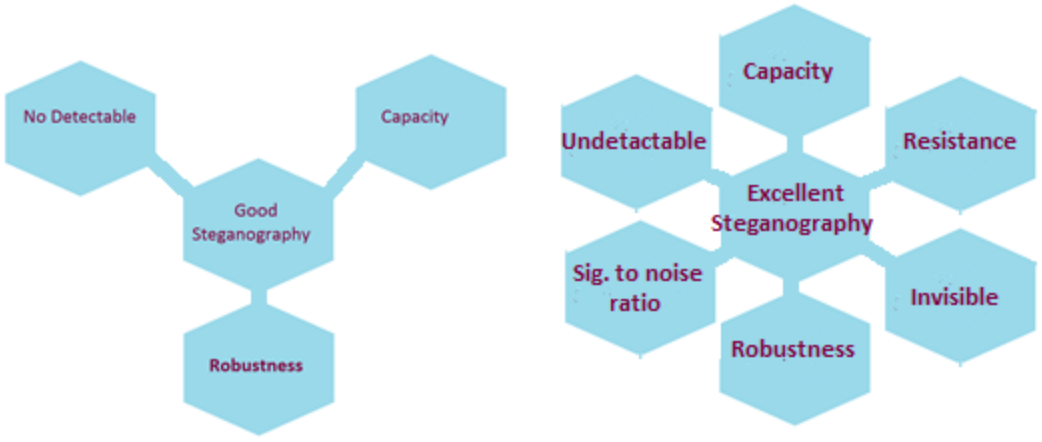
Discrete Fourier transforms (DFT).
Discrete wavelet transforms (DWT). -JPEG 2000
Discrete Cosine transforms (DCT). -JPEG

Generation

There is no original container file. It generates a container file based on the covert data.

Injection

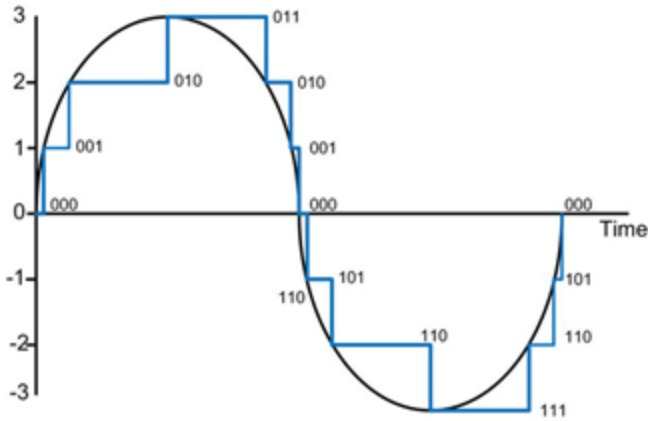
Files suitable for injection techniques: – EXE files – WAV files
Inject data into redundant parts of a file.
Injection is generally deemed to be less secure than substitution or generation techniques.

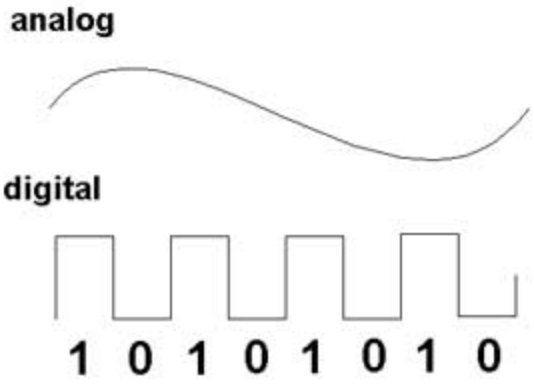


Properties of good steganography

Analysis of Digital Audio

Digital audio is stored on a computer as a sequence of 0's and 1's. With the sufficient tools, it is possible to change the individual bits that make up a digital audio file. The secret message is embedded by slightly altering the binary sequence of sound file.
The figure illustrates a contiguous analog wave being sampled to produce digital audio.

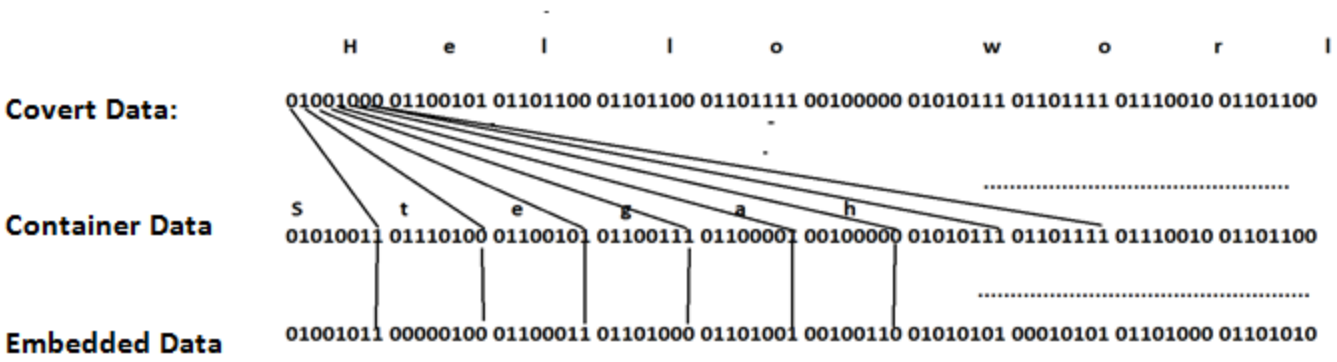




The cover of an ordinary sound note or message, that might look inoffensive, can hide plenty of random information and be covered by what is called “white noise”.

The article “How Guitars Work” talks about the difference between noise, tones and notes. White noise is a type of noise that is produced by combining sounds of all different frequencies together. Because white noise contains all frequencies, it is frequently used to mask other sounds.

We can hear white noise as a nearly silent hiss of a blank tape playing. Thus the basic design principle of the stenographic system is to replace high entropy noise with a high entropy secret transmission.



This figure shows “Hello World” inside container data.

We can apply a common steganographic technique where we can manipulate the least significant bit (LSB). This can be applied to audio formats and works by modifying a portion of the data stored without causing any audible difference.

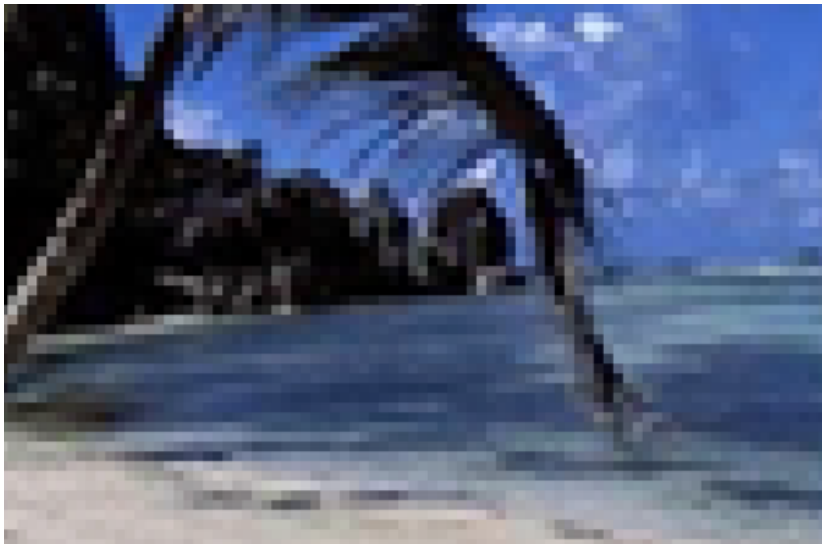
Analysis of Digital Image

The human visual system as a functional unit including the eyes, the nervous system, and the corresponding parts of the brain certainly ranks among the most important means of human information processing. However, there are areas of application where digital image analysis system produce incredible data information about an image.
An image file is a binary file containing representation of the color or light intensity of each picture element (pixel) comprising the image. Images usually use either 8 bit or 24 bit color.

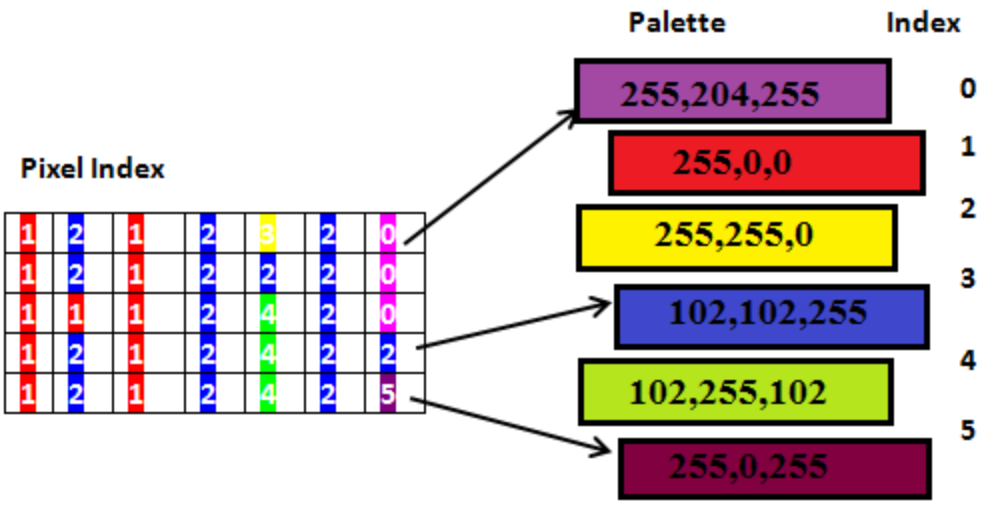
No Message

Attack at midnight

When using 8 bit color, there is a definition of up to 256 colors forming a palette for this image, each color denoted by an 8 bit value.
When using 24 bit color, there will be 24 bits per pixel and allows a better set of colors. Each pixel is represented by three bytes, each byte representing the intensity of the three primary colors, green, red and blue.

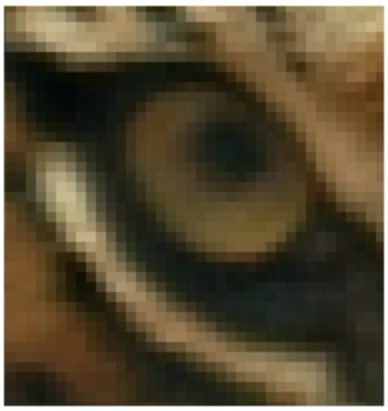


As we can see in the above picture the size of the image is directly related to the number of pixels and the granularity of the color definition. Usually 100x100 px image using require a file of 29.3 KB, whereas a 1024x768 pixel image with high resolution 24 bit color would have 2.36 MB file(1024x768x3 bytes).

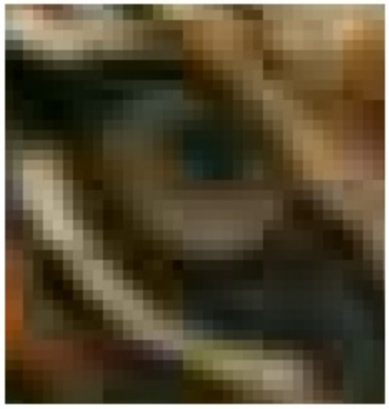


File size = 6x3(palette) + 7x5x1 bytes (indices) = 53 bytes total
Image File = Palette + Indices

File with enormous size can be compressed. For instance, files of type GIF and 8 bit BMP uses a scheme that allows the software to reconstruct the original image, this scheme is called lossless compression. On the other hand JPEG, uses **Lossy Compression**. By doing this the expanded image is very similar to the original but not an exact duplicate.



original image



lossy JPEG format
with "artifacts"

We can conclude that lossless compression is much better for applications where integrity of the original information must be maintained, such as steganography.
The approach to hiding data within an image file called **Least Significant Bit** (LSB). This is a way to take binary representation of the hidden data. Here we can use 24 bit color, the change will be minimal. Here we have an example where we have three adjacent pixels (10 bytes)

100101010 000011010 110010011
100101101 110010100 000011110
110010110 100111110 000100001

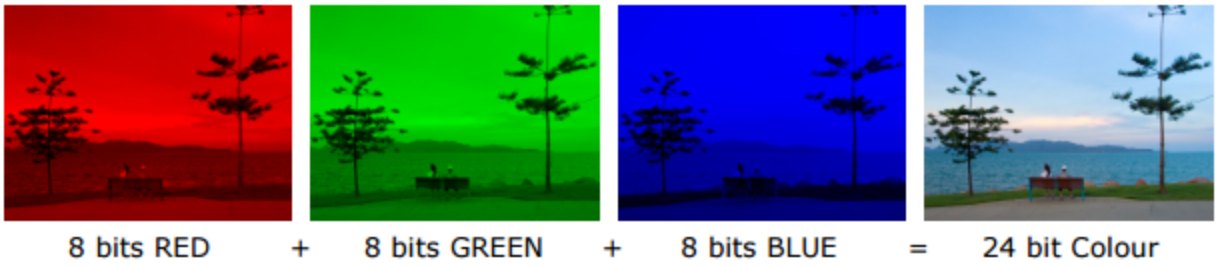
If we want to hide these 10 bits of data, by that we mean that the hidden data is usually compressed prior to being hidden. If we overlay these 10 bits over the LSB, we will get the following result.

100101010 000011010 110010011
100101101 110010100 000011110
110010110 100111110 000100001

We have hidden 10 bits but at a cost of only changing 5, of the LSBs.

This project involves the following image file formats:

Join Photographic Expert Group	(.jpeg)	This file format supports 8 bits for each red, green and blue and 8 bit grayscale image, but the file is affected by generational degradation when repeatedly saved and edited.



Portable Network Graphics	(.png)	Limited to an 8 bit palette with optional transparency for all palette colors and 48 bit truecolor and without alpha channel.
Graphics Interchange Format	(.gif)	Limited to an 8 bit palette or 256 colors

Analysis of Video

Modifications of a video file can be more difficult to detect, because of the brief periods of time frames are displayed on screen. In this case the variations in pixel color induced by steganography will blend into frame.
Due to the complex nature of audio-video, the range of audio-video manipulation libraries is very limited. Most of the manipulations take place under libraries that act such as wrappers for FFmpeg. The data structure of FFmpeg API is an AVFrame, which provides access to low level aspects of encoding.
Video steganography can be very useful for hackers as a relatively new method for exfiltrating hacked data past the more conventional security tools, easily bypassing typical intrusion detection/prevention systems. They do so by hiding the data in a video and then uploading through a trusted or open-source video sharing platform.
Demonstrating the effectiveness of this new form of attack, a Fortune 500 company was hit through a video steganography attack, with sensitive data being compromised and the attacked undetected

until they were alerted duplicate videos were being uploaded from their network to a video sharing website.

Custom Code

We knew this project would require a custom coded program to accomplish what we needed to do, which was:

- Select a file to encrypt.
- Apply the encryption algorithm to the selected file.
- Record an audio and video file, or select an existing one.
- Place the encrypted file into the audio of the video file.
- Merge the modified audio track with the video track into one output file.
- Extract and decrypt a file from a video file.
- Develop a simple user interface that makes it easy to run and use this program.

Once analyzing the requirements was done, we decided this would be most easily done using a program called ffmpeg to manipulate and capture audio and video files. To develop the scripts and user interface, we chose the C# programming language because it was familiar to all members and made implementation of these requirements pretty simple.

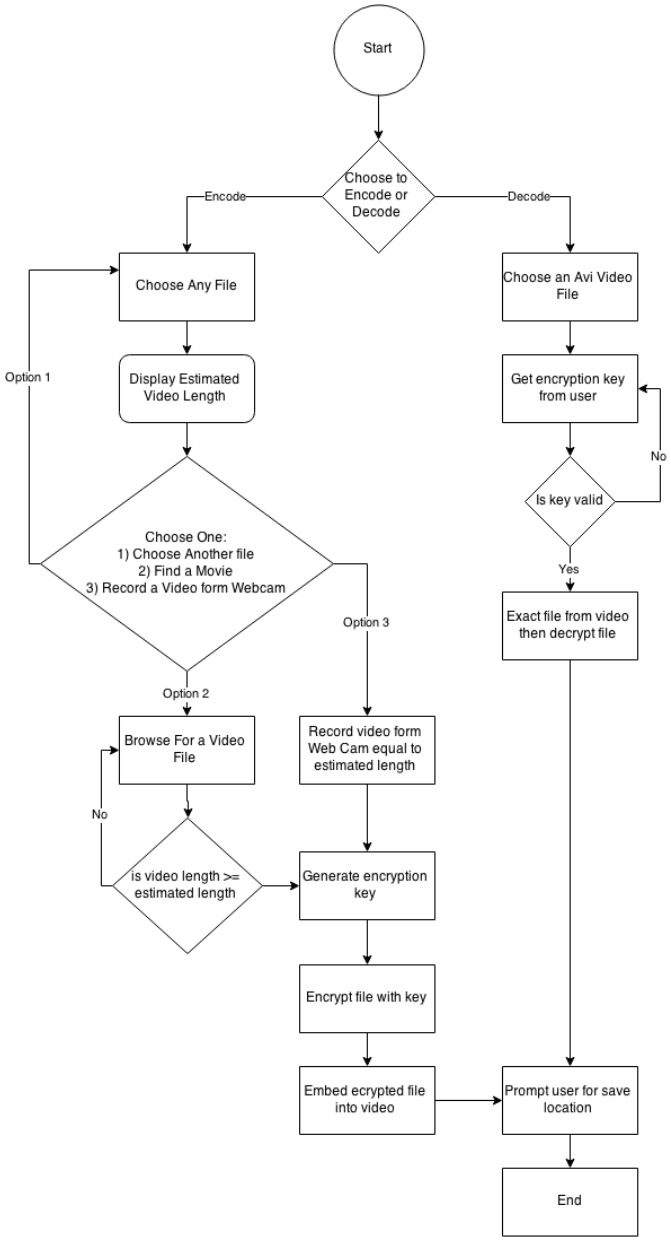
Now that we knew what we needed to do, and how we were going to do it, we split up the work and came up with a design plan for the major components: User Interface, the Encryption Algorithm, and Audio/Video Capture.

User Interface:

We wanted to keep this part relatively simple, since it wasn't to be the main focus of the project, yet we wanted to make sure it was still user friendly. We just used C# forms developed in Visual Studio, with a simple mixture of comboboxes and buttons.
The forms we needed to run this program was:

- Main form: Select whether how to get the video we were going to store the encrypted file in. Whether it be by recording one via webcam, or choosing an existing video file.
- Key form: Enter the key text for the key used in encryption.
- Webcam form: Select a webcam to use to record the video.
- Spinner: Shows current progress of the program running.

The flowchart for how the user interface would interact is pictured here:



Encryption Algorithm:

For this part of the code, our design was to go with the Rijndael Algorithm, also known as AES(Advanced Encryption Standard). This algorithm is characterized by:

- Key sizes of either 128, 192, or 256 bits.
- Block size of 128 bits.
- Number of rounds: 10, 12 or 14 based on key size.

```
namespace SteganographySolution.Common
{
        public static class Encryption
        {
                private static Random rnd;
                private static SymmetricAlgorithm newEncryptionAlgorithm()
                {
                        return new RijndaelManaged();
                }
                private static SymmetricAlgorithm newEncryptionAlgorithm(byte[] keyData)
                {
                        if (keyData == null) throw new ArgumentNullException("keyData");
                        if (keyData.Length != SymmetricKeyLength) throw new ArgumentException("Invalid ke

                        var algorithm = new RijndaelManaged();

                        byte[]
                                key = new byte[keyLength],
                                iv = new byte[ivLength];

                        Buffer.BlockCopy(keyData, 0, iv, 0, iv.Length);
                        Buffer.BlockCopy(keyData, iv.Length, key, 0, key.Length);
                        algorithm.Key = key;
                        algorithm.IV = iv;
                        algorithm.Padding = PaddingMode.PKCS7;

                        return algorithm;
                }
```

The picture above shows us creating a new key, using the keydata fed into the function, and initialization vector while returning the encryption algorithm. This is just part of the encryption class, followed by these functions that do the encrypting and decrypting of blobs or streams.

```
public static Task DecryptStream(this Stream stream, Stream saveToStream, byte[] keyData)
{
        if (saveToStream == null) throw new ArgumentNullException("saveToStream");
        if (stream == null) throw new ArgumentNullException("stream");

        return Task.Factory.StartNew(() =>
        {
                using (var algorithm = newEncryptionAlgorithm(keyData))
                using (var decryptor = algorithm.CreateDecryptor())
                using (var crypto = new CryptoStream(stream, decryptor, CryptoStreamMode.Read))
                        crypto.CopyTo(saveToStream);
        });

        //return stream.CopyToAsync(saveToStream);
}
/// <summary>
/// Encrypts a blob or stream.
/// </summary>
/// <param name="stream">The source stream.</param>
/// <param name="saveToStream">The export stream.</param>
/// <param name="keyData">The key to encrypt the source stream.</param>
/// <returns></returns>
public static Task EncryptStream(this Stream stream, Stream saveToStream, byte[] keyData)
{
        if (saveToStream == null) throw new ArgumentNullException("saveToStream");
        if (stream == null) throw new ArgumentNullException("stream");

        return Task.Factory.StartNew(() =>
        {
                using (var algorithm = newEncryptionAlgorithm(keyData))
                using (var encryptor = algorithm.CreateEncryptor())
                using (var crypto = new CryptoStream(saveToStream, encryptor, CryptoStreamMode.Write))
                        stream.CopyTo(crypto);
        });
        //return stream.CopyToAsync(saveToStream);
}
```

We chose this algorithm just because of how strong it is and how resistant it is to decryption, which while possible, is still very, very far from being practical. We also chose this algorithm because of ease of access, it easy to use and has been thoroughly tested against, the algorithm alone makes the data very secure and the steganography just adds another layer.
We had to write the code for key generation and manipulation, and set the initialization vector. Once those were we could easily call the algorithm using a library.

Audio and Video Capture:

A vital part of the program was to be able to record a video that was longer or equal to the estimated length needed to encrypt the file. Ffmpeg is a command line executable program that allows you to record, convert, and stream audio and video files. This program was perfect for our project, and its well-documented and easy to use command lines made it very simple to write scripts that were easily called and ran throughout our program. Some scripts we used ffmpeg for included:

- Finding all webcam sources on the computer so they can select one to use(Pic 1, *below*).
- Record the video using a selected webcam(Pic 2, *below*).
- Merge the audio and video back together after the encryption takes place(Pic 3, *below*).

```
{
        using (var ffmpeg = new Process())
        {
                //The parameters for the program to start
                ffmpeg.StartInfo.Arguments = "-list_devices true -f dshow -i dummy";
                ffmpeg.StartInfo.CreateNoWindow = true;
                ffmpeg.StartInfo.FileName = "ffmpeg";
                ffmpeg.StartInfo.UseShellExecute = false;
                ffmpeg.StartInfo.RedirectStandardError = true;
                ffmpeg.StartInfo.RedirectStandardInput = true;
                ffmpeg.StartInfo.RedirectStandardOutput = true;
                ffmpeg.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;

                //Start the program
                ffmpeg.Start();

                //Find the line "DirectShow video devices"
                string line;
                while (!(ffmpeg.StandardError.ReadLine() ?? "").Contains("DirectShow video devices

                //Read Web Cams
                while (!(line = ffmpeg.StandardError.ReadLine() ?? "").Contains("DirectShow audio
                        if (!line.Contains("Alternative name \"@"))
                                webCams.Add(line.Substring(line.IndexOf('"')).Trim('"'));

                //Read Audio Devices
                while (!(line = ffmpeg.StandardError.ReadLine() ?? "").Contains("dummy: Immediate
                        if (!line.Contains("Alternative name \"@"))
                                audioDevices.Add(line.Substring(line.IndexOf('"')).Trim('"'));

                //Wait for the program to end
                ffmpeg.WaitForExit(5000);

                //If the program has not ended by now, kill the process
                if (!ffmpeg.HasExited) ffmpeg.Kill();
        }

```

Simple script to find all the media devices on the computer.

```
public static async Task RecordVideo(string deviceName, string audioDevice, TimeSpan videoLength, string outputLoc
{
        await Task.Factory.StartNew(() =>
        {
                using (var ffmpeg = new Process())
                {
                        //The parameters for the program to start
                        ffmpeg.StartInfo.Arguments = String.Format("-f dshow -i video=\"{0}\":audio=\"{1}\" \"{2}\
                                deviceName, audioDevice, outputLocation);
                        ffmpeg.StartInfo.CreateNoWindow = true;
                        ffmpeg.StartInfo.FileName = "ffmpeg";
                        ffmpeg.StartInfo.UseShellExecute = false;
                        ffmpeg.StartInfo.RedirectStandardError = true;
                        ffmpeg.StartInfo.RedirectStandardInput = true;
                        ffmpeg.StartInfo.RedirectStandardOutput = true;
                        ffmpeg.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;

                        //Start the program
                        ffmpeg.Start();

                        //Start a timer for keeping track of how long things have happened
                        var timer = new Stopwatch();
                        timer.Start();

                        //Wait for the program to startup
                        while (timer.Elapsed < TimeSpan.FromSeconds(2))
                                Console.WriteLine(ffmpeg.StandardError.ReadLine());
                        timer.Restart();

                        //Read the output of the program while it records the video
                        while (timer.Elapsed < videoLength)
                                Console.WriteLine(ffmpeg.StandardError.ReadLine());

                        //Send the command to the program to end the recording
                        ffmpeg.StandardInput.Write("q");

                        //Wait for the program to end
```

Simple script to record a video of required length for file encryption.

```
namespace SteganographySolution.Common
{
        public static class MergeMedia
        {
                /// <summary>
                /// Use this method to replace the audio track for a video with an audio flac file.
                /// </summary>
                /// <param name="videoFile">The target video file.</param>
                /// <param name="flacAudioFile">The source audio track.</param>
                /// <param name="output">The save to location for the new video file.</param>
                public static Task ReplaceAudioTrackWithFlacFile(this FileInfo videoFile, FileInfo flacAudioFile, string output)
                {
                        return Task.Factory.StartNew(() =>
                        {
                                using (var ffmpeg = new Process())
                                {
                                        ffmpeg.StartInfo.Arguments = String.Format(
                                                "-i \"{0}\" -i \"{1}\" -c:v copy -c:a flac -strict experimental -map 0:v:0 -map 1:
                                                videoFile.FullName, flacAudioFile.FullName, output);
                                        ffmpeg.StartInfo.FileName = "ffmpeg";
                                        ffmpeg.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;

                                        ffmpeg.Start();
                                        ffmpeg.WaitForExit();
                                }
                        });
                }
        }
}
```

Simple script above uses ffmpeg as a one line command to merge audio and video.

Embedding the File into the Audio Track

The algorithm for embedding the file into the audio track is (blob is the file byte array, source is the mono audio array):

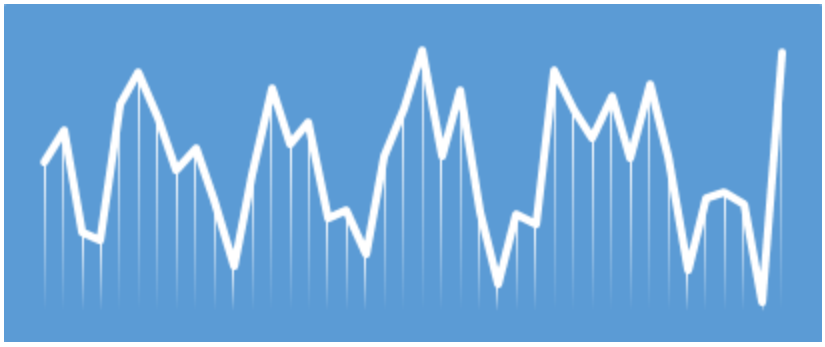
Left Channel:
Min(128, blob[i]) - 96 + source[i]

Right Channel:
Max(blob[i] - 128, 0) + source[i]

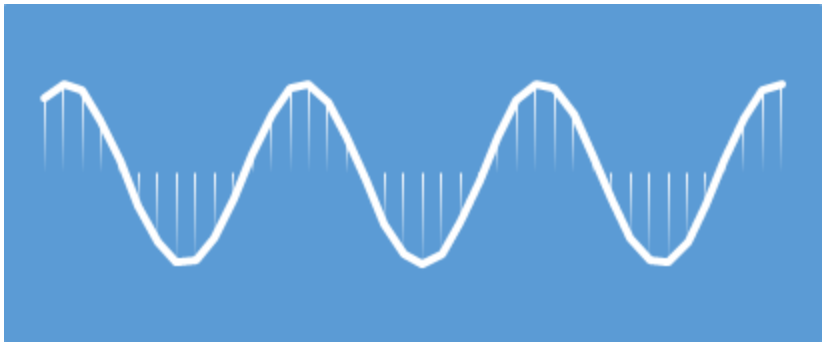
The inverse of these functions are:
left[i] - right[i] + 96

You can see in the examples below how it works (assuming that source is a signed 16 bit audio array)

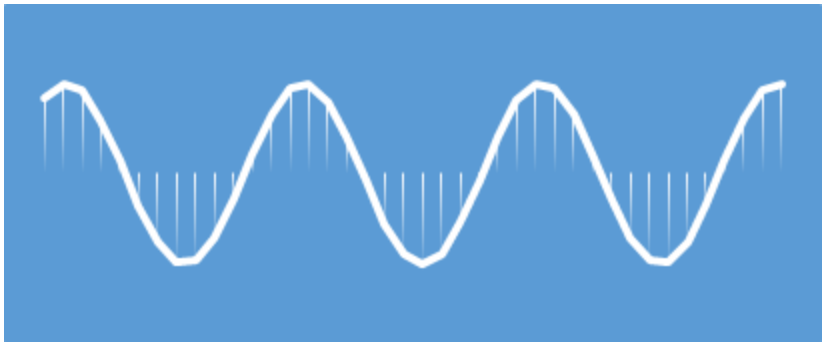
“blob” array:



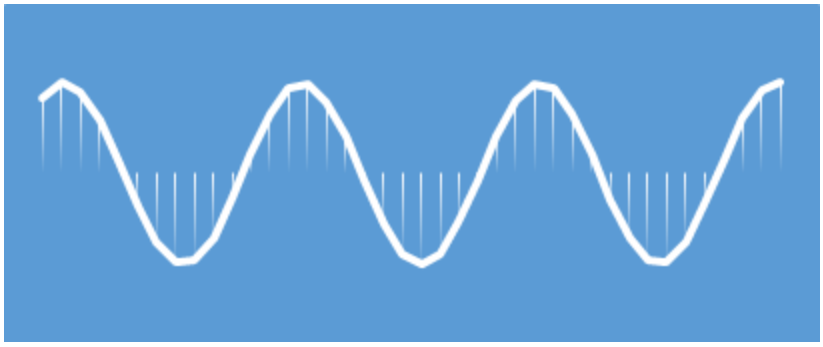
“source” audio array:



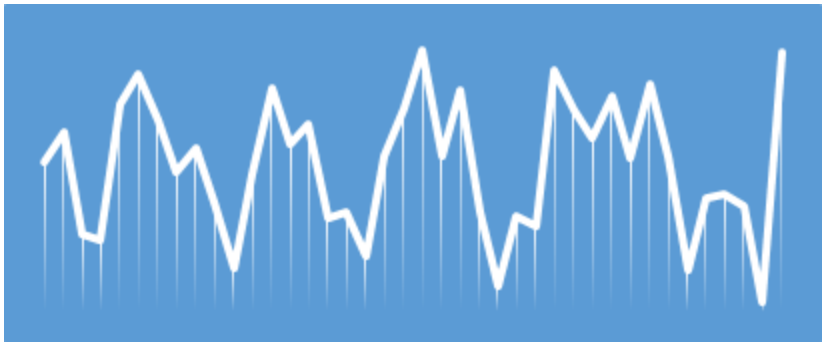
Results:
left channel:



right channel:



The file after it was extracted from the audio track:



It is also important to note that the file is encrypted before it is embedded into the audio track. This allows any given file to sound like soft white noise in the background.

Conclusion

Steganography is not intended to replace cryptology, but both together could make a good combination. If we need to encrypt and hide a message with a steganographic technique, it provides an additional layer of protection. “We never know if a message is hidden”, this is the dilemma that empowers steganography. It is a very discrete way to transmit information if done correctly, and could have a big impact in the the protection and transfer of confidential information.

Steganography is still a new concept to the general public, although this is not likely on the areas of privacy and surveillance. We believe that effort to improve the robustness of steganography techniques will help law enforcement better detect illicit material transmitted through the internet, but that is no easy task.

Glossary

Covert channel:” It is a type of computer security attack that creates a capability to transfer information objects between processes that are not supposed to be allowed to communicate by the computer security policy.”

Data hiding:” It is a characteristic of object-oriented programming. Because an object can only be associated with data in predefined classes or templates, the object can only "know" about the data it needs to know about.”

Data security:” It means protecting data, such as a database, from destructive forces, and from the unwanted actions of unauthorized users.”

Steganography:”A means of overlaying one set of information ("message") on another (a cover).**”**

Watermarking:”A watermark is an identifying image or pattern in paper that appears as various shades of lightness/darkness when viewed by transmitted light (or when viewed by reflected light, atop a dark background), caused by thickness or density variations in the paper**”**

Hidden channel:”Hidden channels (covert channels) are communication channels that transmit information without the authorization or knowledge of the channel’s designer, owner, or operator.**”**

References and Bibliography

1. Kolata, Gina. "Veiled Messages of Terror May Lurk in Cyberspace." *The New York Times*. The New York Times, 29 Oct. 2001. Web. 04 Apr. 2015.
2. How Acoustic Guitars Work - HowStuffWorks." *HowStuffWorks*. N.p., n.d. Web. 04 Apr. 2015.
3. *Dictionary.com*. Dictionary.com, n.d. Web. 04 Apr. 2015.
4. "Welcome to Steganosaurus." *Video Steganography*. N.p., n.d. Web. 04 Apr. 2015.
5. "Hiding Information Perfectly - Historical Overview." *Hiding Information Perfectly - Historical Overview*. N.p., n.d. Web. 04 Apr. 2015.