**Project Proposal**

**Steganographic Encoding Program**

**Message Encryption in Audio Group**

Savannah Pate - U23444733

Denise Nguyen - U89130578

Tom Nguyen - U63542702

Vivien Nguyen - U21147681

Bradley Fritz - U07645545

C. One paragraph description of the project

We will work on steganography encoding program for our project. Steganography is the practice of hiding data and information within other data that is shown in plain sight. This data can range from text, image, to video/audio files. We will narrow down our focus on hiding text inside of a WAV audio file. Our program will support two processes, embedding and extraction; it accepts two inputs, a WAV audio file, and a message. The text in ascii uses 8 bits which takes up a great amount of memory. To increase encoding capacity, it is necessary to compress the text from 8 bits to 5 bits. To encode the audio file with text, the least significant bit of the audio data is replaced with bits from the text data. To extract the text, the program executes the above processes in reverse.

D. Major project milestones, and proposed timeline for completion

* Project proposal due 9/20
* Research day/outline code 10/2
* Code: Compression and decompression of text 10/13
* Code: Embedding text within audio 10/27
* Code: Extracting text from audio 11/10
* Code: Revision, Adjustments, Additions 11/17
* Project Due 11/27

E. Description of the deliverables/expected results.

The deliverables of this project will be high-level language program file(s) to create a steganographic WAV audio file from the original. We chose to work with the WAV format because it is an uncompressed format, that is fairly easy to process and edit. The expected result is that the program will take an audio file, manipulate that audio file to insert a “hidden message”, and output a visually indistinguishable, manipulated version of that audio file. We also expect to be able to view the message in plain text. This will be done by separating the header from the data of the audio, in order to preserve the header. To encode the audio file with text, the least significant bit of the audio data is replaced with bits from the text data.

F. Collaboration summary (who is responsible for what components)

Proposal - all members

Research - all members

Code:

Main function - Vivien Nguyen

Compression - Denise Nguyen

Encode function- Bradley Fritz

Decode function- Savannah Pate

Decompression - Tom Nguyen

Report - all members

G. Possible problems you may encounter, and potential solutions

Certain formats of the audio file must be decided beforehand. Users will need to convert their audio file to the required format before encoding. The original encrypter may need to know if their encryption is correct. We can solve this issue by including a program to check their message. The size of WAV files are larger than mp3 formats, so the length of the file may have a noticeable effect on execution time. For demonstration purposes, it may be best to use a small WAV file to reduce the number of instructions, therefore reduce the execution time.

<https://www.daniweb.com/programming/software-development/threads/340334/reading-audio-file-in-c>

<https://www.youtube.com/watch?v=7MigwaAjLKs>

<http://truelogic.org/wordpress/2015/09/04/parsing-a-wav-file-in-c/>

NOTES:

<https://github.com/samolds/wavsteg> Github with entire MIPS and C implementation

Main idea: Storing text message contents into LSB of each sample in the WAV file.

WAV file contains header and data.

1. Header: 44 bytes long

8 bits in 1 byte

2) Data: contains individual samples

Samples = bit size \* number of channels

Each sample is 16 bits in our example

Encode:

1. Take in text (data) file
2. Take in wav file

* Reads 1 byte at a time from text file. (**fgetc**)
  + Take every bit of byte (8 bits in 1 byte)
* For each iteration through the wav file
* (iterating through each 16 bit sample of wav file):
* Skip 8 bits (1 byte) (GSB) and read the next 8 bits (LSB)