Steganography on VSQx File Format

Muhammad Iqbal Sigid - 13519152
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: sigid.iqbal123@gmail.com

***Abstract*—The Vocaloid Sequence (VSQx) file format is a proprietary file format used by the VOCALOID software for music production and voice synthesis. As it is similar to an audio file that stores audio data in sequence, VSQx can be used as a cover file in steganography. We provide two alternative algorithms to store a secret message within the VSQx file, one of which adds a new unrelated property and the other modifies and existing property inside the VSQx file. The former algorithm is more robust and has smaller MSE than the latter but is more perceptible as a trade-off. The algorithms provided are still basic and open for improvements and explorations, such as adding cryptography, using another encoding technique, and other method to increase its capacity.**

Keywords—Steganography; Audio; Vocaloid Sequence

I. INTRODUCTION

Steganography is the art and science of hiding information within a cover medium in such a way that it remains undetectable to an observer. This technique has been used in communication systems to transmit a secret message without being detected. In digital media, images, videos, and audio files have been used as a cover to send hidden messages.

The Vocaloid Sequence (VSQx) file is a proprietary audio format used by the VOCALOID software for music production and voice synthesis. It contains information about the timing, lyrics, pitch, notes of a song sequentially, which makes it similar to an audio file. As audio files can be used as a cover object for steganography, it is also possible to use a VSQx file as a cover object.

This paper explores the feasibility of using VSQx file in steganography. We first provide a brief overview of steganography and its applications, then discuss the VSQx file format and its structure, highlighting its potential for steganography.

II. FOUNDATIONAL CONCEPT

A. Steganography

Steganography derives from the Greek word *steganos*, meaning covered or secret, and *graphy* (writing or drawing). Steganography is a practice of hiding information within another message, such that the presence of the information is not evident. An example of conventional steganography is hiding writing such as using invisible ink. In computing, a message, image, video, or other files are hidden within another file. Therefore, steganography consists of two properties, which are the secret message and the cover message.

There are a couple of criteria that needs to be considered on steganography:

- Imperceptibility, the secret message is not easily perceived.
- Fidelity, the cover message stays mostly the same as before the secret message is inserted.
- Robustness, the secret message is resistant to modification.
- Recovery, the secret message must be able to be recovered.
- Capacity, related to the amount of information that can be stored by the cover message.

Improvement on one factor can cause a trade-off on the other factors. For example, a steganography technique that can hide information very well may not be able to hold much information, however a less secure one may be able to hold more information.

B. VOCALOID and VSQx File Format

VOCALOID is a singing voice synthesizer software developed by Yamaha. It enables the users to synthesize singing by inputting lyrics and melody. It has a piano roll type interface for the user to input the melody and lyrics into each note. The software can also customize the singing by adding effects such as vibrato or changing the dynamics and tone of the voice. The sequences of notes along with its various properties that are created by the software are stored on a file with the VSQx file format.

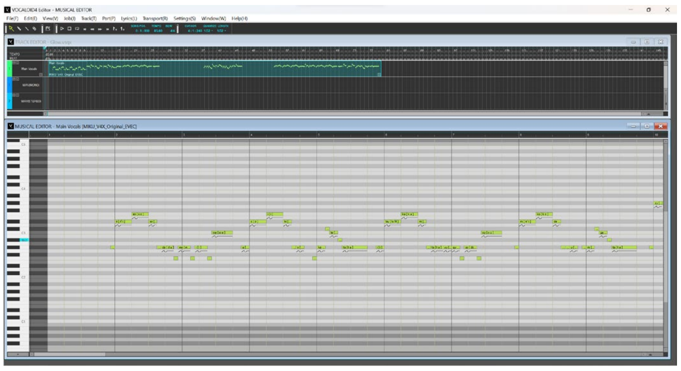


Figure 1. The VOCALOID4 software

The VSQx file format is an updated format of the VSQ file format, which is used to store musical notes and vocal data for the VOCALOID software. VSQ is used for the older VOCALOID2, which was then replaced by the VSQx on the newer VOCALOID3 and VOCALOID4 software. VSQx stores more information than its predecessor, but it's still relatively the same as VOCALOID4 as it can read the older VSQ format. The difference is VSQx uses XML format which makes it easier to process with other applications, compared to the unique VSQ format.

The VSQx file format contains a couple sections that are explained on Table 1.

Table 1 Sections on the VSQx file format

No	Section Name	
1	vVoiceTable and vVoice	Contains the used voicebank data. • Id, the id of the voicebank • name, voicebank name. E.g., Miku V4X Original
2	mixer	Contains information about the synthesizer mixer and its attributes.
3	masterTrack	Contains information about the master track, such as time signature and tempo.
4	vsTrack and vsPart	Contains information about each track in the voice sequence.
4.1.	name	Track's name
4.2.	playTime	Track's duration
4.3.	pStyle	Contains the following information: • accent, accent value of the note. • bendDep, how much the pitch-bend of the current note. • bendLen, how long the pitch-bend of the current note. • decay, audio decay • opening, variation of pronunciation through adjustments of the opening of the mouth. • fallPort, fall portamento value. • risePort, rise portamento value.
4.4.	note	Contains information for each note in the track. • t, the timestamp which the note begins. • dur, the duration of the note • n, the note (ex. G#2) • y, the lyrics on the note • p, the phonetic on the note • nStyle, contains the same information as pStyle but for each note.

Most of the important informations in VSQx files are located in the vsTrack section as it contains all the notes that makes up the music.

C. Peak to Signal Noise Ratio

Peak to Signal Noise Ratio (PSNR) is a metric for comparison between the maximum value (power) of a signal and the power of distorting noise that affect the fidelity of its representation. This can be a useful metric to approximate the effect of distortion to the original signal or file. PSNR is usually used to quantify compression methods on an image and videos but can also be used on audio files. The formula to calculate PSNR is as follows.

$$PSNR = 20 \times \log_{10}\left(\frac{MAX}{rms}\right)$$

The MAX represents the maximum value of the signal. This value will be different depending on the data representation. In image, this is equal to 255 since an image uses 8 bits of data to store a pixel. A wav file uses 16 bits, so the MAX value will be 65535. RMS is root mean square error, which represents the difference between values of the original file and the modified file. The formula for RMS is as follows.

$$rms = \sqrt{\frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M}\left(I_{ij} - \hat{I}_{ij}\right)^2}$$

PSNR is represented using decibel (dB) and is in reverse with RMS. A low rms will result in high PSNR, meaning that the difference between the two files is low and vice versa. A good PSNR value will depend on its MAX value. For 16-bit data, the typical values for PSNR are around 60 dB to 80 dB[1].

III. METODOLOGY

To hide information inside the VSQx file format, the secret message needs to be imperceptible or hard to find by the naked eye. Taking the inspiration of the LSB steganography on an image, where the secret message is broken up into parts and each pixel stores a part of the secret message, we can implement a similar algorithm. However, the difference is a pixel on an image only stores one byte for grayscale image or 3 bytes for RGB image, whereas each note on a VSQx contains much more information. This makes information storing on VSQx more flexible.

A. *Basic Steganography Design*

Since there are several sub-properties inside the note property, we need to choose on how to add additional information without changing the data such that it becomes perceivable by a human or becomes unreadable by the VOCALOID software. The lyrics and the phonetics are stored as a string with its own format, which means modifying them will cause the file to be unreadable or causes a notable change to the note. The same applies to the note, which is stored as an integer and even a slight change can cause a perceivable change by the human ear. A slight change to the time and duration of a note may be unnoticed by some, but the safer option is to change the nStyle.

The nStyle property contains several sub-properties, but instead of modifying one or more property, we choose to add another sub-property separate to those that are used by the VOCALOID software. This was chosen due to its simplicity, and it doesn't change the information in the note. Adding another property to the note property cannot be done as it causes the file to be unreadable, while adding a property to the nStyle does not.

Since each note will be used to hide a part of the secret message, the secret message must be divided into parts. Another thing that needs to be considered is how the parts of data will be represented on each note and how it can be reconstructed. A simple method is to store one byte on each note, but there is a downside as larger data will require a lot of notes. Each byte stored on a note will be represented as an integer as other property values are also integers.

B. *Implementation*

Implementation of the steganography program will be using Python programming language with ElementTree library for parsing XML files. The steganography process is illustrated in Figure 2.

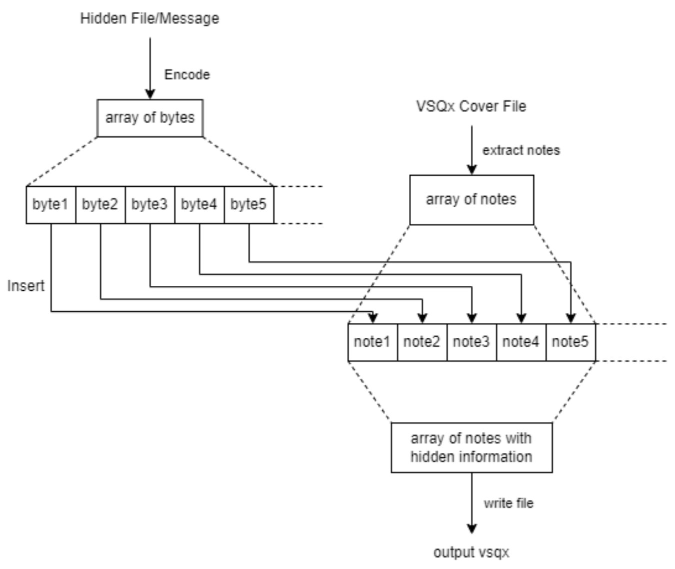


Figure 2. Hidden message insertion process

The program takes two inputs, the VSQx cover file and the message or information that we want to hide. For the sake of simplicity, the implementation will only take a string as an input as one note only stores one byte of information. The program will then encode the message, in this case using the UTF-8 encoding, and separate each byte. Using the ElementTree library, the notes inside the VSQx file are extracted by its XML tag and attributes. Iterating each note, the array of bytes will then be inserted into the nStyle property of the note. After the last byte is inserted, an extra character is added as an end flag. The extra character that is used is '1000.' The modified VSQx file is then written into an output file.

The result of the modified note section is shown in Table 2. The insertion added one extra line with a tag 'v' and id 'stego' that has a value of 72. The number 72 here represents one of the bytes of the message. The id name "stego" is chosen for easier debugging during development. In practice, it should be named something that is similar to the other attributes to make it less obvious.

Table 2. Result of modification on one of note section

Original Note Section

```
<note>
   <t>1800</t>
   <dur>120</dur>
   <n>56</n>
   <v>64</v>
   <y><![CDATA[fu]]></y>
   <p><![CDATA[p\ M]]></p>
   <nStyle>
      <v id="accent">50</v>
      <v id="bendDep">8</v>
      <v id="bendLen">0</v>
      <v id="decay">50</v>
      <v id="fallPort">0</v>
      <v id="opening">127</v>
      <v id="risePort">0</v>
      <v id="vibLen">0</v>
```

```
        <v id="vibType">0</v>
    </nStyle>
</note>
```

Modified note section

```
<note>
    <t>1800</t>
    <dur>120</dur>
    <n>56</n>
    <v>64</v>
    <y><![CDATA[fu]]></y>
    <p><![CDATA[p\ M]]></p>
    <nStyle>
        <v id="accent">50</v>
        <v id="bendDep">8</v>
        <v id="bendLen">0</v>
        <v id="decay">50</v>
        <v id="fallPort">0</v>
        <v id="opening">127</v>
        <v id="risePort">0</v>
        <v id="vibLen">0</v>
        <v id="vibType">0</v>
        <v id="stego">72</v>
    </nStyle>
</note>
```

The hidden message extraction process is simpler as illustrated in Figure 3. Using the ElementTree library, each note and its hidden message is extracted by using their tag and attribute. The extracted information is then combined sequentially until the end flag and the original message can be read.

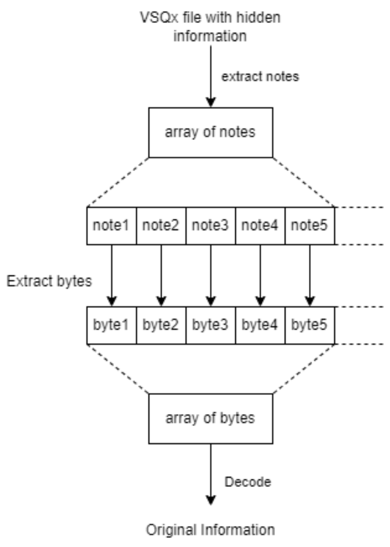


Figure 3. Hidden message extraction process

C. *Alternative Method*

Rather than creating a new sub-preporty inside the nStyle property, we can also choose a certain sub-property that doesn't majorly affect the note. Based on the testing that was done by modifying the note property, the opening and decay sub-property only affect the note slightly. Some people may be able to recognize it, but it is still a subtle change. The opening parameter reproduce the variation of pronounciation through adjustments of the opening mouth, while the decay parameter is the rate which the note drops.

In the VOCALOID software, opening and decay are represented by the value of 0 to 100. The other reason why this is chosen to be modified other than the subtle changes it made, is because a greater value than 100 can still be read by the software, but it still registers as 100. This may be due to the software limiting its values.

Since we can use these two sub-properties and value exceeding 100 won't affect the note any further, we can use both sub-properties to store parts of the hidden message, each holding more than 1 byte. But as mentioned before, storing too many bytes may make it obvious to detection. In this case, the alternative method is to use both opening and decay properties to store either 1 or 2 bytes of the hidden message. The result of the modified note using this algorithm is shown in Table 3.

Table 3. Result of the note modification using the alternative method

Modified note section

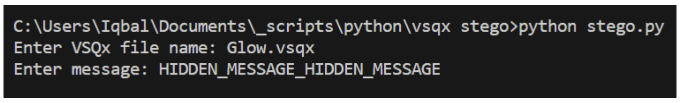
```
<note>
    <t>1800</t>
    <dur>120</dur>
    <n>56</n>
    <v>64</v>
    <y>fu</y>
    <p>p\ M</p>
    <nStyle>
        <v id="accent">50</v>
        <v id="bendDep">8</v>
        <v id="bendLen">0</v>
        <v id="decay">72</v>
        <v id="fallPort">0</v>
        <v id="opening">73</v>
        <v id="risePort">0</v>
        <v id="vibLen">0</v>
        <v id="vibType">0</v>
    </nStyle>
</note>
```

IV. Testing and Result

A. *Message Hiding and Extraction*

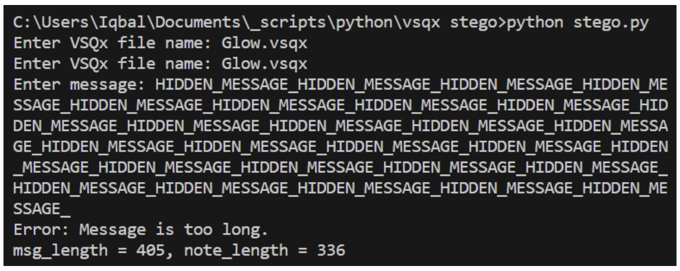
Testing of the steganography program will be using a VSQx file acquired from VocaDB as the cover file. The file contains one track with 336 notes. Using 1 byte per note, the file can hide 336 bytes of message. The program takes the input of the file name and the hidden message as shown in

Figure 4. In case the message is too long for the cover file to hide, an error message will pop up as shown in Figure 5.



```
C:\Users\Iqbal\Documents\_scripts\python\vsqx stego>python stego.py
Enter VSQx file name: Glow.vsqx
Enter message: HIDDEN_MESSAGE_HIDDEN_MESSAGE
```

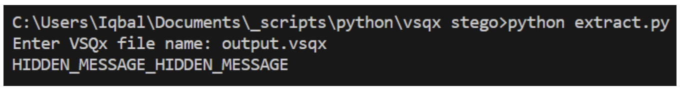
Figure 4. Command line interface for message hiding



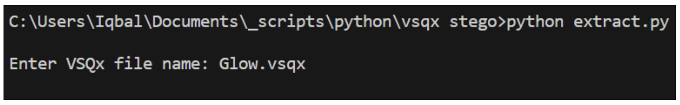
```
C:\Users\Iqbal\Documents\_scripts\python\vsqx stego>python stego.py
Enter VSQx file name: Glow.vsqx
Enter VSQx file name: Glow.vsqx
Enter message: HIDDEN_MESSAGE_HIDDEN_MESSAGE_HIDDEN_MESSAGE_HIDDEN_ME
SSAGE_HIDDEN_MESSAGE_HIDDEN_MESSAGE_HIDDEN_MESSAGE_HIDDEN_MESSAGE_HID
DEN_MESSAGE_HIDDEN_MESSAGE_HIDDEN_MESSAGE_HIDDEN_MESSAGE_HIDDEN_MESSA
GE_HIDDEN_MESSAGE_HIDDEN_MESSAGE_HIDDEN_MESSAGE_HIDDEN_MESSAGE_HIDDEN
_MESSAGE_HIDDEN_MESSAGE_HIDDEN_MESSAGE_HIDDEN_MESSAGE_HIDDEN_MESSAGE_
HIDDEN_MESSAGE_HIDDEN_MESSAGE_HIDDEN_MESSAGE_HIDDEN_MESSAGE_HIDDEN_ME
SSAGE_
Error: Message is too long.
msg_length = 405, note_length = 336
```

Figure 5. Error when the cover couldn't hide the message

Extracting a hidden message from a VSQx file will only take the file name of the VSQx file. The program will print out the hidden message directly into the command line. If there is no hidden message, then the output will be empty. The output of the program is shown in Figure 6.



```
C:\Users\Iqbal\Documents\_scripts\python\vsqx stego>python extract.py
Enter VSQx file name: output.vsqx
HIDDEN_MESSAGE_HIDDEN_MESSAGE
```



```
C:\Users\Iqbal\Documents\_scripts\python\vsqx stego>python extract.py

Enter VSQx file name: Glow.vsqx
```

Figure 6. Command line interface for message extraction

The program can hide and extract a string message successfully, with the limitation in its message size. Adding an extra byte hidden inside a note can increase the limit, but adding it too much will make the message obvious. The program can also hide other file format with the right preprocess, but given the cover file only has 336 notes, it can only fit a very low resolution image even using 4 bytes per note.

B. *Modifying the File*

One of the good criteria of a steganography algorithm is its robustness toward modification. Meaning that modifying the file won't cause the hidden message to be unreadable or unextractable. To test the robustness of the steganography algorithm, the following modification is done to the VSQx file that contains a hidden message using the VOCALOID4 software:

1. Adding a new note at random places.
2. Adding a new track and note.
3. Modifying a note that contains a hidden message.
4. Removing a note that contains a hidden message.
5. Removing a note that doesn't contain a hidden message.

The result of the modifications is as following:

1. Adding a new note anywhere doesn't affect the hidden message as the extraction process ignores any note that doesn't contain a hidden message.
2. Adding a new track also doesn't affect the hidden message for the same reason as above.
3. Modifying a note that contains a hidden message won't affect the message as the VOCALOID software does not change the extra property that is used to store the message.
4. Removing a note that contains a hidden message will change the message. If the character is encoded using one byte (e.g., UTF-8) then a character will be missing, but different encoding may affect the whole message.
5. Removing a note that doesn't contain a hidden message doesn't affect the message.

The test result shows that the steganography algorithm is robust as it is resistant to some modifications. The trade-off is in its perceptibility as an extra line can be found on the note property.

C. *Testing the Alternative Method*

The program takes the same input as the basic algorithm and outputs a VSQx file that contains the secret message. The extraction process also works well and returns the same message as the original. The difference is the alternative method can hide twice as much data as the basic algorithm, and the information is less perceptible as it uses an existing property rather than adding a new one.

However, using an existing property reduces its robustness. The output file from the alternative algorithm is still resistant to modification as it keeps its secret message on addition of a new note, but modifying a note that contains a secret message will damage the secret message.

D. *Quantifying the Audio*

We can compare the difference between the original file and the modified file using PSNR as explained in section 2.C. The VSQx file must be exported into an audio file before computing the PSNR. The audio file that will be used is a 16-bit wav with mono audio. The results of the calculations are shown in Table 4.

Table 4. PSNR result of each method audio.

No.	Method	RMSE	PSNR (dB)
1.	Addition of a new "stego" property.	3.3033e-06	205.9505
2.	Insertion on "decay" property.	0.00023467	168.9202
3.	Insertion on "opening" property.	0.0018255	151.1021
4.	Insertion on "decay" and	0.013192	133.9229

	"opening" property.		

The basic algorithm, which adds a new property rather than modifying an existing one has the lowest error at 3.3 x 10^{-6} and highest PSNR at 205.95 dB. This result may show that adding a property that is unrelated to the VOCALOID software might cause a change in the audio file albeit very slightly. Modifying the decay and opening property lowers the PSNR to 168.9 dB and 151.1 dB respectively, and modifying both property lowers it further to 133.93. This is still an acceptable value as it is higher than 80 dB.

V. Conclusion and Future Works

The VSQx file is feasible to be used as a cover file in steganography. We provided two alternative steganography algorithms, one of which adds a new property that is unrelated to the VOCALOID software, the other modifies an existing property to store the secret message. The former has higher resilience and higher PSNR value, while the latter is less resilience and has lower PSNR but is less perceptible and can contains twice as much data due to using two properties.

The algorithms that are provided here are still the basics and can be improved further. Some suggestions are:

1. Adding cryptography to protect the data using a key. This can be done using symmetric-key algorithm.
2. Exploring other encoding techniques to make the secret message less perceptible.
3. Exploring other properties and methods to increase the capacity and hide bigger files other than texts.

Source Code Link

The repository contains the source code for both the basic and alternative algorithm used to insert and extract information to the VSQx file. An example VSQx file is also provided taken from VocaDB. The GitHub repository can be accessed at:

https://github.com/PlumStream24/vsqx-stego

Acknowledgment

This paper was made as an assignment for the course Cryptography at Institut Teknologi Bandung. I would like to express gratitude to Ir. Rinaldi Munir for teaching cryptography and its application this semester. I would also like to thank my seniors for the inspiration for this paper. I apologize for any mistakes or errors that may be present in this paper. May this paper provide some useful information to the readers.

References

[1] S T Welstead, "Fractal and wavelet image compression techniques." SPIE Publication, 1999. pp. 155–156.

[2] R Munir, "Kriptografi," Penerbit Informatika.

[3] E. Cole, "Hiding in Plain Sight: Steganography and the Art of Covert Communication," Wiley, 2003.