International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 5, Issue 4, ISSN No. 2455-2143, Pages 431-436
Published Online August 2020 in IJEAST (http://www.ijeast.com)

AUDIO STEGANOGRAPHY USING LIVE AUDIO STREAM AS COVER WITH DYNAMIC KEY GENERATION

Namitha M.V
Department of CSE
JNNCE, Shivamogga, Karnataka, India

Dr.Manjula G R
Department of CSE
JNNCE, Shivamogga, Karnataka, India

***Abstract*— This paper presents a novel approach to audio steganography, by considering stream covers (live audio) as cover. A stream cover is a cover where the sender has no access to the entire sequence of numbers in the embedding process as defined by Stefan et al(2000). The cover utilized to hide secret data(stego) and the original cover should be similar and thereby not raising any suspicion to the intruder. Because of this requirement audio files are well suited cover for embedding the secret information as they are popular among humans. In live audio steganography there is an added advantage as the cover is unique each and every time, as it may be the communication between two individuals just like a casual telephonic talk. The intruder will not have the reference audio to compare perceptual and non-perceptual quality of the audio unlike famous audio files which are publicly available. LSB insertion is a common, simple approach to embedding information in a cover. Unfortunately, this approach is vulnerable to steganalysis. Instead of using every byte, it is possible to select some bytes in a random manner according to a secret key. Sharing of this key is a time-consuming process with the risk of man in the middle attack, hence its proposed in this paper where in the key will be based on the current time. The sender and the receiver will generate the key by considering current time as seed followed by mathematical manipulations by bringing the advantage of randomness without the risk of key sharing.**

Keywords— Live audio steganography, HAS (human auditory system), LSB (least significant bit).

I. INTRODUCTION

Different types of communication technologies are existing in today's technical era. Every technology is meant for certain situations with their own tradeoffs. There are many sophisticated technologies to protect the data during its transmission. Few important practices are steganographic and cryptographic methods. Cryptography is the process of keeping secret data in an unintelligible form through various ciphers. Whereas steganography is the process of hiding data in a cover data (image, audio, video) using a tool.

Audio steganography is a technique used to transmit hidden information by modifying an audio signal in an imperceptible manner. Ideally the host message before steganography and stego message after steganography should have the same characteristics.

It is true that human auditory system (HAS) is sensitive, at the same time HAS has many properties which can be exploited to perform audio steganography which retains perceptual transparency. It is both art and science to efficiently embed the secret data by retaining Imperceptibility, good embedding rate, high security, Robustness and reduced computational complexity, as all the requirements are contradictory to each other.

This paper is an effort in the Temporal or time domain with a simple LSB technique along with the novelty of live audio as cover, followed by LSB with random interval method, However in traditional approach the stego key is shared by sender and the receiver but here we are reading the current time through the internet (local system time may differs in different system) to use as a seed to generate the key.

A. Related work

There are 3 categories of audio steganography. Namely, **Temporal or time** domain, **transform** domain and codecs or **compression** domain as categorized by Kaliappan Gopalan (2003) et al in their work.

In temporal domain the cover data itself is modified to hide the secret. There are 3 techniques that come under this category. Namely, **least significant bits (LSBs), echo hiding and silence intervals.**

Data hiding in the LSBs of audio samples is one of the simplest algorithms. However, adjusting of the LSBs of audio samples introduces noise that becomes audible as number of LSBs used for hidden data increases as reported by Nedeljko et al (2002). Bazyar et al (2015) depicts the regular LSB technique is prone to attacks, hence many variants have been proposed.

Fatiha Djebbar et al (2012) says echo hiding method embeds data into audio signals by introducing a short echo to the host signal. The nature of the echo is a resonance added to the host audio. Therefore, the problem of the HAS sensitivity to the

International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 5, Issue 4, ISSN No. 2455-2143, Pages 431-436
Published Online August 2020 in IJEAST (http://www.ijeast.com)

additive noise is avoided. After the echo has been added, the stego signal retains the same statistical and perceptual characteristics.

A simple and effective embedding method can be achieved using silence intervals in speech signal. Initially, the silence intervals of the speech and their respective lengths (the number of samples in a silence interval) are determined. Secret data can be hidden in these intervals by various bit modification methods. This method has a good perceptual transparency but obviously it is sensitive to compression as justified by Fatiha Djebbar et al (2012) .

The literature review in the Fatiha Djebbar et al (2012) also states that there are around 9 different techniques in audio steganography, but none of them used live audio as cover.

However, Real-time Steganography with RTP (Real-time Transport Protocol) has implemented by Druid(2007), there is a challenge listed in the aspect of reliability as RTP uses UDP (User Datagram Protocol), which is connection less protocol. Hence data split across multiple packets may arrive out of order and one or more parts of data split across multiple packets may not arrive at all.

This work uses sockets to perform steganography with live audio as cover, as sockets are reliable advantageous over RTP as it uses TCP/IP (Transmission Control Protocol/Internet Protocol).

The live audio steganography hardly raises any suspicion as it is just a casual conversation between parties at the outset. Live audio causes hindrances for the intruder for statistical analysis of the stego audio as reference audio will not be available. Hence it acts as motivation for the development of steganographic techniques which involves live audio as cover.

Upon that, the dynamically generated key will bring the new dimension in the key based steganography without transmitting the key we are able to obtain the same key at both ends.

The rest of the paper is organized as follows. Proposed embedding and extraction algorithms are explained in section II. Experimental results are presented in section III. Concluding remarks are given in section IV.

II. PROPOSED ALGORITHM

The work was carried out in 4 stages. Each stage is having its own embedding and extraction algorithm. The methodology is improved in each stage in terms of embedding process so that any unintended receiver will find it hard to extract the embedded secret.

1. Hiding secret text bits in LSB of each byte of audio stream starting from first byte

---

Algorithm 2.1: Embedding Process, least significant bit(LSB) substitution

---

for $i = 1, \dots, l(c)$ do
if ($l(m)$)
Embed i^{th} bit of secret in i^{th} byte of live audio in LSB position.
$s_i \leftarrow c_i \leftrightarrow m_i$
end for

The embedding process consists of converting the audio stream as byte array and in every byte each bit of secret data will be placed by replacing the LSB if secret bits exist. Then the updated audio stream will be transmitted in the normal way to the intended destination. This is summarized by Algorithm 2.1.

Algorithm 2.2: Extraction process , least significant bit substitution

---

for $i = 1, \dots, l(c)$ do
if ($l(m)$)
Extract i^{th} bit of secret in i^{th} byte of live audio in LSB position.
$m_i \leftarrow LSB(c_i)$
end for

The extraction process consists of extraction of LSB of each byte till the length of the message from the audio stream received as shown in Algorithm 2.2.

2. Hiding secret text bits in LSB of each byte of audio stream starting from selected byte (the starting byte is obtained by the current time read by the two machines through internet)

Algorithm 2.3: Embedding Process, least significant bit (LSB) substitution starting from chosen byte

---

Generate a key K_1 by using current time *t* as the seed
for $i = 1, \dots, l(c)$ do
$j=i+K_1$
if ($l(m)$)
Embed i^{th} bit of secret in j^{th} byte of live audio in LSB position.
$s_j \leftarrow c_j \leftrightarrow m_i$
end for

The embedding process consists of converting the audio stream as byte array to facilitate the hiding of secret data bits. Current time will be read through the internet, through which a key K_1 will be generated. From the position K_1 in every byte each bit of secret data will be placed by replacing the LSB if secret bits exist. Then the updated audio stream will be transmitted in the normal way to the intended destination. This is summarized by Algorithm 2.3.

International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 5, Issue 4, ISSN No. 2455-2143, Pages 431-436
Published Online August 2020 in IJEAST (http://www.ijeast.com)

Algorithm 2.4: Extraction process, least significant bit (LSB) substitution starting from chosen byte substitution

Generate a key K_1 by using current time *t* as the seed
for $i = 1, \dots, l(c)$ do
j=*i*+K_1
if ($l(m)$)
Extract *i*[th] bit of secret in *j*[th] byte of live audio in LSB position.
$m_i \leftarrow LSB(c_j)$
end for

The extraction process consists of converting the audio stream as byte array to facilitate the unhiding of secret data bits. current time will be read through the internet, through which a key K_1 will be generated. From the position K_1 in every byte each LSB bit is extracted to obtain secret data bits. This is summarized by Algorithm 2.4.

3. Hiding secret text bits in LSB of each byte of audio stream starting from selected byte and also the text to be hidden is encrypted using DES encryption method(the starting byte and the key to the DES encryption can be obtained by the current time read by the two machines through internet)

Algorithm 2.5: Embedding Process, least significant bit (LSB) substitution starting from chosen byte and secret text is encrypted using DES

Generate a key K_1 by using current time *t* as the seed
Generate a key K_2 by using current time *t* as the seed
for $i = 1, \dots, l(c)$ do
j=*i*+K_1
new$m_i \leftarrow (E_{k_2}(m_i))$
if ($l(newm)$)
Embed *i*[th] bit of secret in *j*[th] byte of live audio in LSB position.
$s_j \leftarrow c_j \leftrightarrow newm_i$
end for

The embedding process consists of converting the audio stream as byte array to facilitate the hiding of secret data bits. Current time will be read through the internet, through which two keys K_1 and K_2 are generated. The secret data will be encrypted using key K_2. From the position K_1 in every byte each bit of encrypted secret data will be placed by replacing the LSB if encrypted secret bits exist. Then the updated audio stream will be transmitted in the normal way to the intended destination. This is summarized by Algorithm 2.5

Algorithm 2.6: Extraction Process, least significant bit (LSB) substitution starting from chosen byte and secret text is decrypted using DES

Generate a key K_1 by using current time *t* as the seed
Generate a key K_2 by using current time *t* as the seed
for $i = 1, \dots, l(c)$ do
j=*i*+K_1
if ($l(newm_i)$)
Extract *i*[th] bit of secret in *j*[th] byte of live audio in LSB position.
new$m_i \leftarrow LSB(c_j)$
$m_i \leftarrow (D_{k_2}(newm_i))$
end for

The extraction process consists of converting the audio stream as byte array to facilitate the unhiding of secret data bits. Current time will be read through the internet, through which two keys K_1 and K_2 are generated. From the position K_1 each bit of encrypted secret data will be present in LSB as long as secret data exists. The secret data will be decrypted using key K_2. This is summarized by Algorithm 2.6.

4. Hiding secret text bits in LSB of selected bytes of audio stream starting from selected byte and also the text to be hidden is encrypted using DES encryption method (bytes selection, the starting byte and the key to the DES encryption can be obtained by the current time read by the two machines through internet)

Algorithm 2.7: Embedding Process, least significant bit (LSB) substitution starting from chosen byte and by selecting further bytes based on the step size and secret text is encrypted using DES

Generate a key K_1 by using current time *t* as the seed
Generate a key K_2 by using current time *t* as the seed
Generate step size *S* by using current time *t* as the seed
for $i = 1, \dots, l(c)$ do
j=*i*+K_1
new$m_i \leftarrow (E_{k_2}(m_i))$
if ($l(newm)$)
Embed *i*[th] bit of secret starting from *j*[th] byte of live audio in the step of *S* in LSB position.
$s_j \leftarrow c_j \leftrightarrow newm_i$
end for

International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 5, Issue 4, ISSN No. 2455-2143, Pages 431-436
Published Online August 2020 in IJEAST (http://www.ijeast.com)

The embedding process consists of converting the audio stream as byte array to facilitate the hiding of secret data bits. Current time will be read through the internet, through which two keys K_1, K_2 and *S* are generated. The secret data will be encrypted using key K_2. From the position K_1 in selected bytes based on the step size *S* each bit of encrypted secret data will be placed by replacing the LSB if encrypted secret bits exist. Then the updated audio stream will be transmitted in the normal way to the intended destination. This is summarized by Algorithm 2.7.

Algorithm 2.8: Extraction Process, least significant bit (LSB) substitution starting from chosen byte and by selecting further bytes based on the step size and secret text is encrypted using DES

Generate a key K_1 by using current time *t* as the seed
Generate a key K_2 by using current time *t* as the seed
Generate step size *S* by using current time *t* as the seed
for $i = 1, \dots, l(c)$ do
$j=i+K_1$
if $(l(newm_i))$
Extract i^{th} bit of secret in j^{th} byte of live audio in the step of *S* in LSB position.
$newm_i \leftarrow LSB(c_j)$
$m_i \leftarrow (D_{k_2}(newm_i))$
end for

The extraction process consists of converting the audio stream as byte array to facilitate the unhiding of secret data bits. Current time will be read through the internet, through which two keys K_1, K_2 and *S* are generated. From the position K_1 in selected bytes based on the step size *S* each bit of encrypted secret data will be present in LSB as long as secret data exists. The secret data will be decrypted using key K_2. This is summarized by Algorithm 2.8.

III. Experiment and Results

A. Experimental Setup

To implement a steganographic technique using live audio as a cover, two systems are connected in the network using sockets as shown below in Figure 1. The setup in the Figure 1 indicates both 'Machine A' and 'Machine B' are physically and logically connected and they can transmit data to each other.

'Machine A' is connected with the microphone to record audio to be sent to 'Machine B'. 'Machine A' sends audio along with secret data and sends to the 'Machine B', by using the live audio as cover. 'Machine B' receives and plays the audio via speaker and also displays the secret message to the receiver.

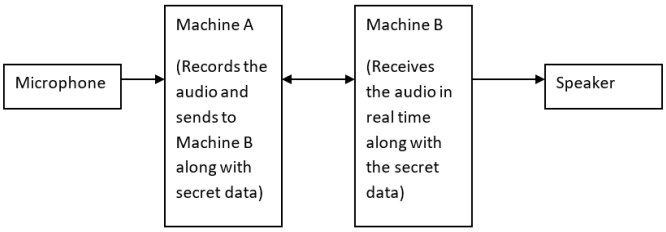

Figure 1: 'Machine A' and 'Machine B' are connected via sockets.

3.1 Procedure at 'Machine A'

STEP1: 'Machine A' establishes the logical connection with 'Machine B'

STEP2: User inputs a casual audio message to the 'Machine A' (which will be sent to 'Machine B' in terms of CHUNKS (pre decided quantity to buffer the live audio) after embedding).

STEP3: 'Machine A' facilitates the embedding of secret data with the audio using embedding algorithm.

STEP4: 'Machine A' feds audio with secret data to the 'Machine B'.

3.2 Procedure at Machine B

STEP1: 'Machine B' waiting for incoming connections.

STEP2: 'Machine B' accepts the connection with 'Machine A'.

STEP3: 'Machine B' receives audio (embedded with secret) from 'Machine A' in terms of CHUNKS (pre decided quantity to buffer the live audio) and plays it to the user of 'Machine B'.

STEP4: 'Machine B' facilitates the extraction of secret data with the audio using extracting algorithm.

STEP5: go to STEP1.

3.3 Embedding Algorithm.

The Recorded audio after converting into byte array, each and every byte is considered and the LSB of each byte is

International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 5, Issue 4, ISSN No. 2455-2143, Pages 431-436
Published Online August 2020 in IJEAST (http://www.ijeast.com)

replaced with one bit of secret data. The algorithm is as follows.

STEP1: Read the secret data from the file.

STEP2: Convert the secret data into bits.

STEP3: Consider audio in terms of byte array.

STEP4: Embed according to chosen algorithm (Algorithm 2.1/2.3/2.5/2.7)

3.4 Extracting Algorithm.

The Received audio after converting into byte array, each and every byte is considered and the LSB of each byte is extracted to form secret data. The algorithm is as follows

STEP1: Consider the audio Received.

STEP2: Convert the Audio in to array of bytes.

STEP3: Extract according to chosen algorithm in embedding procedure (Algorithm 2.2/2.4/2.6/2.8)

STEP4: Print it on the output screen for User on 'Machine B'.

B. Results

In this section results are discussed. Results are considered in two aspects. First and foremost is the amount of data embedded in different audio stream by 4 different algorithms depicted in Table 3.1.

Table 3.1: Amount of data that can be hidden according to various algorithm in terms of bits in different length of audio stream.

Sl. No	Audio time in second	Amount of data that can be hidden according to various algorithm in terms of bits.			
		Algorithm 3.1 and Algorithm 3.2	Algorithm 3.3 and Algorithm 3.4	Algorithm 3.5 and Algorithm 3.6	Algorithm 3.7 and Algorithm 3.8
1	179	32768	32768 - *n*	32768 – *n -(c+r)*	[32768 – *n -(c+r)]/S*
2	119	16384	16384 -*n*	16384 - *n -(c+r)*	[16384 – *n -(c+r)]/S*
3	59	8192	8192 - *n*	8192 – *n -(c+r)*	[8192 – *n -(c+r)]/S*
4	29	4096	4096 - *n*	4096 – *n -(c+r)*	[4096 – *n -(c+r)]/S*
5	14	2048	2048– *n*	2048– *n -(c+r)*	[2048 – *n -(c+r)]/S*

The above table is showcasing the results with respect to 4 algorithms used. According Algorithm 2.1 and Algorithm 2.2, 14 seconds of audio stream is able to hold 2048 bits of secret data and the pattern continues in the same way till 32768 bits of data.

According Algorithm 2.3 and Algorithm 2.4, 14 seconds of audio stream is able to hold 2048— *n* bits of secret data where $n = [roundup(j/8) * 8]$, j is obtained by $j=i+K_1$ Where K_1 is generated by using current time *t* as the seed at both embedding and extracting algorithm and the pattern continues in the same way till 32768 bits of data.

According Algorithm 2.5 and Algorithm 2.6, 14 seconds of audio stream is able to hold 2048— *n -(c+r)* bits of secret data where *n* is obtained in the same way as in Algorithm 2.3 and 2.4. The values of *c* and *r are* increase in length of original secret data bits after encryption and radix-64 conversion respectively and the pattern continues in the same way till 32768 bits of data.

According Algorithm 2.7 and Algorithm 2.8, 14 seconds of audio stream is able to hold *[*2048— *n -(c+r)]/S* bits of secret data where *n, c, r* is obtained in the same way as in Algorithm 2.5 and 2.6. The value of *S* is obtained by using current time *t* as the seed at both embedding and extracting algorithm and the pattern continues in the same way till 32768 bits of data.

Secondly difference between delay in reaching 'Machine B' (Receiver) without embedding secret data and Delay in reaching 'Machine B' (Receiver) with embedding secret data in terms of seconds stream by 4 different algorithms in Table 3.2. Since the cover is live audio, much delay raises suspicion.

Table 3.2: Delay in reaching 'Machine B' (Receiver) without embedding secret data and with embedding secret data in terms of seconds.

Sl. No	Delay in reaching 'Machine B'(Receiver) with embedding secret data according to various algorithm in terms of seconds.				
	Without embedding Secret data	Embedding According to Algorithm 3.1 & 3.2	Embedding According to Algorithm 3.3 & 3.4	Embedding According to Algorithm 3.5 & 3.6	Embedding According to Algorithm 3.7 & 3.8
1	0.32	0.31	0.32	0.32	0.34
2	0.29	0.29	0.32	0.33	0.33
3	0.34	0.32	0.32	0.32	0.33
4	0.36	0.32	0.32	0.32	0.32
5	0.29	0.32	0.30	0.31	0.32
6	0.30	0.33	0.32	0.31	0.32
7	0.29	0.32	0.30	0.30	0.33
8	0.34	0.32	0.32	0.34	0.33
9	0.29	0.32	0.33	0.33	0.34
10	0.30	0.31	0.33	0.34	0.32
Average	0.312	0.316	0.320	0.322	0.328

As per the data available in Table 3.2 the time required sending the data to the 'Machine B' from 'Machine A' for different amount of data in bitsusing different algorithms is not having considerable amount of difference. Hence secret data can be sent without any additional processing time by ensuring the live aspect of the audio stream which is used as cover.

IV. CONCLUSION

This paper presents a novel approach by considering live audio as cover for steganography, which stands advantageous in the aspect of steganalysis as each time the cover is different, and the cover is not available to anyone else for perceptual and non-perceptual analysis. There is no considerable additional delay because of the process of embedding and extraction in all the 4 sets of Algorithms, As depicted in the Table 3.2 Hence the approach is considerable in the field of live audio steganography. The direction of our work is making the intruders job tough but at the same time the hiding capacity is getting considerably reduced. In future we would like to take up work in a direction where in hiding capacity should be considerably good.

V. ACKNOWLEDGMENT

This work is partially supported by Vision Group on Science and Technology under CISEE and Jawaharlal Nehru National College of Engineering(JNNCE), Shivamogga.

VI. REFERENCE

[1] Nedeljko Cvejic, Tapio Seppben(2002) Increasing the capacity of LSB-based audio steganography, IEEE workshop on Multimedia Signal Processing, (pp.336-338)

[2] Bazyar M, Sudirman R (2015) A New Method to Increase the Capacity of Audio Steganography Based on the LSB Algorithm.,Jurnal Teknologi 74(6), (pp.49–53)

[3] Kaliappan Gopalan(2003) Audio steganography using bit modification Multimedia and Expo, international Conference on Acoustics, Speech, & Signal Processing 49, (pp.629-632)

[4] Yahya T.Qassim, Tim R.H. Cutmore, David D. Rowlands(2016) Optimized FPGA based continuous wavelet transform, Computers and Electrical Engineering, (pp.84–94)

[5] Ahmed Hussain Ali, Loay Edwar George, A. A. Zaidan, Mohd Rosmadi Mokhtar(2018) High capacity, transparent and secure audio steganography model based on fractal coding and chaotic map in temporal domain, Multimed Tools Appl 77,(pp.31487–31516)

[6] Fatiha Djebbar, Beghdad Ayad, Karim Abed Meraim and Habib Hamam(2012) Comparative study of digital audio steganography techniques, EURASIP Journal on Audio, Speech, and Music Processing.

[7] Druid, "Real-time Steganography with RTP", *DefCon*, (2007) Available Online*:* https*://www.*defcon.org/images/defcon-15/dc15-presentations/dc-15-druid.pdf

[8] Stefan Katzenbeisser, Fabien A.P. Petitcolas(2000) Information Hiding Techniques for Steganography and Digital Watermarking, Artech House Print on Demand. (Textbook)

[9] Jayaram P, Ranganatha H R, Anupama H S (2011) Information hiding using audio steganography – a survey, The International Journal of Multimedia & Its Applications (IJMA) Vol.3, No.3.

[10] Gunjan Nehru, Puja Dhar (2012) A Detailed look of Audio Steganography Techniques using LSB and Genetic Algorithm Approach, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 2.

[11] Ramya Devi Ra, D Pugazhenthib Quaid (2016) Ideal Sampling Rate to reduce distortion in Audio Steganography, International Conference on Computational Modeling and Security (CMS 2016) available online at www.scincedirect.com, Procedia Computer Science 85 (pp.418 – 424)

[12] Sheelu, Enhancement of Data Hiding Capacity in Audio Steganography,(2013) IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p- ISSN: 2278-8727,Volume 13, Issue 3 (pp. 30-35)