Audio Steganography with Asymmetric Cryptography

Sai Aditya Thalluri
aditya369thalluri@unm.edu
University of New Mexico
Albuquerque, New Mexico

ABSTRACT

The aim of this project is to implement audio steganography along with asymmetric cryptography on top of it as an additional layer of security. It is a programmatic implementation in Python language which takes the audio file, secret text as inputs and embeds the text into audio file. These audio files can be transferred to whomever it is supposed to be rather than sending the secret text directly. Algorithms such as Least Significant Bit (LSB) can be used to embed the text into audio files. Before embedding the text into audio, it will be public key encrypted using RSA algorithm for more security. This project also includes the extraction of text back from audio files and decrypting it with private key which will only be available with the receiver. Sender and Receiver can share the secret keys and only they will be able to decrypt the message once they extract it from audio file.

KEYWORDS

audio steganography, asymmetric cryptography, public key encryption, least significant bit, RSA algorithm.

1 INTRODUCTION

With the increasing security attacks and concerns around the world, there developed many ways to secure information transfer. The main objective is to prevent the message from being stolen by attackers and also hide its existence. Steganography is one such technique of hiding the files or information within a carrier file. There are many types of carriers such as image, audio, text, video, etc. The word steganography comes from Greek origin which means "hiding or covering". It has become very popular in the communication world where we will make sure that every information we transfer is secure. When compared with other cryptographic techniques, a good measure with steganography is that attackers will hardly get to know that there is some secret message within an image, audio or Text. Because embedding the message is not going to change the look of an image, frequency of audio wave or readability of text [1]. They allow a user to hide even a large amount of data within the carrier files. This is what attracts many people to use steganography for secure communication.

Steganography and cryptography are closely related to each other. Steganography was used heavily before the cryptographic systems were developed. Cryptography is generally used for privacy whereas steganography is meant for secrecy which means both of them are intended to make communications secure. They are usually combined to get double protection. Also, these two techniques are not efficient individually as they do together. In this project, we are combining audio steganography with public key encryption. With audio steganography, we hide messages within audio files which provides secrecy. Many algorithms can be used for embedding the information in the audio files. We use the Least Significant Bit(LSB) algorithm. RSA being one of the strongest algorithms is the best choice to implement public key encryption and provide confidentiality for the information.

2 BACKGROUND

In steganography, secret text can be hidden in any of the carrier files such as image, audio, video, text, etc. There are many algorithms available to hide text in carrier files. Least Significant Bit (LSB) algorithm is one such method embedding the secret text into carrier files. Programmatic operations are generally performed in bytes. Least bit is the rightmost bit of a byte which has lest value among all the bits. According to LSB, even if the least bit of every byte in the carrier is changed it is not going to create a considerable change [3]. Take the text we want to hide and convert into bytes. After converting the carrier file into bytes, a least significant bit of each byte will be replaced with the bit of secret text. Then the modified bytes are again converted into carrier file. In this way, message bytes are embedded in carrier frames in such a way that it should not change the originality of carrier and no attacker would be able to know that there is something hidden in that. The same process will be reverted at the carrier destination to get back the original message from it.In audio steganography, audio is used as a carrier file[4]. Converting the audio file into bytes, the secret text is embedded into it using the LSB algorithm.

Asymmetric cryptography is using different keys for encryption and decryption. It is also known as public key encryption. In public key encryption, we have public and private key pair. While embedding the secret text into the carrier, first encrypt it with the public key of the destination. So when the carrier file is sent to its destination, only the person with the corresponding private key can extract and decrypt the message. Even if an attacker captures the carrier file and extracts the message from it, they won't be able to decrypt it without the private key. The strength of cryptography depends on the strength of the keys used. RSA is a public key algorithm well known for its key strength named after the initials of its co-founders Rivest, Adi Shamir, and Leonard Adleman. It is widely used and is the best choice to provide confidentiality of messages being transferred. It's keys are based on the factorization of two large prime numbers and are very difficult to crack[6]. To generate the public and private key pair in this project, we are using RSA algorithm. It is been already taught in class lectures along with practical assignments on generating keys using this algorithm.

3 METHODOLOGY

To implement this, I have divided the whole process into two steps as shown below.

- Encrypting the secret text with RSA public key and embedding it into the audio file.
- Extracting the text from the audio file and decrypting it with RSA private key to the original message.

I choose only audio files of WAV extension as carrier files for this project. Because other extensions such as MP3 don't follow standard formatting of bytes and keep changing from one to another. WAV is the standard uncompressed bitstream of audio used by Microsoft and IBM. I have tried to implement on MP3 files too, but it didn't work out and audio will no longer sound like the original[2]. So WAV files are best to work upon for audio steganography.

To implement the first part, I will generate an RSA key pair in Python using the library *crypto*. Then take the message from the user that is to be embedded and encrypt with the public key and convert it to hex. Convert the hex into bytes. Then take the carrier audio file and convert it to frames(i.e., bytes). Let't not touch the first 44 frames as they contain header information about the audio properties. So to embed the message in the audio, I used the Least Significant Bit(LSB) algorithm. It is the most simple and efficient algorithm for steganography. LSB algorithm is replacing the least significant digit of each audio frame with each message bit until all the message bits are embedded. Starting from the 45th frame of an audio file, apply LSB between message bytes and audio frames [3, 5]. With the modified audio frames generate a new audio file using the same parameters of the former and it sounds exactly the same.

Below diagrams provide a more clear understanding of embedding the messages into an audio file.

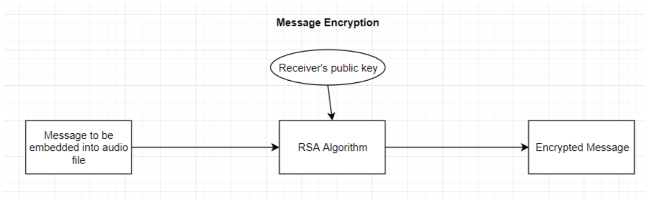


Figure 1: Encrypting the message with RSA public key before embedding it into audio file.

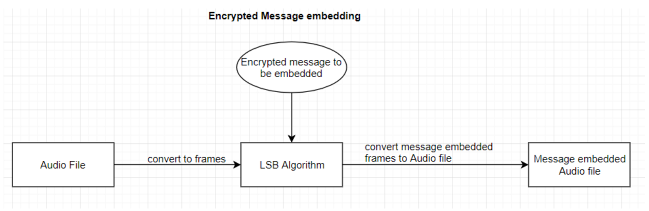

Encrypted Message embedding
Encrypted message to be embedded
Audio File
convert to frames
LSB Algorithm
convert message embedded frames to Audio file
Message embedded Audio file

Figure 2: Embedding the encyrpted message into audio file using LSB algorithm.

Now let's implement the second part which is the extraction of a message from audio file. First, convert the audio file with message embedded in it to frames(or bytes). According to the same Least Significant Bit technique which is used while embedding the message, extract the encrypted message. Starting from the 45th byte of the audio frames, collect the least significant bit from each of the frames. Considering it as encrypted message hex, convert it into ASCII format. Then by using the RSA private key generated before, decrypt it with private key [5]. This process should return the original message which is embedded in the audio file.

Below diagrams provide a more clear understanding of extracting the messages from the audio file.

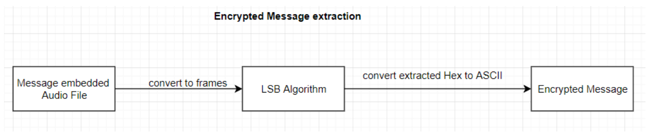

Encrypted Message extraction
Message embedded Audio File
convert to frames
LSB Algorithm
convert extracted Hex to ASCII
Encrypted Message

Figure 3: Extracting the embedded message from the audio file using LSB algorithm.

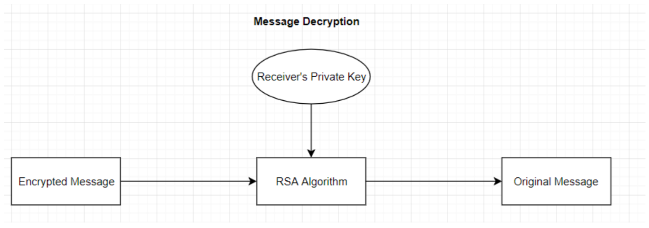


Figure 4: Decrypting the extracted message using RSA private key to get the original message.

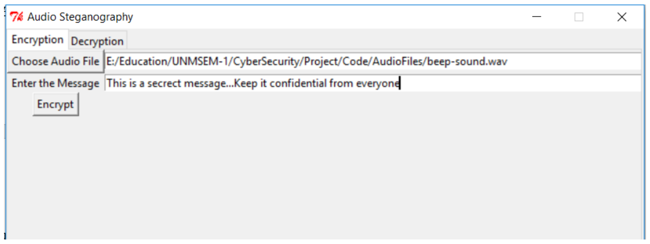
Coming to the programmatic implementation, I wrote the code into two according to the steps discussed above in python. For the first step, I wrote code into a file *EmbedMessageIntoAudio.py* with separate functions for generating RSA key pair, encrypting the message, embedding an encrypted message into audio frames and converting the audio frames into a new audio file. For the second step too, I wrote code in a file *ExtractMessageFromAudio.py* with separate functions for extracting the message from an audio file and decrypting it with the generated RSA private key.

Apart from these, I have also developed a UI application in Python using package *tkinter* to make the whole process simple. It will have two tabs encryption and decryption, which will take the corresponding inputs from the user and call corresponding code files to describe above. The tab encryption represents embedding the message into the audio file. It takes an audio file and message as inputs and executes the code in *EmbedMessageIntoAudio.py* generating a new audio file in the same directory. The tab decryption only takes message embedded audio file as input, extracts the encrypted message and decrypts it to show the original message on UI by executing the code file *ExtractMessageFromAudio.py*.

4 RESULTS AND DISCUSSION

For the audio steganography tool described above, lets give the input and check the results. Lets show results using the UI application.

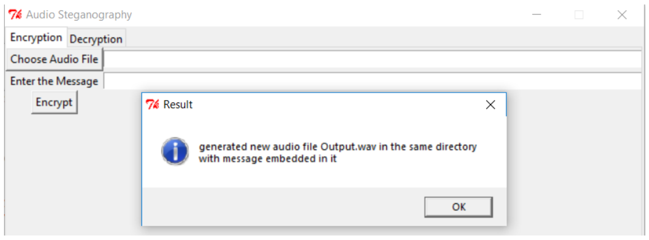
To evaluate the first step of methodology which is the message embedding process, I will provide inputs to the encryption tab in the developed audio steganography UI application. It requires two inputs, one of which is to browse the audio file and the other is the message to be embedded. After providing both and on click of the encrypt button, it should perform the whole process and generate a new audio file with message embedded in it. Also, the new audio file should sound exactly the same as the original one. Below images show the results of the execution of the first part.



E:/Education/UNMSEM-1/CyberSecurity/Project/Code/AudioFiles/beep-sound.wav

This is a secrect message...Keep it confidential from everyone

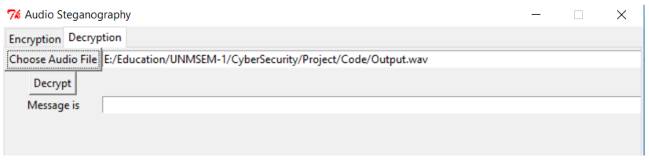
Figure 5: Encrypting and embedding the message into audio file using the audio steganography tool



generated new audio file Output.wav in the same directory with message embedded in it

Figure 6: Generation of new audio file with the message embedded frames using audio steganography tool

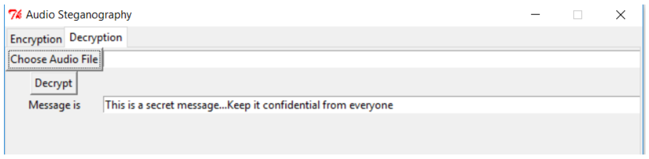
So the tool has generated a new audio file with the message embedded and it sounds the same as the original one. Now lets implement the extraction process using the same audio generated in the first step. I do this using the decryption tab in an audio steganography UI application. It requires only one input which is a path to the audio file with the message embedded in it. It takes the audio file path, executes the message extraction process and should display the original message in a text box on the tab on click of the decrypt button. Below images show the result of the message extraction. The message should be the same as before and should not vary even a single character.



E:/Education/UNMSEM-1/CyberSecurity/Project/Code/Output.wav

Figure 7: Browsing to the message embedded audio file using the audio steganography tool

As shown in the images above, by choosing the image path and on click of the decrypt button it extracts the message and displays



This is a secret message...Keep it confidential from everyone

Figure 8: Extracting and decrypting the embedded message to show it on the UI using the audio steganography tool

it on the text box which is the same as the original message. So this audio steganography tool can be used to safely embed messages into audio files and send them to appropriate destinations. Only the receivers will be able to extract and decrypt the messages as they share the keys and information with them. Even if any attacker catches the audio files, first of fall they won't be able to detect the embedded messages and cannot do anything without the decryption key even if they extract the message. This is how it gives double protection on information security by combining steganography and cryptography.

5 CONCLUSION

Therefore, I have successfully implemented the audio steganography along with public key cryptography. To do the whole process I have also developed a user interface tool in python which makes it so easy to embed and extract the messages from audio files rather giving the inputs manually on the console. It is also easy to transfer the application to many users rather than transferring and asking them to execute it. We used audio files with WAV extension as only they follow standard header and body structure. Whereas the structure can change from file to file for other extensions. If we have more time we can look more into the problems in audio steganography with files other than WAV extension. More research can be done on what makes inefficient with files such as MP3. It can also be considered as a research project as steganography has a huge scope in the real world. By developing the audio steganography tool to even more standards, it can be commercialized as a secure information sharing product.

REFERENCES

[1] Wojciech Mazurczyk Elzbieta Zielinska and Krzysztof Szczypiorski. 2012. Development Trends in Steganography. *arXiv* (Feb. 2012), 36–44. https://doi.org/arXiv:1202.5289
[2] Ghazali Sulong Akram Zeki Mohammed Salem Atoum, Subariah Ibrahimn and Adamu Abubakar. 2013. Exploring the Challenges of MP3 Audio Steganography. In *International Conference on Advanced Computer Science Applications and Technologies*. IEEE, Washington DC, USA. https://doi.org/10.1109/ACSAT.2013.38
[3] Junaid Gilani Muhammad Asad and Adnan Khalid. 2011. An enhanced least significant bit modification technique for audio steganography. In *International Conference on Computer Networks and Information Technology*. IEEE, Washington DC, USA. https://doi.org/10.1109/ICCNIT.2011.6020921
[4] G. G. Rajput and Ramesh Chavan. 2017. A Novel Approach for Image Steganography based on LSB Technique. In *Proceedings of the International Conference on Compute and Data Analysis (ICCDA '17)*. ACM, New York, NY, USA, 167–170. https://doi.org/10.1145/3093241.3093247
[5] Shivani Sharma. 2007. Audio Steganography using ZDT. In *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies (ICTCS '16)*. ACM, New York, NY, USA. https://doi.org/10.1145/2905055.2905272
[6] Xin Zhou and Xiaofei Tang. 2011. Research and implementation of RSA algorithm for encryption and decryption. In *Proceedings of 2011 6th International Forum on Strategic Technology*. IEEE, Washington DC, USA. https://doi.org/10.1109/IFOST.2011.6021216