



JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY

Information Security - Lab

Project-Based-Learning

TITLE

Audio Steganography using RSA Algorithm

GROUP MEMBERS:

Abhijot Singh - 19103180 - B6

Kshitij Shakya - 19103181 - B6

Ayush Jaiswal- 19103185 - B6

Vansh Sachdeva - 19103194 - B6

Problem Statement:

To ensure confidentiality and protection against theft or loss of important data, we need to make sure that no one can access them without our permission.

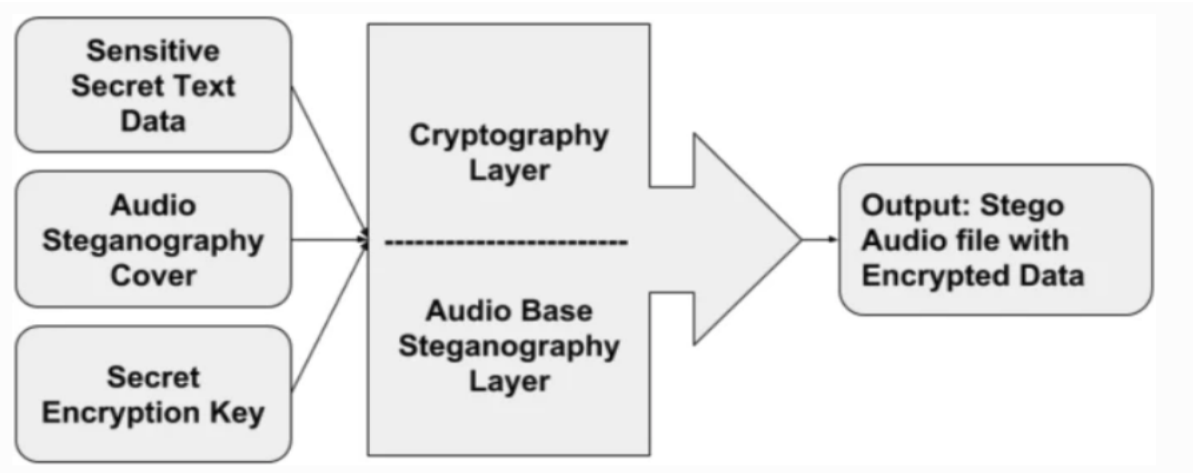
To tackle this problem, we have created this project which is about hiding the secret message into the audio. This technique is used to secure the transmission of secret information or hide their existence. Additional layer of security is attached by encrypting the message beforehand using highly secured RSA encryption.

Abstract:

This project proposes a system for protecting sensitive text data on a PC by combining both encryption and steganography techniques. In the project, a system is created by including RSA encryption followed by audio-based steganography as two consecutive layers to ensure the highest possible security and take advantage of both. We have modelled, implemented, and tested the system to investigate security, capacity, and data dependency relationships.

Introduction:

Generally, you need to protect sensitive information stored on your personal computer, such as email messages, credit card information, and company information. Hiding sensitive confidential data within an individual computer has the advantage that the files available on the computer are used as cover media. To provide users with confidence and security in protecting the information on their PCs, encryption and steganography techniques can be combined to hide sensitive data. The security achieved by using steganography to hide sensitive data in a cover sheet relies on the belief that no one can suspect that the data is hidden. However, if someone notices that the title media has changed, you can discover sensitive data. Therefore, it is recommended that you use another technique, such as encryption, to encrypt sensitive data before hiding it on cover media. This prevents anyone from learning the content, even if the embedded text is detected, because it is encrypted. Therefore, to enhance security, a combination of the two techniques can be used to ensure that confidential data remains intact or is not misused in the event of a very difficult security breach. This project proposes a restoring security system that uses audio-based steganography and RSA encryption as a data link layer, depending on the use of files available from the PC. Techniques for hiding the system include encryption and steganography as two layers to ensure complete protection of sensitive information on the PC. Steganography is generally a science that hides information through some process in different types of media like Text, images, voice, voice. After combining the secrets in the cover object, the result file is called the Stego File.



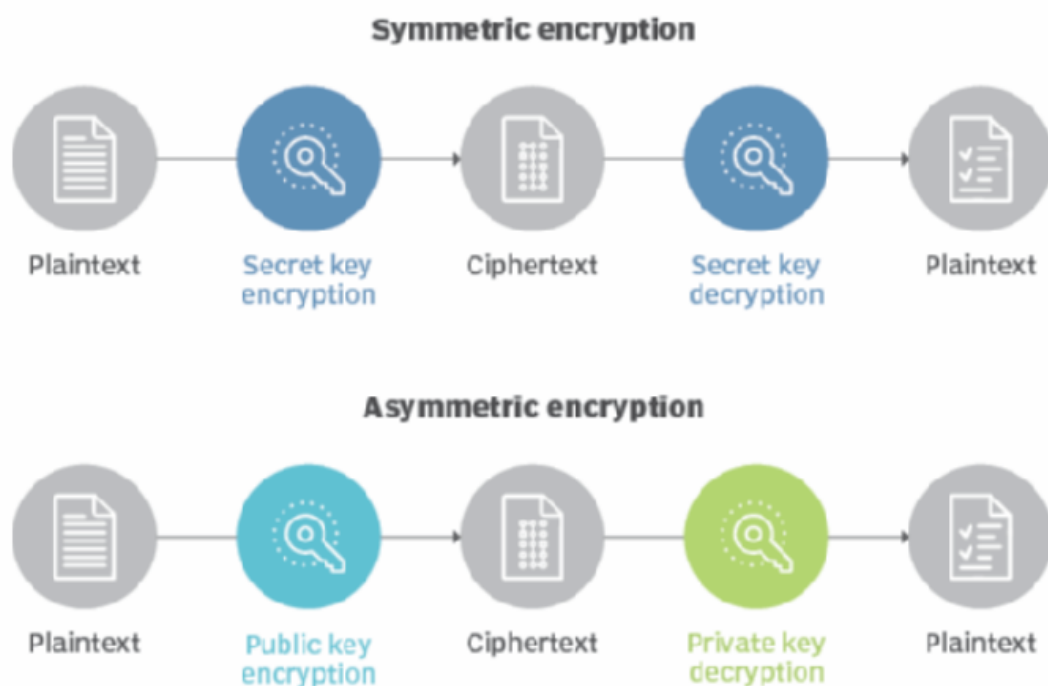
The first layer, encryption, uses RSA encryption, and the second layer, steganography, relies on audio-based steganography. This hides the encrypted data with the least significant bit (LSB). Proposed to increase the LSB to improve capacity. The main improvement in this study over other similar tasks is the ability to hide confidential information for personal use by using some of the available audio PC files that are considered undetectable media. This task revolutionises human hearing in security applications that are not normally used. It proves that it may be acceptable or more similar to other security applications that are completely different from normal visual security research.

As a second layer of this security system, encryption is fundamentally different from steganography. Cryptography is known as coding a secret plaintext into a ciphertext. Encryption requires a private key in the encoding / decoding process to protect the secret data. In our system, the secret text data that passes through the encryption layer using the security key is followed by the steganography layer, which produces a file containing secret encrypted data called the stego audio file. Figure above gives an overview of the two-tier security system.

About the RSA Algorithm:

The RSA algorithm is an asymmetric encryption algorithm. Asymmetry actually means that it works with two different keys. These are Public and Private keys. As the name implies, the public key is given to everyone and the private key is kept secret.

The RSA (Rivest-Shamir-Adleman) algorithm is the foundation of a cryptographic system. It is a set of cryptographic algorithms which are used for some particular security services or purposes such as to enable public key cryptography. RSA is used oftenly for the protection of personal/sensitive data from the Internet.



RSA encryption allows you to encrypt messages with both public and private keys. The public key is used to encrypt the message derived from the server. But now, the same key isn't needed for decryption. Private key is used to decrypt the message. This feature is one of the reasons behind the world wide usage of RSA, by providing assurity of confidentiality, integrity and reliability of electronic communications and data storage.

It is also used in software programs. Browsers are a clear example because they need to establish secure connections and verify digital signatures over insecure networks such as the Internet.

RSA signature verification is frequently used operations in today's world of the networked systems.

Why is the RSA algorithm used?

The working of RSA and how it secures the message is based on the difficulty of factoring large numbers, which are the products of two large prime numbers. Multiplying these two numbers is easy, but even today's supercomputers take time, and it is considered impossible to find the original prime number from the sum or factorization.

The public and private key generation algorithms are the most complex part of RSA encryption. The two large primes p and q are generated using the RabinMiller prime test algorithm. Module n is calculated by multiplying p and q . This number is used for both public and private keys and creates a link between them. Its length, usually specified in bits, is called the key length.

The public key consists of module n and public index e . This is not a prime number that is too large, so it is usually set to 65537. The public key is shared with everyone, so the electronic number does not have to be a secretly selected prime number. The private key consists of a modulus n and a secret exponent d . It is calculated using the extended Euclidean algorithm to find the reciprocal of n with respect to the torient.

How Secure is RSA?

RSA security relies on the difficulty of factoring large integers. With increased computing power and the discovery of more efficient factorization algorithms, the ability to factor even larger numbers will also improve.

The strength of the encryption is directly linked to the key size. Doubling the key length reduces performance but can increase strength exponentially. RSA keys are usually 1024-bit or 2048-bit long, but experts believe that 1024-bit keys are no longer completely protected from all attacks.

As a result, governments and some industries are moving to a minimum key length of 2048 bits. Elliptic curve cryptography (ECC) is an alternative to RSA for implementing public key cryptography, although it will take years before a longer key is needed, except for an unexpected breakthrough in quantum computing. It is gaining popularity among many security professionals. You can create faster, smaller, and more efficient encryption keys.

The latest hardware and software are ECC compliant and may become more popular. It offers comparable security, computing power and low battery consumption, making it more suitable for mobile apps than RSA. Researchers, including Adi Shamir, co-inventor of the RSA, used acoustic cryptographic analysis to successfully generate a 4096-bit RSA key. However, keep in mind that cryptographic algorithms are vulnerable to attacks.

Steganography and Cryptography

Cryptography (the science of writing secret code) deals with all the elements needed to communicate securely over insecure channels: data protection, confidentiality, key exchange, authentication, and non-repudiation. However, encryption does not provide secure communication.

The advantage of steganography over Cryptography only is that the message is unnoticed by itself, the messenger, or the recipient. The goal of Cryptography is to make data unreadable by a Third Party, whereas, Steganography's Goal is to hide data from third parties. Steganography and Cryptography are often used together to secure secret messages.

SCOPE OF STEGANOGRAPHY:

Steganography is a very interesting and advantageous science these days and has following uses:

- **Digital Watermarking:**
To protect the copyright of information. Photo collections sold on CD often have hidden messages in their photos that allow them to detect fraudulent use. The same techniques used for DVDs are even more effective as the industry is developing DVD recorders that detect and ban copies of protected DVDs.
- The simplest and oldest are used in mapping. Cartographers may add small fictitious roads to their maps for the purpose of prosecuting Copycat.
- Steganography does not only apply to written communication. Radio and television news from World War II to the present can be used to hide codes and hidden messages. Some government sources suspect that Osama bin Laden's recorded video played on television stations around the world contains a hidden message.
- Even biological data, stored on DNA, may be a candidate for hidden messages, as biotech companies seek to prevent unauthorized use of their genetically engineered material. The technology is already in place for this: three New York researchers successfully hid a secret message in a DNA sequence and sent it across the country.

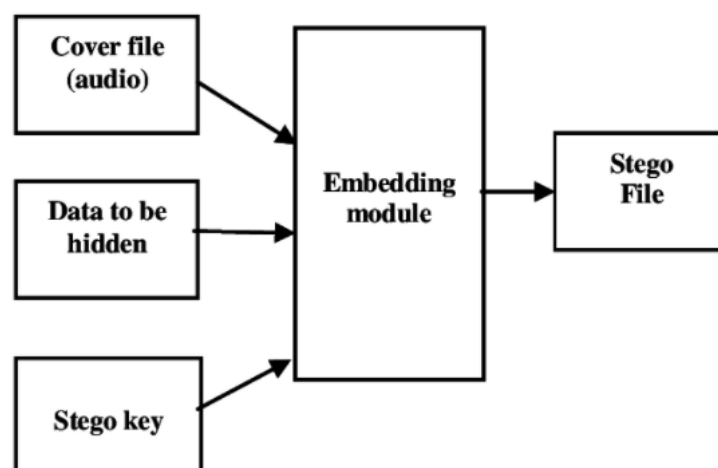
Introduction to Audio Steganography:

Audio steganography is the hiding of secret messages in the audio. It is the science of hiding some secret text or audio information in a host message. This is a technique used to protect the transmission of confidential information or to hide its existence. You can also add confidentiality to a secret message if the message is encrypted.

The method of Audio Steganography can embed the messages in WAV, AU files, and even MP3 sound files as well.

Working of Audio Steganography:

The pre-steganography host message and the post-steganography stego message have the same properties. Embedding a secret message in digital audio is a more difficult process. Various technologies have been established for embedding information in digital audio. In this project, we used the LSB method. The least significant bit (LSB) technology is one of the simplest approaches to secure data transmission.



Working of Project:

The project is divided into three parts:

1. **RSA File Segment** -

This file handles most of the steganography parts that are needed in this project.

The main functions of this file include generation of a random set of public key and private key for asymmetric encryption.

Along With this, the decryption of cipher text using the correct set of private keys is also taken care of using its functions.

2. **Encode File Segment** -

As its name suggests, this file is used to take an audio input file from the user and a secret message to hide inside it. The audio file is used as a cover object to hide the secret message, without changing the file size of the original audio.

After encryption of the message, a set of public and private key which was used to hide the message in audio is given to the user, which he/she can use at the time of decryption to successfully extract the correct message.

The user also has the option to save output in any format as desired as per his convenience.

3. **Decode File Segment** -

This file is responsible for extracting the secret message from the audio file.

The user is prompted to provide the audio file which has an encrypted message hidden inside it. Then this audio file is parsed to separate the audio file contents from the cipher text.

Finally, the user is prompted to enter his set of private keys, which will be used in converting the cipher text back to its plain text.

The secret message can only be decoded using the right set of private keys, in the absence of which, the message will be in a non-readable format.

CODE:

1. RSA.py:

```
1 from math import sqrt
2 import random
3
4 def gcd(a, b):
5     if b == 0:
6         return a
7     else:
8         return gcd(b, a % b)
9
10
11 def mod_inverse(a, m):
12     for x in range(1, m):
13         if (a * x) % m == 1:
14             return x
15     return -1
16
17
18 def isprime(n):
19     if n < 2:
20         return False
21     elif n == 2:
22         return True
23     else:
24         for i in range(2, int(sqrt(n)) + 1, 2):
25             if n % i == 0:
26                 return False
27     return True
28
29
30 p = random.randint(1, 1000)
31 q = random.randint(1, 1000)
32
33 def gen_key(p, q):
34     keysize = pow(2, 4)
35     nMin = 1 << (keysize - 1)
36     nMax = (1 << keysize) - 1
37     primes = [2]
38     start = 1 << (keysize // 2 - 1)
39     stop = 1 << (keysize // 2 + 1)
40
41     if start >= stop:
42         return []
43
```

```

44     for i in range(3, stop + 1, 2):
45         for p in primes:
46             if i % p == 0:
47                 break
48         else:
49             primes.append(i)
50
51     while (primes and primes[0] < start):
52         del primes[0]
53
54     while primes:
55         p = random.choice(primes)
56         primes.remove(p)
57         q_values = [q for q in primes if nMin <= p * q <= nMax]
58         if q_values:
59             q = random.choice(q_values)
60             break
61     n = p * q
62     phi = (p - 1) * (q - 1)
63     e = random.randrange(1, phi)
64     g = gcd(e, phi)
65
66     while True:
67         e = random.randrange(1, phi)
68         g = gcd(e, phi)
69         d = mod_inverse(e, phi)
70         if g == 1 and e != d:
71             break
72
73     return ((e, n), (d, n))
74
75
76 def encrypt(msg_plaintext, package):
77
78     e, n = package
79     msg_ciphertext = [pow(ord(c), e, n) for c in msg_plaintext]
80     return msg_ciphertext
81
82
83 def decrypt(msg_ciphertext, d, n):
84
85     msg_plaintext = [chr(pow(c, d, n)) for c in msg_ciphertext]
86     return ''.join(msg_plaintext)
87

```

2. encode.py:

(Here, we are importing the “RSA” file that we have implemented as shown before)

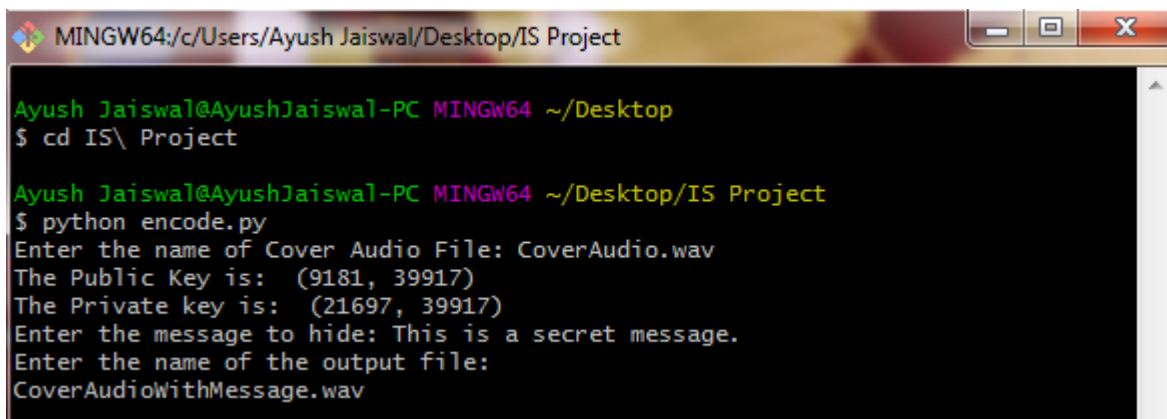
```
1 from RSA import *
2 import wave
3
4 audioCoverName = input("Enter the name of Cover Audio File: ")
5 coverAudio = wave.open(audioCoverName, mode='rb')
6 AudioInBytes = bytearray(coverAudio.readframes(coverAudio.getnframes()))
7
8 publicKey, privateKey = gen_key(p,q)
9
10 print("The Public Key is: ", publicKey)
11 print("The Private key is: ", privateKey)
12
13 message = input("Enter the message to hide: ")
14
15 secretMsg= str(encrypt(message, publicKey))
16
17 secretMsg = secretMsg + int((len(AudioInBytes)-(len(secretMsg)*8*8))/8) * '|'
18 bit_arr = list(map(int, ''.join(bin(ord(i)).lstrip('0b')).rjust(8,'0') for i in secretMsg)))
19
20 for i, j in enumerate(bit_arr):
21     AudioInBytes[i] = (AudioInBytes[i] & 254) | j
22
23 AudioAfterEncode = bytes(AudioInBytes)
24
25 print("Enter the name of the output file: ")
26 outputName = input()
27 with wave.open(outputName, mode='wb') as Done:
28     Done.setparams(coverAudio.getparams())
29     Done.writeframes(AudioAfterEncode)
30 coverAudio.close()
```

3. decode.py:

```
1 import wave
2 from RSA import decrypt
3
4 def converter(msg):
5     l = len(msg)
6     c=[]
7     tp=''
8     for i in range(l):
9         if(msg[i]=='['):
10             continue
11         elif(msg[i]==']'):
12             d=int(tp)
13             c.append(d)
14             tp=''
15             break
16         elif(msg[i].isdigit() == True):
17             tp = tp+msg[i]
18         elif(msg[i]==','):
19             d=int(tp)
20             c.append(d)
21             tp=''
22     return c
23
24 AudioModified = input("Enter the name of the audio you want to decode: ")
25
26 AudioFile = wave.open(AudioModified, mode='rb')
27 AudioInBytes = bytearray(AudioFile.readframes(AudioFile.getnframes()))
28
29 extracted = [AudioInBytes[i] & 1 for i in range(len(AudioInBytes))]
30 rough = "".join(chr(int("".join(map(str,extracted[i:i+8])),2)) for i in range(0,len(extracted),8))
31
32 secretMessageEncrypted = rough.split("|")[0]
33 AudioFile.close()
34
35 PrivateKey1 = int(input("Enter the private key(part 1): "))
36 PrivateKey2 = int(input("Enter the private key(part 2): "))
37
38 cipher = converter(secretMessageEncrypted)
39 print("The hidden message is: " + decrypt(cipher,PrivateKey1,PrivateKey2))
40
```

Output :

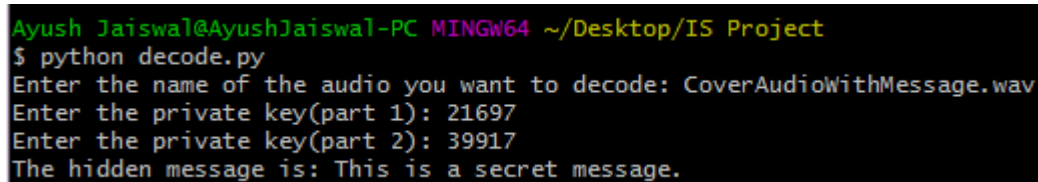
Encrypting message in audio file -



```
MINGW64:/c:/Users/Ayush Jaiswal/Desktop/IS Project
Ayush Jaiswal@AyushJaiswal-PC MINGW64 ~/Desktop
$ cd IS\ Project

Ayush Jaiswal@AyushJaiswal-PC MINGW64 ~/Desktop/IS Project
$ python encode.py
Enter the name of Cover Audio File: CoverAudio.wav
The Public Key is: (9181, 39917)
The Private key is: (21697, 39917)
Enter the message to hide: This is a secret message.
Enter the name of the output file:
CoverAudioWithMessage.wav
```

Decrypting secret message using private keys -



```
Ayush Jaiswal@AyushJaiswal-PC MINGW64 ~/Desktop/IS Project
$ python decode.py
Enter the name of the audio you want to decode: CoverAudioWithMessage.wav
Enter the private key(part 1): 21697
Enter the private key(part 2): 39917
The hidden message is: This is a secret message.
```