**PERSONAL INFORMATION AS CIPHER HIDDEN IN AUDIO (PICHA)**

**A PROJECT REPORT**

###### ***Submitted by***

##### Aman Kumar Pandey (20BCY10147)

**Anand Singh** (20BCY10150)

**C. Sai Anand** (20BCY10184)

**Dheeraj Yadav** (20BCY10212)

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

*in*

# COMPUTER SCIENCE AND ENGINEERING

****

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**

**VIT BHOPAL UNIVERSITY**

**KOTHARI KALAN, SEHORE**

**MADHYA PRADESH - 466114**

##### DEC 2021

**VIT BHOPAL UNIVERSITY, KOTHARI KALAN, SEHORE**

**MADHYA PRADESH – 466114**

**BONAFIDE CERTIFICATE**

Certified that this project report titled **“PERSONAL INFORMATION AS CIPHER HIDDEN IN AUDIO (PICHA)”** is the bonafide work of **“AMAN KUMAR PANDEY (20BCY10147) ANAND SINGH (20BCY10150) C. SAI ANAND (20BCY10184) DHEERAJ YADAV (20BCY10212)”** who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported at this time does not form part of any other project/research work based on which a degree or award was conferred on an earlier occasion on this or any other candidate.



**PROGRAM CHAIR PROJECT GUIDE**

Dr. R. Rakesh, Assistant Professor Dr. Muneeswaran V, Senior Assistant Professor

School of Computer Science and Engineering School of Computer Science and Engineering

VIT BHOPAL UNIVERSITY VIT BHOPAL UNIVERSITY

The Project Exhibition I Examination is held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ACKNOWLEDGEMENT**

First and foremost, I would like to thank the Lord Almighty for His presence and immense blessings throughout the project work.

I wish to express my heartfelt gratitude to Dr. R. Rakesh, Program Chair, Division of Cyber Security and Digital Forensics, Assistant Professor, School of Computing Science and Engineering (SCSE) for much of his valuable support and encouragement in carrying out this work.

I would like to thank my internal guide Dr. Muneeswaran V, Senior Assistant Professor, School of Computing Science and Engineering (SCSE) for continually guiding and actively participating in my project, giving valuable suggestions to complete the project work.

I would like to thank all the School of Computer Science, Department of Cyber Security, and Digital Forensics technical and teaching staff who extended all support directly or indirectly.

Last, but not least, I am deeply indebted to my parents who have been the greatest support while I worked day and night for the project to make it a success.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | List of Abbreviations  List of Figures and Graphs  List of Tables  Abstract | iv  vii  viii  ix |
| 1 | **CHAPTER-1:**  **PROJECT DESCRIPTION AND OUTLINE** Introduction 1.2 Motivation for the work  1.3 About the project and Technique  1.5 Problem Statement  1.6 Objective  1.7 Organization of the project  1.8 Summary | 1-2  1  1  1  2  2  2 |
| 2 | **CHAPTER-2:**  **RELATED WORK INVESTIGATION**  2.1 Introduction  2.2 Core area of the project  2.3 Existing Approaches/Methods  2.3.1 Approach -1  2.3.2 Approach -2  2.4 Pros and Cons  2.5 Issues and Observations  2.6 Summary | 3-5  3  3  3  3  3  4  4  5 |
| 3 | **CHAPTER-3:**  **REQUIREMENT ARTIFACTS**  3.1 Introduction  3.2 Hardware and Software requirements  3.3 Specific Project requirements  3.3.1 Data requirement  3.3.2 Functions requirement  3.3.3 Performance and security requirement  3.4 Summary | 6-7  6  6  6  6  7  7  7 |
| 4 | **CHAPTER-4:**  **DESIGN METHODOLOGY AND ITS NOVELTY**  4.1 Methodology and goal  4.2 Functional modules  4.3 Software Architectural designs  4.4 User Interface designs  4.5 Summary | 8-10  8  8  9  9  10 |
| 5 | **CHAPTER-5:**  **TECHNICAL IMPLEMENTATION & ANALYSIS**  5.1Outline  5.2 Code solutions  5.3 Working Layout  5.4 Prototype submission  5.5 Test and validation  5.6 Performance Analysis  5.7 Summary | 11-21  11  11  19  20  20  21  21 |
| 6 | **CHAPTER-6:**  **PROJECT OUTCOME AND APPLICABILITY**  6.1Outline  6.2 Key implementations outline of the System  6.3 Significant project out  6.4 Project applicability on Real-world applications  6.5 Inference | 22-23  22  22  22  23  23 |
| 7 | **CHAPTER-7:**  **CONCLUSIONS AND RECOMMENDATION**  7.1 Limitations and Constraints  7.2 Future Enhancements  7.3 Inference | 24-25  24  24  25 |
| 8 | Glossary  References | 26  27 |

**LIST OF ABBREVIATIONS**

* PICHA: - Personal Information as Cipher Hidden in Audio
* LSB: - Least Significant Bit
* RSA: - Rivest, Shamir, Adleman
* GUI: - Graphical User Interface
* GCD: - Greatest Common Divisor
* ASCII: - American Standard Code for Information Interchange

**LIST OF FIGURES AND GRAPHS**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **TITLE** | **PAGE NO.** |
| Figure 5.1 | The first look at the GUI for the sending party | 31 |
| Figure 5.2 | The first look at the GUI for the receiving party | 31 |
| Graph 5.1 | The graph of % change in size vs the text file size | 33 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE NO.** | **TITLE** | **PAGE NO.** |
| Table 4.1 | Description of all the modules used in P.I.C.H.A with their functions and usage. | 19 |
| Table 5.1 | Experiment by taking different plain text sizes and calculating the % change in audio size. | 32 |

**ABSTRACT**

This particular project deals with cryptography and steganography, using them to hide data and give them a layer of security. The idea is to make use of their essence and formulate a tool that can implement both of them. Cryptography secures the data while steganography hides it. Using RSA cryptosystem as well as LSB audio steganography the data can be hidden and transferred accordingly. Although there is a tradeoff between capacity and security, this issue can be eliminated by dividing data into several parts and sending them via different keys. Moreover, choosing what is more important between capacity and security can help make the tool useful. The coding of the whole project is done in python and the GUI is made by the Tkinter package in python.

**PROJECT DESCRIPTION AND OUTLINE**

**INTRODUCTION: -**

The concept of data privacy is really important in the field of computer science. Nobody sane would want their data to be distributed to strangers for no good reason. This is where cryptography comes into play. It makes your data unreadable and non-understandable. The only one who can read this would be the person who has the key for your “lock”. There are two types of cryptography- asymmetric key and symmetric key cryptography. Asymmetric key cryptography is when we get two keys to work with while symmetric key cryptography is the one where only one key is deployed.

While cryptography makes your data unreadable, steganography makes your data undetectable. In short, we “hide” the data so that it won't be visible to any person. The only person who can access it would be the one who knows something about the file and has the corresponding tool to do the extraction. There are quite a few steganographic methods. Steganography requires a “cover” to remain undetectable. This cover can be an image, audio, or any text file itself.

**MOTIVATION FOR THE WORK: -**

Humans are beings that fear something or the other. Some are hydrophobic but the others are acrophobic. There is also one kind of phobia that drives us forward- the fear of the unknown. Modern usage has confined it to just fear of strangers or foreigners but its meaning was quite broad. In our case we fear our privacy getting invaded, we fear our data being stolen by the unknown, we fear being used by the unknown in an unknown way. This fear of the unknown is what drives us forward.

**ABOUT THE PROJECT AND TECHNIQUE: -**

This project is a small prototype of a tool that can use steganography and cryptography. This tool implements the RSA technique to lock the data in an unreadable ciphertext. The data is then hidden in an audio cover. It would then be transferred to any desired PC by any means. The tool would be put to use once again and extract the ciphertext from the audio cover. The private key of the receiver will then be used to get back the original message which was to be transferred.

**PROBLEM STATEMENT: -**

The handling of highly sensitive data has always been an issue with many companies and even defense personnel. If that kind of data is compromised, a great loss can occur which can reach hundreds of million dollars to several billion dollars. For handling this kind of data and messages, a special tool that can encrypt the data as well as hide it is required. Major loss can be prevented if the data is hidden perfectly out of the sight of an intruder. Moreover, if he finds it by chance, it would require the private key to decrypt the data. The wrong key if entered would give the same output which would not make any sense.

**OBJECTIVE: -**

Creating a prototype of a tool that can help keep sensitive information encrypted as well as hidden. An easy-to-use GUI interface that can interact with the user through keyboard and mouse can help the non-programmers use the tool thus achieving abstraction. Dividing the entire code into different modules and then importing it to a final GUI input can make the program more readable, easy to understand, and easy to debug as well. The modules can be used by any programmer to get references.

**ORGANIZATION OF THE PROJECT: -**

Aman Kumar Pandey (20BCY10147) was responsible for the writing and modification of the code as well as creating the GUI interface, regulating the project flow. Anand Singh (20BCY10150) was responsible for writing code, researching, and finding reference material. C. Sai Anand (20BCY10184) was in charge of putting together the presentation, testing the application on various platforms, and detecting errors. Dheeraj Yadav (20BCY10212) was responsible for creating the presentation template and diagrams, testing the application in different conditions, and getting bugs if any.

**SUMMARY: -**

The combined effort of all four of us has led to the successful completion of this prototype. In the course of completing this project, we have learned details about RSA and its implementation, audio steganography, and its implementation as well as making any GUI application in python using the Tkinter package.

**RELATED WORK INVESTIGATION**

**INTRODUCTION: -**

A substantial amount of investigation has to be done to ensure that the project is good and useful for starting a project. From getting a basic understanding of the similar models available on the internet to finding their faults and making a better model that can be used by a large number of people, everything comes under investigation.

**CORE AREA OF THE PROJECT: -**

The core area of this project is to secure highly sensitive data through steganography. Hiding the data would make it undetectable to any potential attackers. The output that we would get bypassing the audio file through the tool would be pretty identical making it more confusing to guess the actual steganographic audio file. The security of the file would be ensured by the RSA technique. Entering the wrong key would give an output that would just not make any sense.

**EXISTING METHODS AND APPROACHES: -**

Approach 1: Simple approach by [Sumit Arora](https://sumit-arora.medium.com/audio-steganography-the-art-of-hiding-secrets-within-earshot-part-2-of-2-c76b1be719b3)

In this approach, Sumit has implemented only the steganography hiding technique by taking the files which should be moved beforehand to the parent directory. The command is run and the message in the string variable is hidden in the cover named “song.wav”. The output would be obtained as an audio file named “song\_embedded.wav”. To get the output the other script is run and the input would be provided automatically. The output would be obtained in the in-built terminal of the IDE (Microsoft VSCode in our case).

Approach 2: Command Line Interface approach by [TechChip](https://www.youtube.com/watch?v=UPQD7L9FNrk&ab_channel=TechChip)

The approach involves implementing the OS package in python and giving it a command-line interface that can be accessed by Command Prompt in Windows and Terminal in Mac and Linux. You can choose the message by typing it and the song by loading it in the parent directory beforehand.

**PROS AND CONS: -**

The 1st approach was very simple. Due to its simplicity, it became very easy for a programmer to go through the code and understand it. Changing the values present in the variables before runtime can change the output.

Having a command line interface by implementing the OS package of python, the 2nd approach made the program more interactive for a programmer. The message and file can be selected by changing the values at runtime. The output obtained would be an audio file (in encoding) and the message is printed on the screen (in extracting).

If the 1st approach is said to be the base, then the 2nd approach would be the extension of the first approach. Although the program was a little complex, the interaction with the help of the command line interface made it worthwhile.

Both the approaches only dealt with steganography. Not using cryptography can leave the data vulnerable, which may be exploited if a brute force approach is used. All the files should be put in the parent directory beforehand. Failing in doing so would result in the false execution of the program. From a non-programmer’s point of view, the usage of the two approaches is a bit difficult and can be confusing. It would be an inconvenience to the newbies to run the code or to understand how the script is working.

**ISSUES AND OBSERVATIONS: -**

First and foremost, the addition of a pretty strong cryptosystem is required to secure the data barely up to the standards. If we want to secure confidential and priceless data, we should be prepared thoroughly. To make this tool user-friendly, a GUI should be used instead of a command-line interface. Using the GUI would make it a more interactive and easier-to-use tool, thus increasing the audience who can implement it. Dividing the whole source code into different modules which can be called upon to do their job can make this tool faster and reliable. It can also be helpful to the newbies to understand it thoroughly. It also made the debugging of the source code very easy.

**SUMMARY: -**

Taking into consideration the various pros and cons, we have tried our best to keep the pros and reduce or eliminate the cons. In the journey to make this prototype more user-friendly and helpful, we have implemented several strategies and eliminated the ones that may be cumbersome, thus raising the efficiency.

**REQUIREMENT ARTIFACTS**

**INTRODUCTION**

For an application to run, the system has some minimum requirements that need to be met. As we all know software and hardware go side by side when running an application. Hardware is required for computation running the application. The software would tell it how to function and what to give as the output. Both have their uses in any system.

**HARDWARE AND SOFTWARE REQUIREMENTS**

P.I.C.H.A. has certain hardware and software requirements that are:

Hardware requirements:

· Processor – intel i3 7th generation and above

· Hard Disk – 500 GB HDD

· Memory – above 4 GB RAM

Software requirements:

· Windows 7 or higher

· Visual Studio Code

· Python 3 or above

**SPECIFIC PROJECT REQUIREMENTS**

Data Requirement: -

The basic data which we require before starting the tool would be the audio file (the cover) in which the data would be hidden and the actual data in the form of a text file made in any text editor.

In case you are on the receiving end of the communication, the steganographic audio should be known along with your pair of private keys to start extracting the data.

Functions Requirement: -

Prompting the user to upload the necessary data and launching would encrypt the data as well as encode it. The data will be hidden with no data loss and thus you can transfer the data to the receiver who would be waiting with his pair of private keys to decode the content. Pretty secure and reliable communication can be observed here.

Performance and Security Requirement: -

Apart from the basic requirements given above, the higher the computational ability of the computer, the higher will be the speed of encryption - encoding and decryption - extraction. The performance of the system is proportional to the speed with which the application works because the application contains a lot of mathematical operations which need to be carried out to get the output we desire.

The security requirements would be simple but strict at the same time. DISCLOSE YOUR PUBLIC KEY BUT DO NOT DISCLOSE YOUR PRIVATE KEY. Once the private key is disclosed anyone can see the contents of the audio file and thus the first line of defense is breached. If he can pinpoint the audio file then the second line of defense is also breached and your data would be compromised. So, it should be noted that private keys must not be disclosed.

**SUMMARY: -**

The crux of the above-mentioned points is basic to basic computers having the latest python compiler that can run this application. The higher the specs the better. Certain requirements should be fulfilled beforehand and the personal private key should never be compromised.

**DESIGN METHODOLOGY AND ITS NOVELTY**

**METHODOLOGY AND GOALS: -**

The methodology behind the implementation of PICHA was a python script that could merge two programs- an audio steganography program and an RSA cryptography program. The two programs would be used in such a way that first the RSA program would encrypt the data then the encrypted ciphertext would be sent to the audio steganography program which would hide the data. After that, the write mode of the wave package would write an audio file that would be transported from one place to another using any means.

On the receiver’s side, the program of the extraction would be executed which would extract the ciphertext from the audio. The ciphertext would then be taken and given as input to the RSA script. Along with the private key of the receiver, the program would take in the ciphertext. It would then run and give you the secret message which you have wanted.

The goal for doing these is simple- we need to secure the confidential data. We are not concerned with the less valuable data. The only thing we are concerned about is the methods that we will implement to make our tool encrypt as well as encode it. If kept along with a bunch of other audio files, the attacker would be confused.

**FUNCTIONAL MODULE: -**

|  |  |
| --- | --- |
| **utilities.py** | This module is packed with mathematical tools like GCD, prime checking, and modulo inverse. It also has a custom converter that converts a list-like string to a list. |
| **RSA.py** | This module has the encryption and decryption of the RSA cryptography. It also has the key-generation algorithm which would be used to generate keys. |
| **encode.py** | This module deals with the encoding of the text in the audio. Dividing the frames into bytes, making of the byte array of the song, conversion of the ASCII of the string into a byte array. Hiding the byte array into the LSB of the byte array of the song and finalizing and writing the new song, all these things happen here. |
| **extract.py** | This module deals with extraction. Similar to the encoding module, the song is opened and a byte array is made. LSB is taken and then the ASCII value is calculated. The characters obtained are appended to get the plaintext. |
| **SenderGUI.py** | The GUI is made here using the Tkinter package. The interface is created and is attached to the encode.py as well as the utility.py code. |
| **ReceiverGUI.py** | The GUI is made using the same Tkinter package. The interface is linked with extract.py |

Table 4.1

button.png, upload.png, icon.ico - support files for building GUI.

Plaintext, Wahran\_Randall.wav - example files for testing.

**SOFTWARE ARCHITECTURAL DIAGRAM: -**

The SenderGUI script would be executed. The keys would be generated with the help of the key generate function and would be executed by pressing the generate key button. The file would be browsed and their paths would be saved into a variable. The public key would also be the input. It must be noted that the public key of the receiver is used. The key generated would also belong to the receiver. When the application is launched, the value of the path of the song and text file is passed into the encoding function. The value of the public key is also taken with the help of the get() function from the IntVar taken from the input field. Finally, a beautifully created audio file having the contents of the text file is presented in the directory, along with a prompt saying that the task has been completed.

In the ReceiverGUI the private key is taken with the help of an entry widget as an IntVar. The location of the steganographic audio containing the message is also taken. With all the things provided, the application is launched. Finally, a text file is created as ‘OUTPUT.txt’ is created, pretty smoothly and simply.

**USER INTERFACE DESIGN: -**

The Sender Interface contains three types of widgets - labels, entries, buttons. The heading label is shown, followed by the browse dialog box which can be done by importing the file dialog of the Tkinter package. In that, we need to use the askopenfilename function so that we can get only the pathname of the mentioned file. The buttons would have that function (open\_text or open\_song) in their command attribute. The ‘Generated Key’ button is linked with the RSA module’s gen\_key() function. Finally, the ‘Launch’ button is used to take in the input from the entry widget and start the encoding function present in the RSA module. After that, a message box is deployed with the help of the message box module’s ‘showinfo’ message box.

In the ReceiverGUI script, the buttons take the path of the file and store it. There is the entry widget that helps to get data (Private Key) from the user and pass all of that to the LAUNCH function which would be executed when the ‘Launch’ button is pressed. The LAUNCH function also has the extraction function call. After that function is executed successfully, the ‘showinfo’ message box will confirm that the task is completed.

**SUMMARY: -**

The easy-to-use GUI with the backing of a reliable cryptosystem like RSA and hidden from the eyes of the attacker with the help of LSB audio steganography will make it a pretty reliable tool that can defend your confidentiality. Moreover, the division of the whole project into distinct parts will make it reusable and easy-to-use to understand from a programmer’s point of view.

**TECHNICAL IMPLEMENTATION & ANALYSIS**

**OUTLINE: -**

The usage of several packages and functions is what makes a program perfect to execute. Without their usage, the readability of the program will become less. Importing just everything without properly using them would only take a toll on the computer memory and won’t do any significant job.

**CODE SOLUTION: -**

* **utilities.py:** The utility files

*from* math *import* sqrt

*def* *gcd*(a, b):

*if* b *==* 0:

*return* a

*else*:

*return* gcd(b, a *%* b)

*def* *mod\_inverse*(a, m):

*for* x *in* range(1, m):

*if* (a *\** x) *%* m *==* 1:

*return* x

*return* *-*1

*def* *isprime*(n):

*if* n *<* 2:

*return* False

*elif* n *==* 2:

*return* True

*else*:

*for* i *in* range(2, int(sqrt(n)) *+* 1, 2):

*if* n *%* i *==* 0:

*return* False

*return* True

*def* *converter*(msg):

l *=* len(msg)

c *=* []

tp *=* ''

*for* i *in* range(l):

*if*(msg[i] *==* '['):

*continue*

*elif*(msg[i] *==* ']'):

d *=* int(tp)

c.append(d)

tp *=* ''

*break*

*elif*(msg[i].isdigit() *==* True):

tp *=* tp*+*msg[i]

*elif*(msg[i] *==* ','):

d *=* int(tp)

c.append(d)

tp *=* ''

*return* c

* **RSA.py:** The file for RSA implementation.

*import* random

*from* utilities *import* *\**

p *=* random.randint(1, 1000)

q *=* random.randint(1, 1000)

*def* *gen\_key*(p, q):

keysize *=* pow(2, 4)

nMin *=* 1 *<<* (keysize *-* 1)

nMax *=* (1 *<<* keysize) *-* 1

primes *=* [2]

start *=* 1 *<<* (keysize *//* 2 *-* 1)

stop *=* 1 *<<* (keysize *//* 2 *+* 1)

*if* start *>=* stop:

*return* []

*for* i *in* range(3, stop *+* 1, 2):

*for* p *in* primes:

*if* i *%* p *==* 0:

*break*

*else*:

primes.append(i)

*while* (primes *and* primes[0] *<* start):

*del* primes[0]

*while* primes:

p *=* random.choice(primes)

primes.remove(p)

q\_values *=* [q *for* q *in* primes *if* nMin *<=* p *\** q *<=* nMax]

*if* q\_values:

q *=* random.choice(q\_values)

*break*

n *=* p *\** q

phi *=* (p *-* 1) *\** (q *-* 1)

e *=* random.randrange(1, phi)

g *=* gcd(e, phi)

*while* True:

e *=* random.randrange(1, phi)

g *=* gcd(e, phi)

d *=* mod\_inverse(e, phi)

*if* g *==* 1 *and* e *!=* d:

*break*

*return* ((e, n), (d, n))

*def* *encrypt*(msg\_plaintext, e, n):

msg\_ciphertext *=* [pow(ord(c), e, n) *for* c *in* msg\_plaintext]

*return* msg\_ciphertext

*def* *decrypt*(msg\_ciphertext, d, n):

msg\_plaintext *=* [chr(pow(c, d, n)) *for* c *in* msg\_ciphertext]

*return* (''.join(msg\_plaintext))

* **encode.py:** The module for encoding text into audio.

*from* RSA *import* *\**

*import* wave

*def* *encoding*(e, n, song\_path, text\_path):

song *=* wave.open(song\_path, mode*=*'rb')

fr\_bytes *=* bytearray(song.readframes(song.getnframes()))

*with* open(text\_path, 'r') *as* *file*:

plaintext *=* *file*.read()

sct\_msg *=* str(encrypt(plaintext, e, n))

sct\_msg *=* sct\_msg *+* int((len(fr\_bytes)*-*(len(sct\_msg)*\**8*\**8))*/*8) *\** '|'

bit\_arr *=* list(

map(int, ''.join(bin(ord(i)).lstrip('0b').rjust(8, '0') *for* i *in* sct\_msg)))

*for* i, j *in* enumerate(bit\_arr):

fr\_bytes[i] *=* (fr\_bytes[i] *&* 254) *|* j

fr\_mod *=* bytes(fr\_bytes)

*with* wave.open("Output.wav", mode*=*'wb') *as* Done:

Done.setparams(song.getparams())

Done.writeframes(fr\_mod)

song.close()

* **extract.py:** The module for extraction of the ciphertext from audio.

*import* wave

*from* RSA *import* decrypt

*from* utilities *import* converter

*def* *extraction*(d, n):

song *=* wave.open("Output.wav", mode*=*'rb')

fr\_bytes *=* bytearray(song.readframes(song.getnframes()))

ext *=* [fr\_bytes[i] *&* 1 *for* i *in* range(len(fr\_bytes))]

rough *=* "".join(

chr(int("".join(map(str, ext[i:i*+*8])), 2)) *for* i *in* range(0, len(ext), 8))

sct\_msg *=* rough.split("|")[0]

song.close()

cipher *=* converter(sct\_msg)

plain *=* decrypt(cipher, d, n)

*with* open("OUTPUT.txt", 'w') *as* f:

f.write(plain)

* **SenderGUI.py:** The module where the GUI for the sender’s part is made.

*from* encode *import* *\**

*from* tkinter *import* *\**

*from* tkinter *import* font

*from* tkinter *import* filedialog

*from* tkinter *import* messagebox

root *=* Tk()

root.geometry("750x400")

root.title("Personal Information as Cipher Hidden in Audio (PICHA)")

root.iconbitmap("Icon.ico")

*def* *open\_song*():

*global* song\_path

song\_path *=* filedialog.askopenfilename(

initialdir*=*"C:/Users/Aman Pandey/OneDrive/Desktop/PICHA")

*return* song\_path

*def* *open\_text*():

*global* text\_path

text\_path *=* filedialog.askopenfilename(

initialdir*=*"C:/Users/Aman Pandey/OneDrive/Desktop/PICHA")

*return* text\_path

*def* *LAUNCH*():

d *=* part1entry.get()

n *=* part2entry.get()

encoding(d, n, song\_path, text\_path)

messagebox.showinfo("Status", "DONE\nCheck The Parent Directory")

*def* *gen\_dialog*():

public, private *=* gen\_key(p, q)

messagebox.showwarning(

"THESE ARE YOUR KEYS", *f*"Public Key: {public} and Private Key: {private}\nKEEP THEM SAFE")

heading *=* Label(text*=*"Steganographic Encoding", font*=*"algerian 16 ")

heading.grid(row*=*0, column*=*3)

button\_img *=* PhotoImage(file*=*'button.png')

generate\_btn *=* Button(root, image*=*button\_img,

command*=*gen\_dialog, borderwidth*=*0)

generate\_btn.grid(row*=*1, column*=*3)

search\_song *=* Label(root, text*=*"UPLOAD THE SONG: -",

font*=*"comicsansms 12 ", padx*=*5, pady*=*20)

search\_song.grid(row*=*1, column*=*0)

search\_text *=* Label(root, text*=*"UPLOAD THE TEXT: -",

font*=*"comicsansms 12 ", padx*=*5, pady*=*20)

search\_text.grid(row*=*2, column*=*0)

public *=* Label(root, text*=*"Enter The Public Key: -",

font*=*"comicsansms 12 ", padx*=*5, pady*=*20)

public.grid(row*=*3, column*=*0)

part1entry *=* IntVar()

part1 *=* Entry(root, width*=*10, relief*=*SUNKEN,

borderwidth*=*3, textvariable*=*part1entry)

part1.grid(row*=*3, column*=*1)

part2entry *=* IntVar()

part2 *=* Entry(root, width*=*10, relief*=*SUNKEN,

borderwidth*=*3, textvariable*=*part2entry)

part2.grid(row*=*3, column*=*2)

upload\_btn *=* PhotoImage(file*=*"upload.png")

song\_btn *=* Button(root, image*=*upload\_btn, command*=*open\_song, borderwidth*=*0)

song\_btn.grid(row*=*1, column*=*2)

text\_btn *=* Button(root, image*=*upload\_btn, command*=*open\_text, borderwidth*=*0)

text\_btn.grid(row*=*2, column*=*2)

launch\_img *=* PhotoImage(file*=*"Launch.png")

launch *=* Button(root, image*=*launch\_img, command*=*LAUNCH, borderwidth*=*0)

launch.grid(row*=*4, column*=*3)

root.mainloop()

* **ReceiverGUI.py:** The receiver’s GUI is made here.

*from* extract *import* *\**

*from* tkinter *import* *\**

*from* tkinter *import* font

*from* tkinter *import* filedialog

*from* tkinter *import* messagebox

root *=* Tk()

root.geometry("750x350")

root.title("Personal Information as Cipher Hidden in Audio (PICHA)")

root.iconbitmap("Icon.ico")

*def* *LAUNCH*():

d *=* part1entry.get()

n *=* part2entry.get()

extraction(d, n)

messagebox.showinfo("Status", "DONE\nCheck The Parent Directory")

*def* *open\_song*():

*global* song\_path

song\_path *=* filedialog.askopenfilename(

initialdir*=*"C:/Users/Aman Pandey/OneDrive/Desktop/PICHA")

*return* song\_path

heading *=* Label(text*=*"Steganographic Extraction", font*=*"algerian 16 ")

heading.grid(row*=*0, column*=*3)

song\_text *=* Label(root, text*=*"UPLOAD THE SONG: -",

font*=*"comicsansms 12", padx*=*5, pady*=*20)

song\_text.grid(row*=*1, column*=*0)

Private *=* Label(root, text*=*"Enter The Private Key: -",

font*=*"comicsansms 12 ", padx*=*5, pady*=*20)

Private.grid(row*=*2, column*=*0)

part1entry *=* IntVar()

part1 *=* Entry(root, width*=*10, relief*=*SUNKEN,

borderwidth*=*3, textvariable*=*part1entry)

part1.grid(row*=*2, column*=*1)

part2entry *=* IntVar()

part2 *=* Entry(root, width*=*10, relief*=*SUNKEN,

borderwidth*=*3, textvariable*=*part2entry)

part2.grid(row*=*2, column*=*2)

upload\_btn *=* PhotoImage(file*=*"upload.png")

song\_btn *=* Button(root, image*=*upload\_btn, command*=*open\_song, borderwidth*=*0)

song\_btn.grid(row*=*1, column*=*2)

launch\_img *=* PhotoImage(file*=*"Launch.png")

launch *=* Button(root, image*=*launch\_img, command*=*LAUNCH, borderwidth*=*0)

launch.grid(row*=*4, column*=*3)

root.mainloop()

**WORKING LAYOUT: -**

Some snippets of the working layout are attached below.

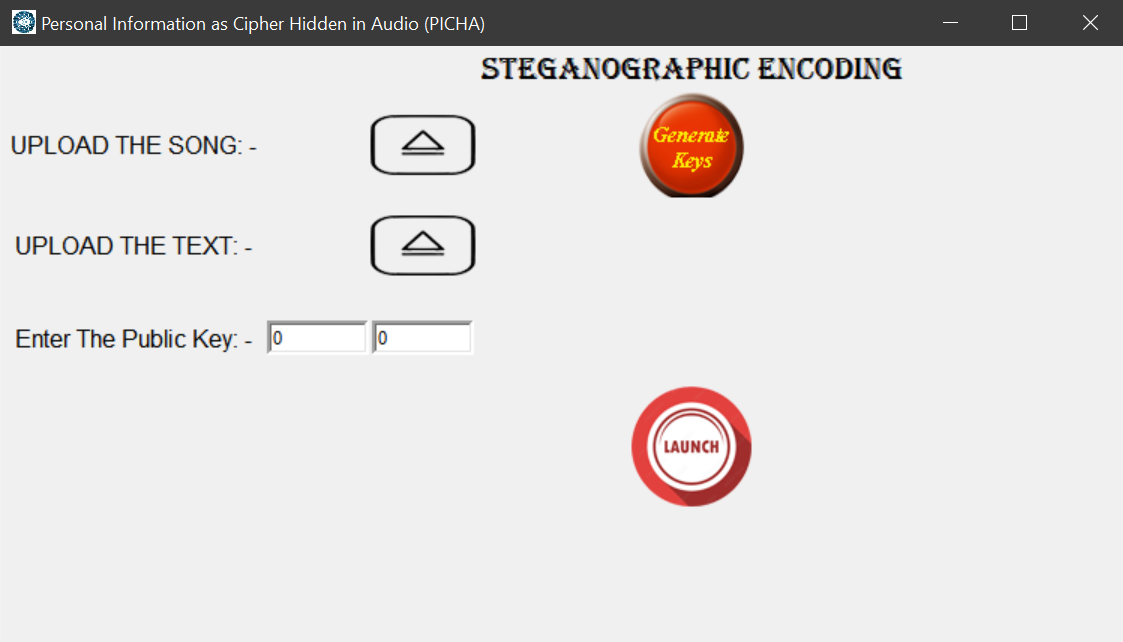


Figure 5.1

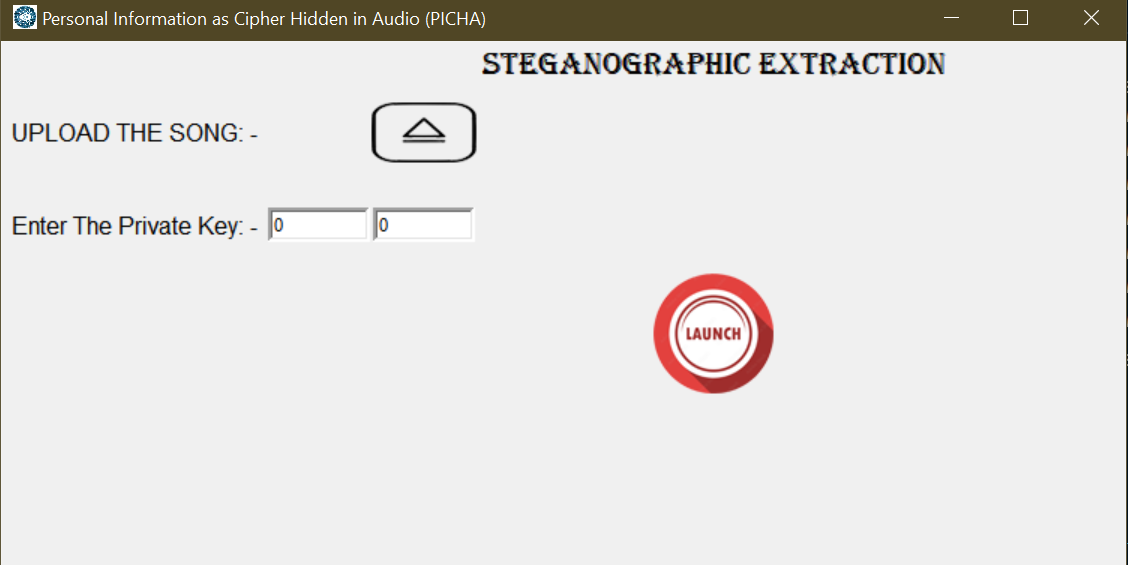


Figure 5.2

**PROTOTYPE SUBMISSION: -**

For the prototype submission tutorial please click [HERE](https://drive.google.com/file/d/1VQJUXzAdfgVz3rhheH0Zl2v9aYfhQIam/view?usp=sharing). This will show you a demo video for how this tool is to be used for sending as well as receiving. It also shows that the data loss is very less and no changes in the quality of the sound are observed when heard.

**TEST AND VALIDATION: -**

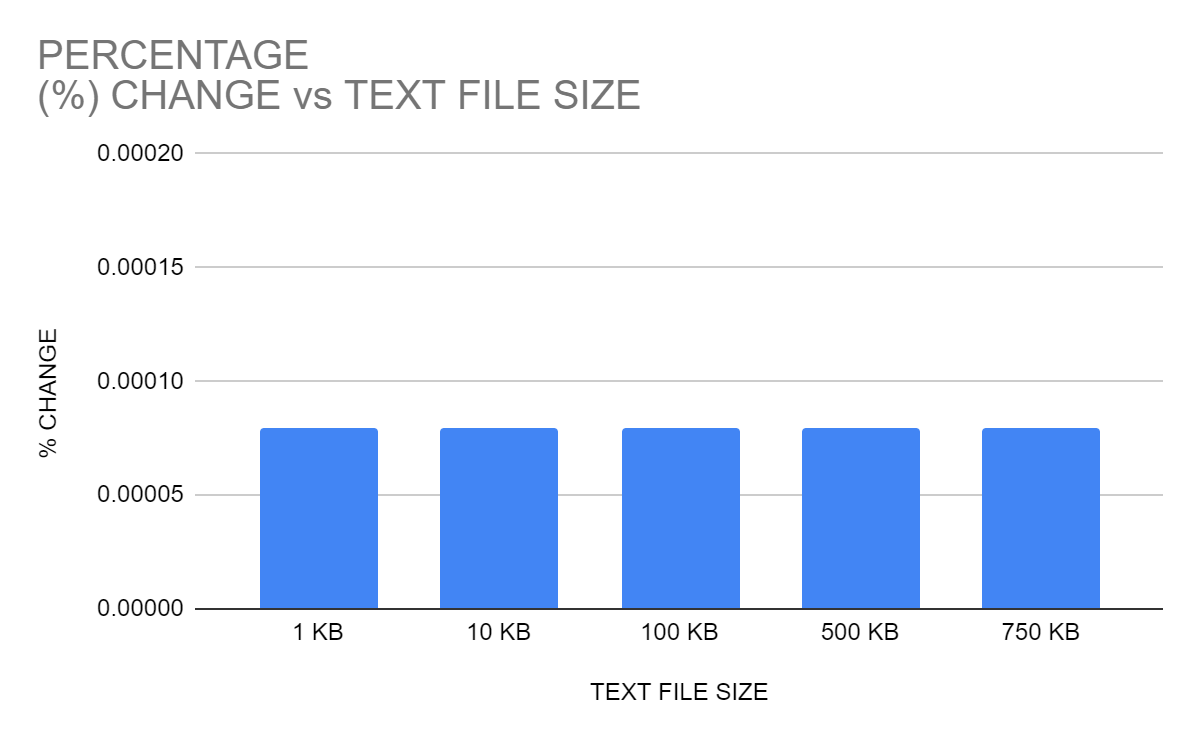
This tool has been tested in various conditions. It is tested using different alphabets and numbers and found them all working except when ‘|’ is used (which we are considering as a dummy symbol). It also executes successfully if the location of the text file and the song is changed. It runs as an actual open command without any complications. We have also tested it by taking different sizes of text files and the results are shown below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S. No.** | **TEXT FILE SIZE** | **ORIGINAL AUDIO SIZE (bytes)** | **STEGANOGRAPHIC AUDIO**  **SIZE (bytes)** | **PERCENTAGE**  **(%) CHANGE** |
| **1** | 1 KB | 4,28,17,614 | 4,28,17,580 | 0.00007940657 |
| **2** | 10 KB | 4,28,17,614 | 4,28,17,580 | 0.00007940657 |
| **3** | 100 KB | 4,28,17,614 | 4,28,17,580 | 0.00007940657 |
| **4** | 500 KB | 4,28,17,614 | 4,28,17,580 | 0.00007940657 |
| **5** | 750 KB | 4,28,17,614 | 4,28,17,580 | 0.00007940657 |

Table 5.1

**PERFORMANCE ANALYSIS: -**

The graph below shows the relationship between the text file size and the percentage change in the size of the audio file.

****

Graph 5.1

**SUMMARY: -**

To summarize and draw the inference from all the given results, it can be concluded that the tool works pretty fine with very small (almost of the order of **10-5**). This insignificant change is very crucial for the security of the file. Moreover, we can also judge that the maximum size of the text file which we can encode in a 4,28,17,614-byte file is 750 kb. Any more than that would create problems. It also indicates that the change in the steganographic audio is independent of the file size. This is because of the addition of the dummy symbol ‘|’.

**PROJECT OUTCOME AND APPLICABILITY**

**OUTLINE: -**

The main crux of the project is to get to know more about steganography or hiding the secret data in various forms like image, text, audio, video, network steganography. Cryptography is added to spice things up, making the data unable to be used if found. We were required to strengthen the security of confidentiality and classified data. The audio file generated by P.I.C.H.A would-be used as a medium of transport or simply to be kept in the system. Applications of this would be in various fields because of its flexibility and robustness.

**KEY IMPLEMENTATIONS OUTLINES OF THE SYSTEM: -**

Firstly, there should be a secret message text file in .txt form and an audio file in which hides the secret message. Most importantly we need Visual Studio Code software to run the Python code. There is a code for RSA encryption of secret messages. It will encrypt the secret message. Then we will get ciphertext. After encryption, the next step is to hide the encrypted secret message. For hiding, the code of audio steganography will hide the secret message in audio and give output to an audio file.

For extraction of encrypted messages, there is a code to extract the encrypted message. After encryption, it will pass through the RSA decryption. After decryption, our secret message will be obtained.

**SIGNIFICANT PROJECT OUTCOMES: -**

The prototype was required to have two outcomes - an audio steganography file containing the text and a text file containing the plaintext. As discussed previously, we were successful in creating both types of files. Firstly, we created a file encoding the encrypted message of the plaintext in an audio file thus accomplishing the first result. This would make the data secure as well as encrypted thus merging the essence of cryptography and steganography beautifully.

Data which cannot be seen is as good as nothing, so we must extract the data and get it back to make our prototype a success. We accomplished that goal too by successfully getting the data in a new text file without any data loss. Thus, the objective of this prototype was successfully fulfilled.

**PROJECT APPLICABILITY ON REAL-WORLD APPLICATIONS: -**

PICHA can also be applied to real life and can be executed in one’s and play a major role. Government official’s data can be secured and protected through this means, and one of the most important responsibilities of the government is to secure the military information of the country because it's quite confidential and important, containing a lot of secrets about arms and ammunition, previous attack records and wars. To protect that data so that it cannot be breached PICHA can show its ability to protect the information. Also, Big Business Tycoons can hide their strategy and information via using PICHA as steganography dramatically decreases the chance of information leaking. In the musical industry, this methodology can be implemented and digital signatures can be implanted into the tune of music thus preventing piracy and supporting the legal recognition of that music.

**INFERENCE: -**

From the above points, it should have become clear that this method is a preventive measure. It is good if the data is of high importance and value. Moreover, it should be noted that normal people are not the audience of this tool. It could be pretty inconvenient for a person to just lock a random text file into a song and then get it back. It might be fun once or twice but at the end of the day, it is simply idiotic and useless.

**CONCLUSION AND RECOMMENDATION**

**LIMITATIONS AND CONSTRAINTS: -**

Although this prototype is pretty smooth and simple, it has its own set of shortcomings. First and foremost is its GUI. Although the GUI looks simple, it gives the vibe that it has been designed by a newbie, which is true. The GUI is pretty simple, containing only a few buttons and many things which make a professional GUI application missing.

The message which would be encrypted inside the audio file is filled up to the brim with the help of ‘|’ symbols. For example: if your message is “Johnny will become the Grindelwald” then the message that would be encoded is “Johnny will become the Grindelwald|||||||||||| (..... up to a maximum number of characters)”. This would waste space and make the file size constant with constant percentage change.

The keys generated by the key generator button cannot be copied and pasted. It can only be typed manually making the process hectic.

The value of the key is small which is kept for simplicity. This would be a serious compromise with the security of the data. A highly sensitive data must have A very large key for its encryption and decryption.

It requires Python 3 and above which can be hectic. It would be a bitter experience to the non-programmer audience albeit being a non-programmer-friendly GUI.

**FUTURE ENHANCEMENT: -**

The first thing which we would like to enhance would be the GUI. We need to make it look a bit better so that it can be up to the standard. It does not have a menu bar or a toolbar. No inbuilt audio checker. No inbuilt size comparison system. Two GUIs for a single application and the look and the look and graphics of the application were barely satisfactory. Changes are thus required in this aspect.

The second is the encryption strength. The strength of the key when we talk about securing highly confidential data is not up to the standards. There is also the problem with the computational ability of our laptops being a restraint but if there is a mainframe with more computational power then greater security measures can be deployed.

The option of Security vs Capacity should be provided. It is of utmost importance because in that way the best of the best services should be provided up to their requirements. There is a tradeoff between security and capacity. A person should not expect a high level of security and capacity in the same audio file simultaneously. Accordingly, 1-LSB, 2-LSB, 3-LSB audio steganography methods should be implemented.

The application should be in excel format so that it would be independent of the installation of python 3 and any IDE which would just be a nuisance to a non-programmer.

**INFERENCE: -**

This universe does not allow anything to be perfect. The beauty of the imperfect is also very beautiful. To a system, pros and cons are always there. It would be the understanding of us that can minimize the cons and use it for our benefit. The pros should also be utilized more efficiently. Only in this way can the potential of a product be pushed to the maximum.

**GLOSSARY**

**Audio Steganography: -** Audio steganography is about hiding the secret message in the audio. It is a technique used to secure the transmission of secret information or hide its existence. It also may provide confidentiality to secret messages if the message is encrypted.

**Encryption: -** The process of converting information or data into a code, especially to prevent unauthorized access.

**Decryption: -** The conversion of encrypted data into its original form is called Decryption. It is generally a reverse process of encryption. It decodes the encrypted information so that an authorized user can only decrypt the data because decryption requires a secret key or password.

**Key**: - In cryptography, a key is a string of characters used within an encryption algorithm for altering data so that it appears random

**RSA Algorithm (in Cryptography): -** RSA algorithm is an asymmetric cryptography algorithm. Asymmetric means that it works on two different keys i.e., Public Key and Private Key. As the name describes, the Public Key is given to everyone and the Private key is kept private.

**Steganography: -** the practice of concealing messages or information within other non-secret text or data.

**GUI (Graphical User Interface):-** The graphical user interface is a form of user interface that allows users to interact with electronic devices through graphical icons and audio indicators such as primary notation, instead of text-based user interfaces, typed command labels, or text navigation.

**LSB(Least Significant Bit):-** In computing, the least significant bit (LSB) is the bit position in a binary integer representing the binary 1st place of the integer.

**REFERENCES**

1. N. Al-Junaid and G. Adnan, "Combining RSA and audio steganography on personal computers for enhancing security.," vol. SN Applied Sciences 1, no. 8, pp. 1-11, 2019.
2. F. Djebbar, A. Baghdad, M. A. Karim, and H. Habib, "Comparative study of digital audio steganography techniques.," *EURASIP Journal on Audio, Speech, and Music Processing,* vol. 1, p. 16, 2012.
3. A. Gambhir and S. Khara, "Integrating RSA cryptography & audio steganography.," *International Conference on Computing, Communication and Automation (ICCCA),* pp. 481-484, 2016.