SECURE DATA HIDING IN IMAGES USING STEGANOGRAPHY

Presented By: Shilpa Pagadala
College Name: Megha Institute of Engineering and Technology for Women
Department : BTech (Computer Science and Engineering)

edunet
foundation

OUTLINE

- Problem Statement
- Technology used
- Wow factor
- End users
- Result
- Conclusion
- Git-hub Link
- Future scope

PROBLEM STATEMENT

- In the digital age, safeguarding sensitive information from unauthorized access is paramount. Traditional encryption can attract unwanted attention, making hidden messages susceptible to interception. This project aims to develop a steganography-based secure data hiding system that embeds secret messages within images without visibly altering them.
- By utilizing the Least Significant Bit (LSB) technique, the system keeps confidential data undetectable while preserving the image's original quality. This method offers a covert communication channel for secure information transfer, making it perfect for cybersecurity, watermarking, and confidential messaging applications

TECHNOLOGY USED

- **Python** – The core programming language used.
- **Tkinter** – Used for the graphical user interface (GUI).
- **PIL (Pillow)** – Used for image processing (opening, modifying, and saving images).
- **Bit Manipulation** - Encoding and decoding the hidden message using binary operations.

SYSTEM REQUIREMENTS:

- **Operating System:** Windows, macOS, or Linux
- **Python Version:** Python 3.xRequired
- **RAM:** At least 2GB RAM, but 4GB or more is recommended for better performance.

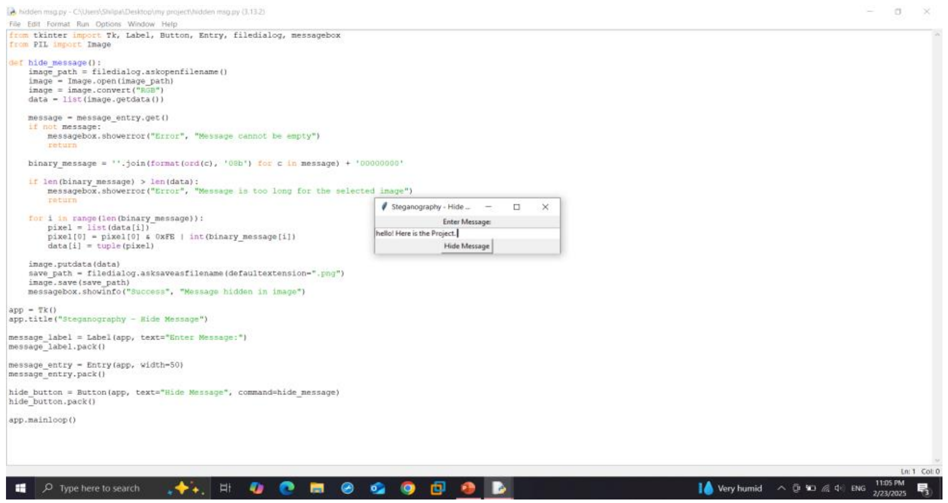
WOW FACTORS

- Uses LSB (Least Significant Bit) Steganography to hide and retrieve messages in images
- Simple and easy-to-use GUI interface.
- Works without requiring complex encryption methods
- Simple yet effective data hiding technique.
- Completely invisible to the human eye.

END USERS

- Individuals looking for secure communication
- Journalists and activists who need to hide sensitive messages.
- Developers interested in steganography
- Cybersecurity enthusiasts exploring data hiding techniques.

RESULTS



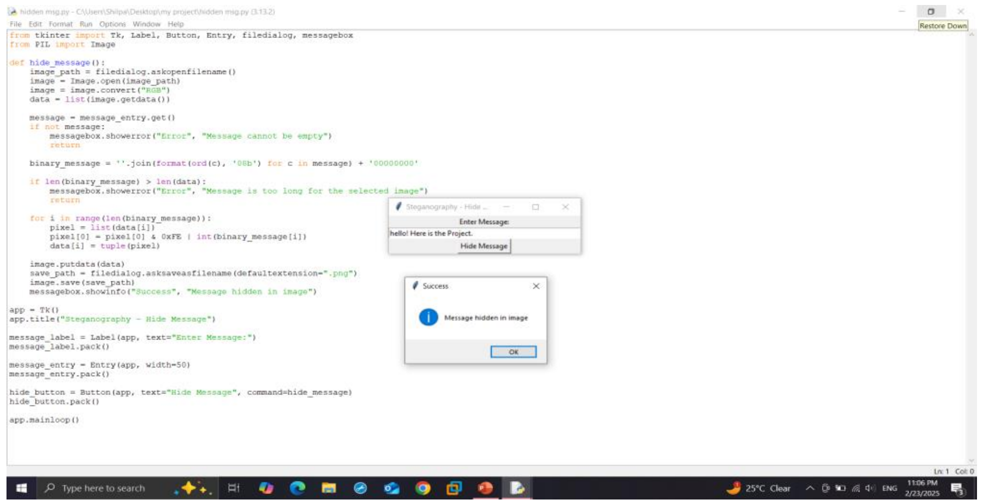
hidden msg.py - C:\Users\Shilpa\Desktop\my project\hidden msg.py (3.13.2)

```python
from tkinter import Tk, Label, Button, Entry, filedialog, messagebox
from PIL import Image

def hide_message():
    image_path = filedialog.askopenfilename()
    image = Image.open(image_path)
    image = image.convert("RGB")
    data = list(image.getdata())

    message = message_entry.get()
    if not message:
        messagebox.showerror("Error", "Message cannot be empty")
        return

    binary_message = ''.join(format(ord(c), '08b') for c in message) + '00000000'

    if len(binary_message) > len(data):
        messagebox.showerror("Error", "Message is too long for the selected image")
        return

    for i in range(len(binary_message)):
        pixel = list(data[i])
        pixel[0] = pixel[0] & 0xFE | int(binary_message[i])
        data[i] = tuple(pixel)

    image.putdata(data)
    save_path = filedialog.asksaveasfilename(defaultextension=".png")
    image.save(save_path)
    messagebox.showinfo("Success", "Message hidden in image")

app = Tk()
app.title("Steganography - Hide Message")

message_label = Label(app, text="Enter Message:")
message_label.pack()

message_entry = Entry(app, width=50)
message_entry.pack()

hide_button = Button(app, text="Hide Message", command=hide_message)
hide_button.pack()

app.mainloop()
```

Steganography - Hide ...

Enter Message:

hello! Here is the Project.



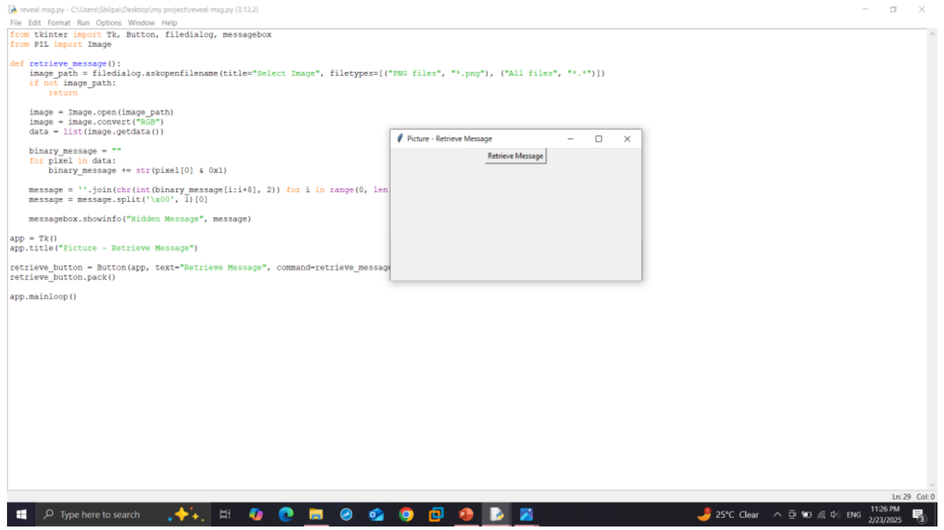
hidden msg.py - C:\Users\Shilpa\Desktop\my project\hidden msg.py (3.13.2)

```python
from tkinter import Tk, Label, Button, Entry, filedialog, messagebox
from PIL import Image

def hide_message():
    image_path = filedialog.askopenfilename()
    image = Image.open(image_path)
    image = image.convert("RGB")
    data = list(image.getdata())

    message = message_entry.get()
    if not message:
        messagebox.showerror("Error", "Message cannot be empty")
        return

    binary_message = ''.join(format(ord(c), '08b') for c in message) + '00000000'

    if len(binary_message) > len(data):
        messagebox.showerror("Error", "Message is too long for the selected image")
        return

    for i in range(len(binary_message)):
        pixel = list(data[i])
        pixel[0] = pixel[0] & 0xFE | int(binary_message[i])
        data[i] = tuple(pixel)

    image.putdata(data)
    save_path = filedialog.asksaveasfilename(defaultextension=".png")
    image.save(save_path)
    messagebox.showinfo("Success", "Message hidden in image")

app = Tk()
app.title("Steganography - Hide Message")

message_label = Label(app, text="Enter Message:")
message_label.pack()

message_entry = Entry(app, width=50)
message_entry.pack()

hide_button = Button(app, text="Hide Message", command=hide_message)
hide_button.pack()

app.mainloop()
```

hello! Here is the Project.

Success

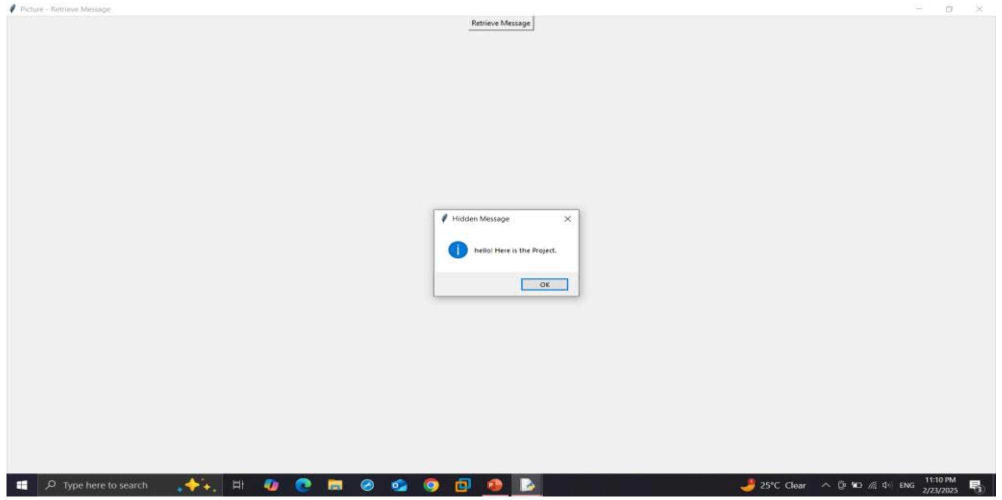
Message hidden in image



reveal msg.py - C:\Users\Shilpa\Desktop\my project\reveal msg.py (3.13.2)

```python
from tkinter import Tk, Button, filedialog, messagebox
from PIL import Image

def retrieve_message():
    image_path = filedialog.askopenfilename(title="Select Image", filetypes=[("PNG files", "*.png"), ("All files", "*.*")])
    if not image_path:
        return

    image = Image.open(image_path)
    image = image.convert("RGB")
    data = list(image.getdata())

    binary_message = ""
    for pixel in data:
        binary_message += str(pixel[0] & 0x1)

    message = ''.join(chr(int(binary_message[i:i+8], 2)) for i in range(0, len
    message = message.split('\x00', 1)[0]

    messagebox.showinfo("Hidden Message", message)

app = Tk()
app.title("Picture - Retrieve Message")

retrieve_button = Button(app, text="Retrieve Message", command=retrieve_messag
retrieve_button.pack()

app.mainloop()
```

Picture - Retrieve Message



Picture - Retrieve Message

Hidden Message

hello! Here is the Project.

CONCLUSION

- This Project delivered a straightforward, Python-based steganography application with a very robust graphical interface. It simplifies the process of embedding and extracting concealed information in images, offering possibilities for communication and content protection.
- This work provides a stepping stone for enhancing confidentiality and data safety in modern networks.

GITHUB LINK

- https://github.com/shilpa724/-Steganography-Project.git

FUTURE SCOPE

- **Enhanced Security & Encryption:** Strengthen encryption and improve resistance against steganalysis and cyber threats.
- **Support for More File Formats:** Extend steganography to videos, audio, and text for secure communication across various media.
- **Real-time & Cloud Integration:** Develop mobile/web applications for instant secure messaging and explore blockchain for decentralized data hiding.
- **Advanced AI-Powered Techniques:** Implement AI for intelligent embedding, detection avoidance, and adaptability against security threats.

THANK YOU

edunet foundation