T.C.
KIRIKKALE ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

PROJE 2

GÖRÜNTÜ ÜZERİNE METİN GİZLEME VE ŞİFRELEME

Hazırlayan
Süleyman Arif Ersoy
200205050

Öğretim Üyesi
Dr. Öğr. Üyesi Fahrettin HORASAN

2024
KIRIKKALE

Özet

Bu projede, görüntüler üzerinde metin gizleme ve şifreleme yöntemleri araştırılmış ve bir uygulama geliştirilmiştir. Geliştirilen uygulama, kullanıcının belirttiği bir metni Sezar şifreleme algoritması ile şifreleyip, LSB (En Az Önemli Bit) steganografi tekniği kullanarak bir görüntüye gizlemekte ve ardından bu metni geri çıkararak çözmektedir. Projenin ana amacı, dijital veri gizliliğini artırmak ve bilgi güvenliğini sağlamaktır.

1. GİRİŞ

1.1 Proje Tanımı

Görsel üzerinde metin gizleme ve şifreleme, gizli bilgilerin güvenli bir şekilde iletilmesini sağlamak için kullanılan tekniklerden biridir. Bu projede, Sezar şifreleme ve LSB steganografi yöntemlerini birleştirerek bir uygulama geliştirilmiştir.

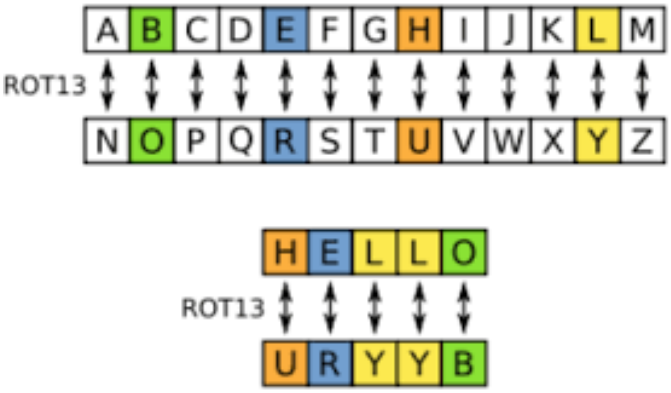
1.2 Amaç ve Hedefler

- Dijital ortamda veri gizliliğini sağlamak.
- Kullanıcı dostu bir arayüz geliştirmek.
- Görüntüler üzerinde metin gizleme işlemini etkin bir şekilde gerçekleştirmek.
- Görsel kaliteyi koruyarak veri gizleme yöntemlerini test etmek.

2. METOTLAR

2.1 Sezar Şifreleme

Sezar şifreleme, basit bir şifreleme yöntemidir. Bu yöntemde her bir harf, belirli bir kaydırma miktarı kullanılarak şifrelenir. Örneğin, bir kaydırma miktarı olan 3 kullanıldığında, A harfi D olur.

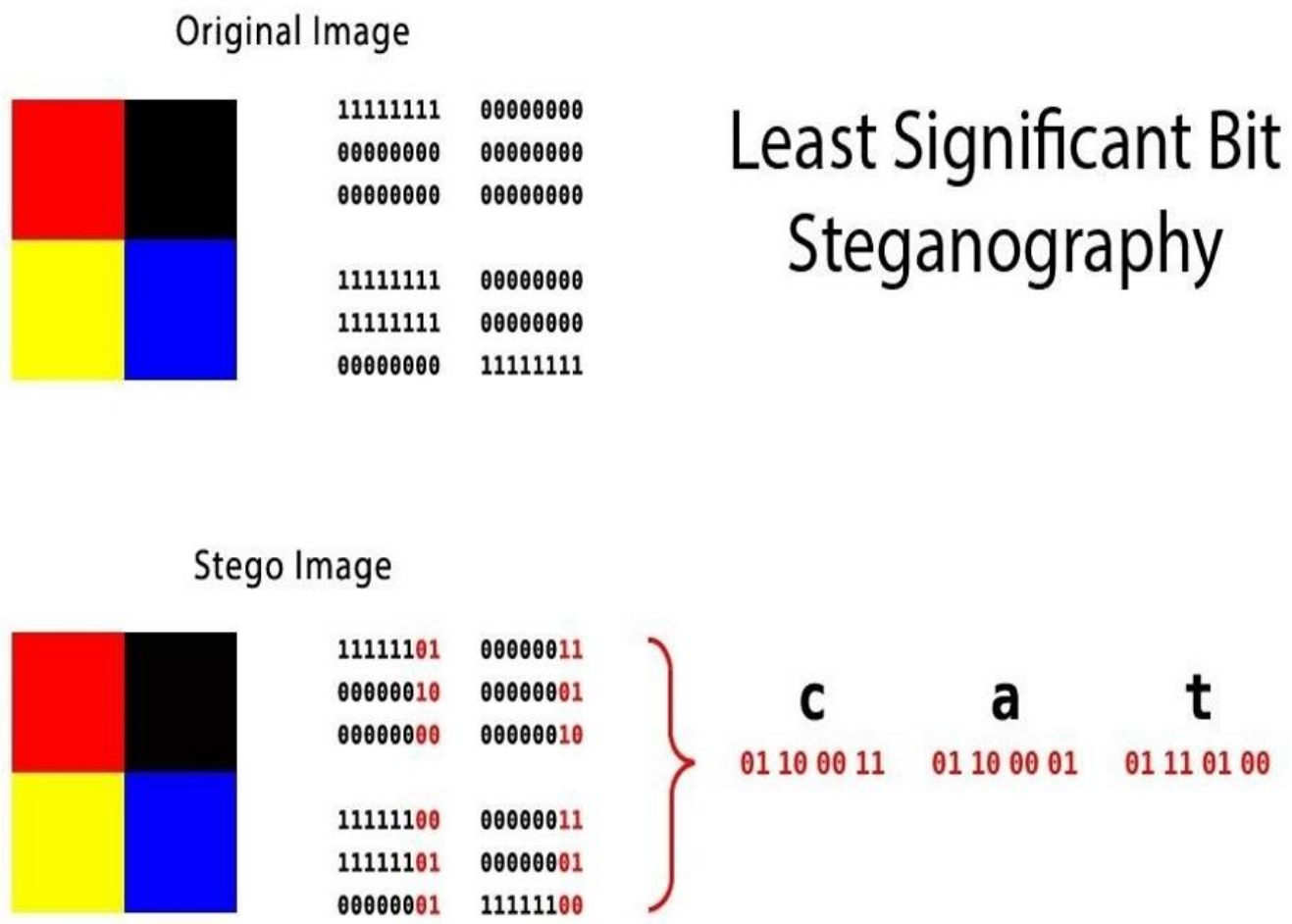


Şekil 1

Şekil 1 ‘deki görselde sezar şifreleme algoritmasından bir örnek verilmiştir. Shift sayısı 13 olarak seçilmiştir ve A harfinin geçtiği yerlerde N olarak alınacak demek oluyor. HELLO kelimesini teker teker harflerini şifreleme algoritmasına soktuğumuz zaman URYYB kelimesi olarak şifrelenmiş halini almış oldu.

2.2 LSB Steganografi

LSB steganografi, bir görüntünün en az önemli bitlerini değiştirerek veri gizlemeye yarayan bir tekniktir. Bu yöntem, gizli veriyi görüntünün görsel kalitesini minimum seviyede etkileyerek gizler. Artık Cesar şifrelemesi algoritmasında çıkmış olan metnimizi LSB yöntemi kullanarak görüntü üzerindeki 3 kanallı (R, G, B) yapısı bulanan pikseller içerisinde bulunan 8 bitlik yapının içerisinde en son bit olarak adlandırılan yerleştirilecektir. Bu yerleştirme işlemi her kelimenin her harfi teker teker 2 bitlik binary ifadesi alınacak ve görüntü pikselleri karıştırılarak bir diziye aktarılacak ve bit değerleri sırasıyla piksellerin son bit değerine yerleştirilecektir.



Şekil 2

LSB yöntemini Şekil 2'deki bir görseldeki örnek ile açıklayacak olursak, görsel üzerindeki piksellerin (R, G, B) değerlerini yazacak olursak;

Kırmızı-> R-> 11111111 G->00000000 B->00000000

Siyah-> R-> 00000000 G->00000000 B->00000000

Sarı-> R-> 11111111 G->11111111 B->00000000

Mavi-> R-> 00000000 G->00000000 B->11111111

Bit değerleri bu şekildedir. Bu görüntü üzerine "cat" kelimesini gizleyecek olursak ilk yapacağımız işlem "cat" kelimesinin her harfinin ASCII kodunu aldıktan sonra 8 bitlik binary formata dönüştürülecektir daha sonrasında görüntüdeki piksellerin R G B bitlerinden en küçük olan bit seviyesine sırasıyla yerleştirme işlemi yapılacaktır.

2.3 PSNR (Peak Signal-to-Noise Ratio)

PSNR, orijinal ve değiştirilen görüntü arasındaki farkı ölçmek için kullanılan bir metriktir. Yüksek PSNR değeri, iki görüntü arasındaki farkın az olduğunu gösterir.

3. Yöntem

3.1 Geliştirilen Uygulama

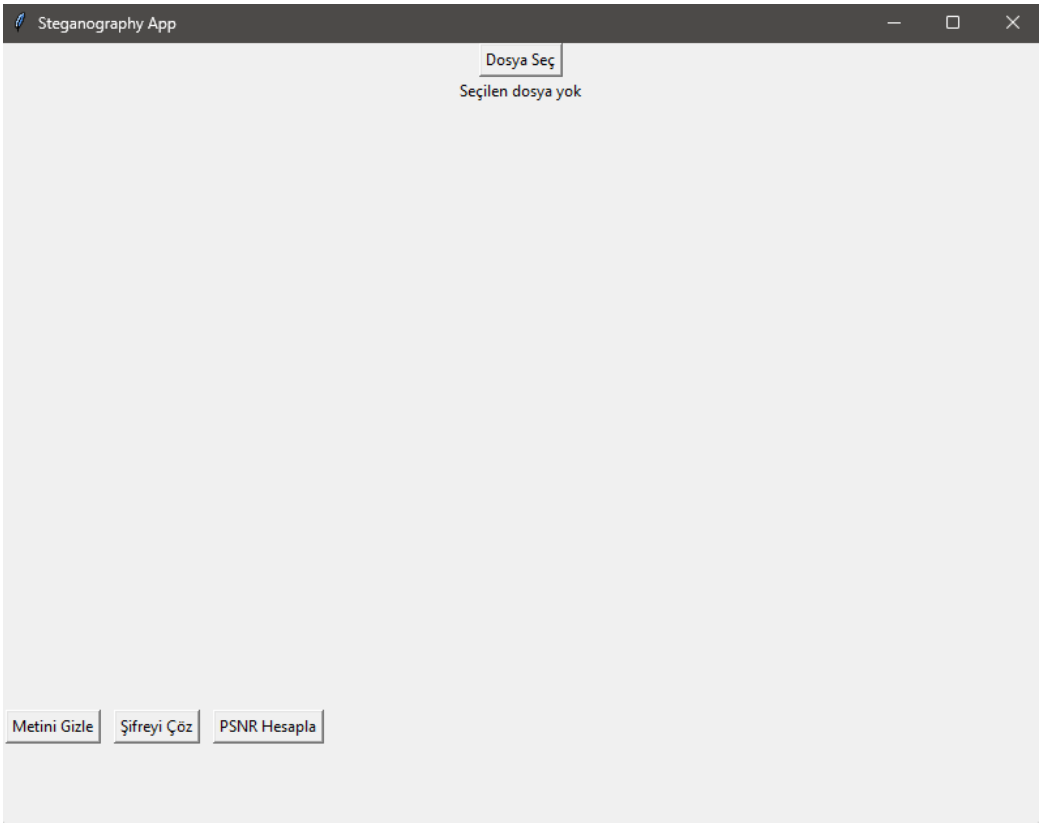
Uygulama, kullanıcı dostu bir arayüz kullanarak metin şifreleme ve gizleme işlemlerini gerçekleştirir. Uygulamanın işleyişi şu adımlardan oluşur:

1. Kullanıcı, bir görüntü dosyası seçer.
2. Metin girişi yapar ve bu metni Sezar algoritması ile şifreler.
3. Şifrelenmiş metin, LSB steganografi kullanılarak seçilen görüntüye gömülür.
4. Görüntü dosyası kaydedilir.
5. Gizli metin, görüntüden çıkarılıp Sezar algoritması ile çözülür ve orijinal metin elde edilir.

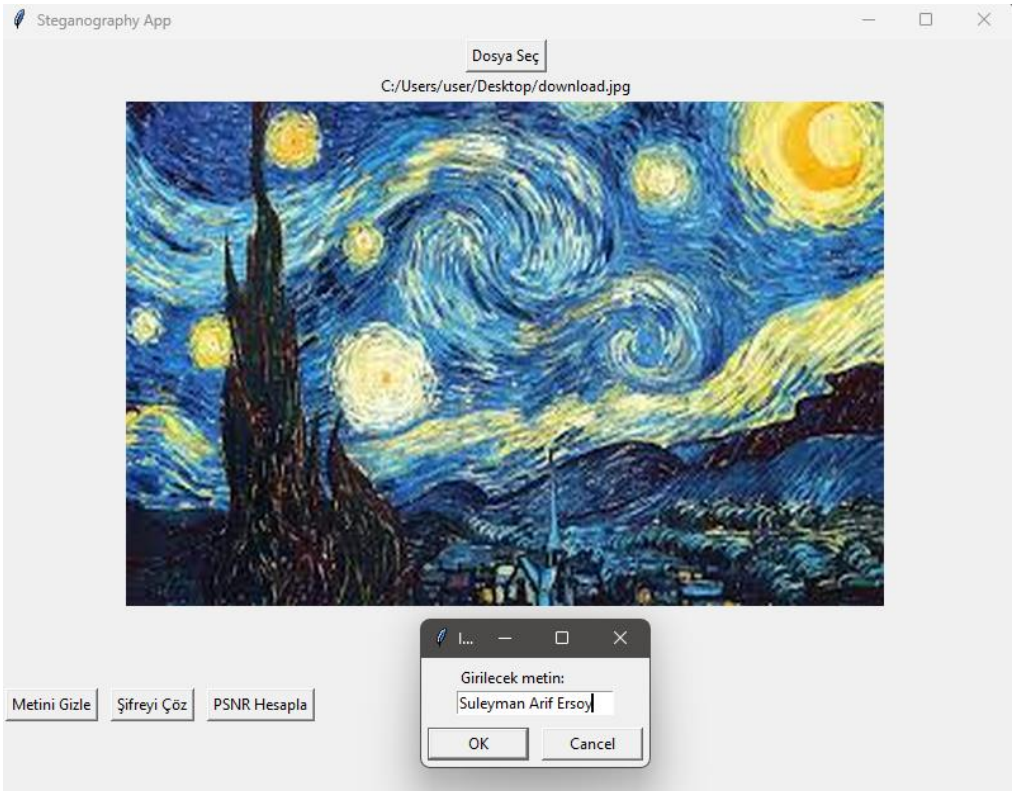
4. Sonuçlar ve Değerlendirme

4.1 Test Sonuçları

Uygulamanın testleri çeşitli görüntüler ve metinler kullanılarak gerçekleştirilmiştir. Test sonuçları, gizli metnin başarıyla gömüldüğünü ve geri çıkarıldığını göstermiştir. Ayrıca, PSNR değerleri ölçülmüş ve görüntülerin kalite kaybı minimum seviyede kalmıştır.

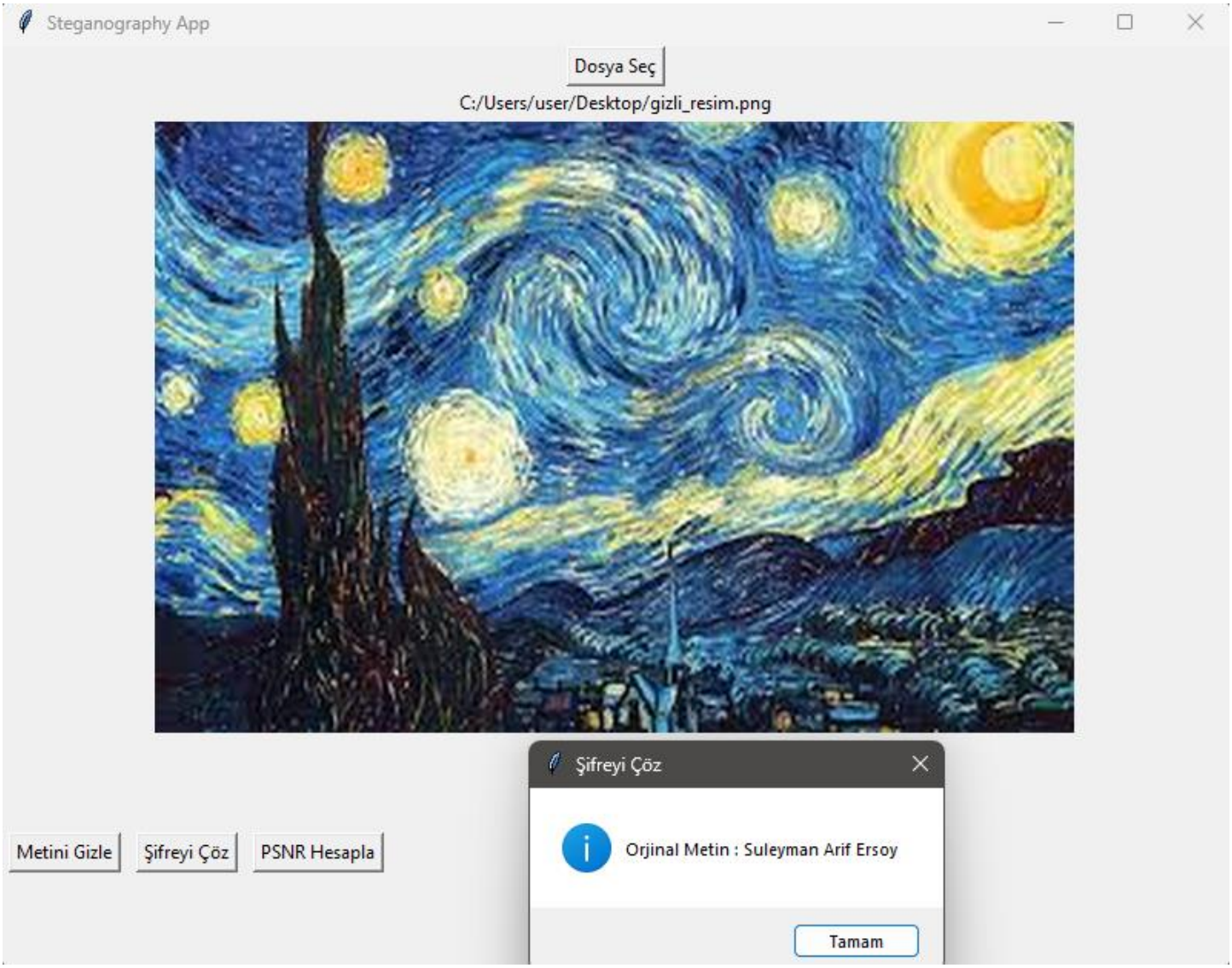


Şekil 3



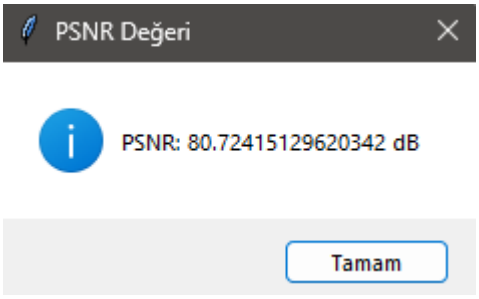
Şekil 4

Şekil 4'te dosya seç butonuna tıklanılarak bilgisayardan resim seçilme işlemi yapılmıştır. Resim seçildikten sonra Metin Gizle butonuna basıldıktan sonra karşımıza gelen pop-up sayfasında gizlenecek metnin girilmesi isteniliyor.



Şekil 5

Şekil 5'teki görselde Metni Gizle butonunun işlemi bittikten sonra gizleme işlemi yapılan resim, bilgisayara kullanıcının istediği yere kaydediliyor. Kaydedilen resim dosya seç butonundan tekrar açılıyor ve şifreyi çöz butonuna basılarak orijinal metni pop-up penceresinde karşımıza getiriyor.



Şekil 6

Şekil 6'da gizleme işlemi yapılan görsel ile orijinal görsel karşılaştırılarak PSNR performans metriği kullanılarak performans ölçümü yapılmıştır. 600x344 bir görsel kullanılmıştır.

5. Sonuç ve Öneriler

5.1 Sonuç

Bu projede geliştirilen uygulama, metin gizleme ve şifreleme yöntemlerini birleştirerek güvenli veri iletimini sağlamıştır. Kullanıcı dostu bir arayüz ile bu işlemler kolayca gerçekleştirilmiştir. Yapılan PSNR perfomans metrikleri ölçütlerinde başka görseller de denenmiştir. PSNR sonucu 80'nin altına düştüğü görülmemiştir. İncelenen diğer çalışmalardan 80 değerinin güzel bir değer olduğu görülmüştür.

5.2 Öneriler

Gelecek çalışmalarda, farklı şifreleme yöntemleri ve steganografi tekniklerinin entegre edilmesi önerilebilir. Ayrıca, daha yüksek çözünürlüklü görüntüler üzerinde testler yapılabilir. Büyük bir metin girildiğinde görsel üzerinde saklanması zor olabilir. Tüm piksellerdeki kanallarda son bite yerleştirme işlemi yapıldıysa artık bir sonraki piksel yani 7.bite saklama işlemi yapılması önerilir. Böylece metnin büyüklüğüne bağlı olarak da performans ölçütümüzde azalmış olacak.

6. Kodlar

Cesar Şifreleme

```python
# Sezar Şifreleme Fonksiyonu
def caesar_cipher(text, shift):
    result = ""
    for i in range(len(text)):
        char = text[i]
        if char.isupper():
            result += chr((ord(char) + shift - 65) % 26 + 65)
        elif char.islower():
            result += chr((ord(char) + shift - 97) % 26 + 97)
        else:
            result += char
    return result

# Sezar Şifre Çözme Fonksiyonu
def caesar_decipher(text, shift):
    return caesar_cipher(text, -shift)
```

Şekil 7

LSB Yöntemi ile Metin Gizleme

```python
# LSB Steganografi ile metin gizleme fonksiyonu
def metin_gomme(image, text,seed=42):
    img = np.array(image)
    # Metni binary formata dönüştür
    binary_text = ''.join(format(ord(char), '08b') for char in text)
    # Stop marker ekle
    stop_marker = '1111111111111111'
    binary_text += stop_marker
    data_len = len(binary_text)
    # Görsel boyutları
    height, width, _ = img.shape
    # Piksel koordinatlarını oluştur
    pixel_indices = [(i, j, k) for i in range(height) for j in range(width) for k in range(3)]
    # Rastgele karıştırmak için seed belirleyin (opsiyonel)
    if seed is not None:
        random.seed(seed)
    # Piksel indekslerini rastgele karıştır
    random.shuffle(pixel_indices)
    # Binary metni rastgele piksellere yerleştir
    data_index = 0
    for i, j, k in pixel_indices:
        if data_index < data_len:
            img[i][j][k] = int(format(img[i][j][k], "08b")[:-1] + binary_text[data_index], 2)
            data_index += 1
        if data_index >= data_len:
            break
    # Yeni görüntüyü döndür
    new_image = Image.fromarray(img)
    return new_image
```

Şekil 8

Metni Açığa Çıkarma

```python
# LSB Steganografi ile metni açığa çıkarma fonksiyonu
def extract_text(image,seed=42):
    img = np.array(image)
    binary_text = ""
    stop_marker = '1111111111111111'
    stop_marker_len = len(stop_marker)

    # Görsel boyutları
    height, width, _ = img.shape

    # Piksel koordinatlarını oluştur
    pixel_indices = [(i, j, k) for i in range(height) for j in range(width) for k in range(3)]

    # Rastgele karıştırmak için seed belirleyin (opsiyonel)
    if seed is not None:
        random.seed(seed)

    # Piksel indekslerini rastgele karıştır
    random.shuffle(pixel_indices)

    # Rastgele piksellerden metni çıkar
    for i, j, k in pixel_indices:
        binary_text += format(img[i][j][k], "08b")[-1]
        if binary_text[-stop_marker_len:] == stop_marker:
            binary_text = binary_text[:-stop_marker_len]
            text = ''.join([chr(int(binary_text[i:i+8], 2)) for i in range(0, len(binary_text), 8)])
            return text

    return ""
```

Şekil 9

PSNR Hesaplama

```python
# PSNR hesaplama fonksiyonu
def psnr(original_image, stego_image):
    original = np.array(original_image).astype(np.float64)
    stego = np.array(stego_image).astype(np.float64)
    mse = np.mean((original - stego) ** 2)
    if mse == 0:
        return 100
    max_pixel = 255.0
    psnr_value = 20 * np.log10(max_pixel / np.sqrt(mse))
    return psnr_value
```

Şekil 10

Arayüz

```python
class SteganographyApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Steganography App")
        self.root.geometry("800x600")

        self.filename = ""
        self.image_label = None

        self.create_widgets()

    def create_widgets(self):
        self.file_button = tk.Button(self.root, text="Dosya Seç", command=self.load_file)
        self.file_button.pack()

        self.filename_label = tk.Label(self.root, text="Seçilen dosya yok")
        self.filename_label.pack()

        self.image_canvas = tk.Canvas(self.root, width=600, height=400)
        self.image_canvas.pack()

        self.embed_button = tk.Button(self.root, text="Metini Gizle", command=self.prompt_for_text)
        self.embed_button.pack(side=tk.LEFT, padx=5, pady=5)

        self.extract_button = tk.Button(self.root, text="Şifreyi Çöz", command=self.extract_text)
        self.extract_button.pack(side=tk.LEFT, padx=5, pady=5)

        self.psnr_button = tk.Button(self.root, text="PSNR Hesapla", command=self.calculate_psnr)
        self.psnr_button.pack(side=tk.LEFT, padx=5, pady=5)
```

Şekil 11

```python
def load_file(self):
    self.filename = filedialog.askopenfilename(initialdir="/", title="Select file",
                                               filetypes=(("jpeg files", "*.jpg"), ("png files", "*.png")))
    if self.filename:
        self.filename_label.config(text=self.filename)
        self.image = Image.open(self.filename)
        self.display_image(self.image)

def display_image(self, img):
    img = img.resize((600, 400), Image.LANCZOS)
    img = ImageTk.PhotoImage(img)
    if self.image_label:
        self.image_canvas.delete(self.image_label)
    self.image_label = self.image_canvas.create_image(0, 0, anchor=tk.NW, image=img)
    self.image_canvas.image = img

def prompt_for_text(self):
    if not self.filename:
        messagebox.showerror("Hata", "Lütfen bir resim dosyası seçin ")
        return
    text = simpledialog.askstring("Input", "Girilecek metin:")
    if text:
        self.embed_text(text)
```

Şekil 12

```python
    def embed_text(self, text):
        shift = 3  # Sezar şifreleme için kaydırma değeri
        encrypted_text = caesar_cipher(text, shift)
        self.stego_image = metin_gomme(self.image, encrypted_text)
        self.display_image(self.stego_image)

        save_filename = filedialog.asksaveasfilename(defaultextension=".png", filetypes=[("PNG files", "*.png"), ("JPEG files", "*.jpg")])
        if save_filename:
            self.stego_image.save(save_filename)
            messagebox.showinfo("Başarili", f"Gizlenen resim şu isimle kaydedildi : {save_filename}")

    def extract_text(self):
        if not self.filename:
            messagebox.showerror("Hata", "Lütfen bir resim seçiniz")
            return

        try:
            encrypted_text = extract_text(self.image)
            shift = 3  # Sezar şifreleme için kullanılan kaydırma değeri
            original_text = caesar_decipher(encrypted_text, shift)
            messagebox.showinfo("Şifreyi Çöz", f"Orjinal Metin : {original_text}")
        except Exception as e:
            messagebox.showerror("Hata", f"Bir Hata oluştu : {str(e)}")

    def calculate_psnr(self):
        if not hasattr(self, 'stego_image'):
            messagebox.showerror("Hata", "Lütfen önce veriyi gizleyin")
            return
        psnr_value = psnr(self.image, self.stego_image)
        messagebox.showinfo("PSNR Değeri", f"PSNR: {psnr_value} dB")


if __name__ == "__main__":
    root = tk.Tk()
    app = SteganographyApp(root)
    root.mainloop()
```

Şekil 13

7. Kaynaklar

- **Github**
- **Python Kütüphaneleri**: PIL, NumPy ve diğer ilgili kütüphanelerin kullanımı.
- **Youtube**
- **ChatGPT**