Znaki Wodne - widoczne i niewidoczne

Adrian Sekuła, Adam Fedoruk Cyberbezpieczeństwo, AGH 2023

Wprowadzenie

Aplikacja "Znaki Wodne - Widoczne i Niewidoczne" umożliwia dodawanie zarówno widocznych, jak i niewidocznych znaków wodnych do różnych mediów, takich jak obrazy, pliki MP3 i strony internetowe. Aplikacja została zaimplementowana w języku Python, korzystając z biblioteki `tkinter` do interfejsu graficznego oraz różnych modułów do obsługi różnych rodzajów znaków wodnych.

Przegląd rozwiązań State of The Art

Widoczne znakowanie wodne (Visible Watermarking):

Widoczne znakowanie wodne jest świadomie dostrzegalne i często używane do wskazywania na autora lub właściciela obrazu, wideo lub innego cyfrowego medium. Przykłady to logo firmy na zdjęciach lub wideo.

- **Złożone grafiki i teksty:** Użycie bardziej skomplikowanych grafik, złożonych wzorów i tekstu, które są trudne do usunięcia bez naruszenia jakości oryginalnego obrazu.
- **Adaptacyjność:** Algoritmy, które dynamicznie dostosowują widoczne znaki wodne do treści, by były mniej inwazyjne dla użytkownika, ale nadal skuteczne w zapobieganiu nadużyciom.
- **Technologie Real-Time:** Możliwość dodawania znaków wodnych na żywo w trakcie streamingu wideo, aby chronić prawa autorskie w czasie rzeczywistym.

Niewidoczne znakowanie wodne (Invisible Watermarking):

Niewidoczne znakowanie wodne jest ukryte i jest przeznaczone do weryfikacji autentyczności i pochodzenia bez zakłócania oryginalnej treści. Przykłady to znakowanie wodne zdjęć, utworów muzycznych, dokumentów.

- **Integracja z Machine Learning:** Wykorzystanie uczenia maszynowego do tworzenia bardziej wyrafinowanych znaków wodnych, które mogą być bardziej odporne na ataki, takie jak zakłócenia stałe, kompresja, obróbka.
- **Frekwencyjne Techniki Znakowania:** Znakowanie wodne w dziedzinie częstotliwości (np. DCT, DWT), które ma tendencję do bycia bardziej odporne na przetwarzanie sygnałów i kompresję.
- **Kryptografia:** Silne algorytmy szyfrowania sprzężone ze znakowaniem wodnym, takie jak techniki z kluczami publicznymi i prywatnymi (np. RSA), zapewniające bezpieczne, trudne do usunięcia znaki wodne.
- **Robustność przeciw Atakom:** Badanie i opracowanie metod znakowania wodnego, które mogą przetrwać różnorodne próby usunięcia i manipulacji, w tym skalaryzację, obracanie, przycinanie obrazów i zniekształcenia.

W miarę rozwoju technologii, stan techniki z rozwiązaniami znakowania wodnego ewoluuje, oferując nowe metody zabezpieczeń danych cyfrowych, które są zarówno bardziej wydajne, jak i trudniejsze do pokonania. Rozwój tych technologii jest napędzany również postępem w zakresie uczenia maszynowego, rozpoznawania obrazów oraz analizy i przetwarzania sygnałów cyfrowych.

Copyright for web content using invisible text watermarking

Praca przedstawia badania dotyczące niewidzialnego cyfrowego znakowania wodnego (watermarkingu) służącego do ochrony praw autorskich treści internetowych. Proponowana technika wykorzystuje ustalone reguły semantyczne i syntaktyczne do tworzenia znaków wodnych, które są szyfrowane, a następnie przekształcane w białe znaki (spacje) za pomocą sterowanych binarnie znaków, zanim zostaną osadzone w strukturze HTML strony internetowej. Badanie weryfikuje zaproponowany system przeciw różnego rodzaju atakom, aby zapewnić optymalną trwałość.

A Review on Digital Watermarking Using LSB

Praca przedstawia przegląd metod cyfrowego znakowania wodnego (watermarkingu) z wykorzystaniem techniki Least Significant Bit (LSB). Istotą znakowania wodnego jest osadzanie lub ukrywanie informacji w pliku cyfrowym bez widocznej zmiany samego pliku. W kontekście rosnącego ryzyka ingerencji w dane i kopiowania nielegalnego, znakowanie wodne stanowi metodę zabezpieczenia danych transmitowanych przez internet. W artykule omówiono różne techniki znakowania oparte na metodzie LSB, które wykorzystywane są do ochrony danych przed różnego rodzaju atakami.

Podstawowa technika LSB polega na zamianie najmniej znaczącego bitu piksela obrazu wartością bitu znaku wodnego, tak aby zmiana była niewidoczna dla ludzkiego oka. W artykule przedstawiono również odmiany metody LSB, takie jak metoda inwersji LSB czy wykrywanie manipulacji w obrazach kolorowych, które mają poprawić autentyczność i wykrywalność ingerencji w dane. Przytoczone prace badają różne podejścia do osadzania znaków wodnych, takie jak techniki robocze w dziedzinie czasowej (spatial domain) oraz przekształcenia w dziedzinie częstotliwości (frequency domain).

Ocenę wydajności technik znakowania wodnego przeprowadza się za pomocą dwóch parametrów: średniego kwadratu błędu (Mean Square Error - MSE) oraz wskaźnika szczytowego stosunku sygnału do szumu (Peak Signal to Noise Ratio - PSNR), które mierzą jakość obrazu ze znakiem wodnym po zastosowaniu techniki znakowania.

Digital Watermarking using Asymmetric Key Cryptography and Spatial Domain Technique

Artykuł koncentruje się na digitalnym znakowaniu wodnym (watermarkingu), które w przeszłości oceniano pod kątem niewidoczności i wytrzymałości. W tej pracy, w oparciu o rozważania bezpieczeństwa, autorzy proponują nowe, bezpieczne podejście inspirowane kryptanalizą. W celu osiągnięcia wyższego poziomu tajemności i efektywności, wykorzystują połączone podejście oparte zarówno na cyfrowym znakowaniu wodnym, jak i kryptografii, do osadzania poufnych informacji. Celem ich pracy jest zwiększenie bezpieczeństwa osadzonych w obrazie danych. W tym celu zaimplementowano technikę kryptografii asymetrycznej (RSA) oraz technikę dziedziny przestrzennej

Digital Watermarking Applications and Techniques: A Brief Review

Praca ta przedstawia przegląd w dziedzinie cyfrowego znakowania wodnego (watermarkingu), które jest wykorzystywane do zapewnienia i ułatwienia uwierzytelnienia danych, bezpieczeństwa oraz ochrony praw autorskich mediów cyfrowych. Znakowanie wodne, stosowane wobec takich treści jak audio, obrazy, wideo czy teksty, jest uważane za jedną z najważniejszych technologii w dzisiejszym świecie, zapobiegających nielegalnemu kopiowaniu danych. Praca zawiera szczegółowe opracowanie definicji znakowania wodnego oraz różnych zastosowań i technik znakowania wodnego, które są wykorzystywane do zwiększenia bezpieczeństwa danych.

Wstęp podkreśla, że rozwój Internetu doprowadził do wielu nowych możliwości tworzenia i dostarczania treści w formie cyfrowej. Wraz z tym pojawiają się jednak pytania dotyczące bezpieczeństwa danych. Obserwuje się, że obecne prawa autorskie nie są wystarczające do radzenia sobie z danymi cyfrowymi, co prowadzi do konieczności ochrony i egzekwowania praw własności intelektualnej w mediach cyfrowych. W tym kontekście cyfrowe znakowanie wodne, które koncentruje się na solidności przekazu i zdolności do przetrwania ataków na usunięcie, takich jak operacje na obrazach (obracanie, przycinanie, filtrowanie), stało się przedmiotem rosnącego zainteresowania.

Digitalne znakowanie wodne łączy pomysły i teorie z różnych dziedzin, takich jak przetwarzanie sygnałów, kryptografia, teoria prawdopodobieństwa, stochastyczna teoria technologii sieciowych, projektowanie algorytmów i inne techniki. Pozwala to na ukrycie informacji dotyczących praw autorskich w danych cyfrowych za pomocą określonego algorytmu, z możliwością późniejszego wykrycia lub ekstrakcji wody znakowej dla różnych celów, w tym zapobiegania kopiowaniu i kontroli

Opis aplikacji

Opis Rozwiązania

Ogólna Struktura Aplikacji

Aplikacja jest zbudowana w oparciu o framework `tkinter` i składa się z kilku podprogramów, z których każdy obsługuje konkretne zadanie związane z dodawaniem znaków wodnych.

Klasa `ZnakiWodneAplikacja`

- Inicjalizuje główne okno i definiuje przyciski do wyboru różnych metod dodawania znaków wodnych.
- Przechowuje przyciski, które po kliknięciu inicjują dodatkowe okna dla różnych metod.

Metody Dodające Znaki Wodne

- `visible_watermarks`: Tworzy nowe okno dla dodawania widocznych znaków wodnych (tekst/obraz).
- `lsb_watermark_password`: Tworzy nowe okno dla dodawania ukrytych znaków wodnych za pomocą metody LSB z hasłem.
- `mp3_watermarker`: Tworzy nowe okno dla dodawania znaków wodnych do plików MP3.
- `web_watermarks`: Tworzy nowe okno dla dodawania znaków wodnych do stron internetowych.

Klasy Implementujące Poszczególne Metody Dodawania Znaku Wodnego

1. **LSB_Watermark_Password**
    - Odpowiada za dodawanie ukrytych znaków wodnych do obrazów za pomocą metody LSB z hasłem.
    - Posiada interfejs graficzny z zakładkami do dodawania i odczytywania znaków wodnych.
2. **MP3_Watermarker**
    - Dodaje ukryty znak wodny w postaci tekstu do plików MP3.
    - Udostępnia interfejs graficzny z zakładkami do dodawania i odczytywania znaków wodnych.
3. **VisibleWatermarks**

- Pozwala na dodawanie widocznych znaków wodnych (tekstowych i obrazowych) do plików graficznych.
- Udostępnia interfejs graficzny z zakładkami do dodawania tekstu i obrazu jako znaków wodnych.

4. **WebWatermarksApp**

- Dodaje ukryty znak wodny w meta tagu HTML na podstawie treści strony internetowej.
- Udostępnia interfejs graficzny dla wprowadzenia adresu URL strony.

Opis Kodu Aplikacji

Pierwszy moduł (main_app.py) stanowi główną część programu odpowiedzialną za uruchamianie podmodułów związanych z dodawaniem różnych rodzajów znaków wodnych. Program jest napisany w języku Python 3.

Struktura Kodu

Importowanie Bibliotek

```
import tkinter as tk
from visible_watermarks import VisibleWatermarks
from LSB_password_gui import LSB_Watermark_Password
from mp3_watermarker_tags import MP3_Watermarker
from web_watermark import WebWatermarksApp
```

Klasa `ZnakiWodneAplikacja`

Główna klasa programu, odpowiedzialna za interfejs graficzny i uruchamianie podmodułów.

Metoda `__init__(self)`

Konstruktor inicjujący główne okno programu.

```
def __init__(self):
    # Inicjalizacja głównego okna
    self.root = tk.Tk()
    self.root.geometry("500x200")
    self.root.title("Znaki wodne - Widoczne i Niewidoczne")

    # Tworzenie etykiety tytułowej
    self.title_label = tk.Label(
        self.root, text="Wybierz jedną z poniższych metod dodawania znaku wodnego:")
    self.title_label.pack()

    # Tworzenie przycisków do uruchamiania różnych metod
    self.launch_button = tk.Button(
        self.root, text="Znak wodny widoczny (tekst/obraz)", command=self.visible_watermarks)
    self.launch_button.pack()

    self.launch_button = tk.Button(
        self.root, text="Znak wodny ukryty - LSB z hasłem", command=self.lsb_watermark_password)
    self.launch_button.pack()

    self.launch_button = tk.Button(
        self.root, text="Ukryty znak wodny w tagu pliku MP3", command=self.mp3_watermarker)
    self.launch_button.pack()

    self.launch_button = tk.Button(
        self.root, text="Ukryty watermark w meta tagu HTML", command=self.web_watermarks)
    self.launch_button.pack()

#### Metoda `launch_app(self)`
Metoda do uruchamiania głównego okna programu.

```python
def launch_app(self):
    self.root.mainloop()
```

Metody Uruchamiające Podmoduły

Metody te tworzą nowe okno dla każdego z podmodułów i uruchamiają je.
```

```
def visible_watermarks(self):
    newWindow = tk.Toplevel(self.root)
    newWindow.grab_set()
    x = self.root.winfo_x()
    y = self.root.winfo_y()
    newWindow.geometry("+%d+%d" % (x, y))
    app = VisibleWatermarks(newWindow)
    app.pack_elements()

def lsb_watermark_password(self):
    # Analogicznie do visible_watermarks

def mp3_watermarker(self):
    # Analogicznie do visible_watermarks

def web_watermarks(self):
    # Analogicznie do visible_watermarks
```

Blok Główny

Sprawdzanie, czy skrypt jest uruchamiany jako plik główny, a nie importowany jako moduł.

```
if __name__ == "__main__":
    app = ZnakiWodneAplikacja()
    app.launch_app()
```

Użycie

Uruchomienie tego skryptu skutkuje otwarciem głównego okna programu, umożliwiającego wybór różnych metod dodawania znaków wodnych.

Wymagania

Program wymaga zainstalowanych bibliotek: `tkinter`, oraz zaimportowanych kolejnych modułów programu `visible_watermarks`, `LSB_password_gui`, `mp3_watermarker_tags`, `web_watermark`.

Znak wodny widoczny (tekst/obraz)

Opis

Moduł `VisibleWatermarks` odpowiada za dodawanie widocznych znaków wodnych do obrazów. Moduł ten umożliwia użytkownikowi dodawanie zarówno tekstowych, jak i obrazowych znaków wodnych do wybranych obrazów.

Struktura Kodu

Importowanie Bibliotek

```
import tkinter as tk
import tkinter.ttk as ttk
from tkinter import filedialog, messagebox
from PIL import Image, ImageDraw, ImageFont, ImageTk
import numpy as np
```

Klasa `VisibleWatermarks`

Klasa ta zawiera metody i elementy interfejsu graficznego potrzebne do dodawania widocznych znaków wodnych.

Metoda `__init__(self, root)`

Konstruktor inicjujący obiekt `VisibleWatermarks`. Przyjmuje jako argument główne okno aplikacji.

```
def __init__(self, root):
    # Inicjalizacja głównego okna
    self.root = root

    # Inicjalizacja zakładek (Notebook)
    self.notebook = ttk.Notebook(root)
    self.text_tab = tk.Frame(self.notebook)
    self.image_tab = tk.Frame(self.notebook)
    self.notebook.add(self.text_tab, text='Watermark - Text')
    self.notebook.add(self.image_tab, text='Watermark - Image')

    # Elementy interfejsu graficznego dla zakładki tekstowej
    # ...

    # Elementy interfejsu graficznego dla zakładki obrazowej
    # ...

    # Dodatkowy przycisk "Pomoc"
    self.info_button = tk.Button(
        root, text="Pomoc", command=self.show_help_box)
```

Metoda `pack_elements(self)`

Metoda ta umieszcza elementy interfejsu graficznego na odpowiednich zakładkach.

```
def pack_elements(self):
    # Pakowanie zakładek
    self.notebook.pack()

    # Pakowanie elementów interfejsu graficznego dla zakładki tekstowej
    # ...

    # Pakowanie elementów interfejsu graficznego dla zakładki obrazowej
    # ...

    # Pakowanie dodatkowego przycisku "Pomoc"
    self.info_button.pack()
```

Metoda `add_watermark_and_save(self)`

Metoda ta obsługuje dodawanie znaków wodnych do obrazu i zapisywanie go. Wywoływana jest po naciśnięciu przycisku "Generuj i zapisz obraz z znakiem wodnym". Metoda ta obsługuje zarówno znaki wodne obrazowe jak i tekstowe. Znajduje się w niej logika odpowiadająca za umieszczenie znaku wodnego w odpowiednim miejscu i w odpowiednim rozmiarze.

```
def add_watermark_and_save(self):
    # ...
```

Metoda `show_help_box(self)`

Metoda ta wyświetla okno pomocy z opisem funkcji i sposobu użycia modułu.

```
def show_help_box(self):
    # ...
```

Użycie

Aby skorzystać z funkcji modułu `VisibleWatermarks`, należy utworzyć obiekt tej klasy i użyć metody `pack_elements()`. Następnie użytkownik może wybrać zakładkę i dostosować ustawienia znaku wodnego, a po kliknięciu przycisku "Generuj i zapisz obraz z znakiem wodnym" wybrać obraz do oznaczenia oraz miejsce zapisu.

Pomoc

Watermark - Text

Pierwsza zakładka służy dodawaniu tekstu jako znaku wodnego do obrazu.

Opcje:

- Tekst znaku wodnego: wpisany przez użytkownika znak wodny tekstowy.
- Kolor znaku wodnego: wybór koloru tekstu znaku wodnego.

- Położenie znaku wodnego: określa położenie tekstu.
- Rozmiar znaku wodnego: określa rozmiar tekstu znaku wodnego.

Aby wygenerować obraz ze znakiem wodnym:

1. Wpisz tekst znaku wodnego.
2. Ustaw opcje znaku wodnego.
3. Kliknij przycisk "Generuj...".
4. W nowym oknie wybierz obraz do oznakowania.
5. W kolejnym oknie wybierz miejsce zapisu obrazu oraz jego nazwę.

Watermark - Obraz

Druga zakładka służy dodawaniu obrazu PNG jako znaku wodnego do obrazu.

Opcje:

- Położenie znaku wodnego: określa położenie obrazu znaku wodnego.

Aby wygenerować obraz ze znakiem wodnym:

1. Ustaw położenie znaku wodnego.
2. Kliknij przycisk "Generuj...".
3. W nowym oknie wybierz obraz, który chcesz oznakować.
4. W kolejnym oknie wybierz znak wodny, który chcesz dodać do obrazu.
5. W kolejnym oknie wybierz nazwę oraz miejsce zapisu oznakowanego obrazu.

Wymagania

Moduł ten wymaga zainstalowanych bibliotek: `tkinter`, `PIL`, `numpy`. Można je zainstalować używając narzędzia do zarządzania pakietami w Pythonie, na przykład pip:

```
pip install tkinter
pip install pillow
pip install numpy
```

Znak wodny ukryty - LSB z hasłem

Opis

Moduł `LSB_Watermark_Password` umożliwia ukrywanie i odczytywanie tekstowych znaków wodnych w obrazach PNG za pomocą metody najmniej znaczącego bitu (LSB). Do zakodowania znaku wodnego wykorzystywane jest hasło, które jest również niezbędne do późniejszego odczytania znaku.

Struktura Kodu

Importowanie Bibliotek

```
import sys
import numpy as np
from PIL import Image
import tkinter as tk
from tkinter import filedialog, messagebox
from tkinter import ttk
```

Klasa `LSB_Watermark_Password`

Klasa ta zawiera metody i elementy interfejsu graficznego potrzebne do zakodowania i odczytania tekstowego znaku wodnego przy użyciu metody LSB.

Metoda `__init__(self, root)`

Konstruktor inicjujący obiekt `LSB_Watermark_Password`. Przyjmuje jako argument główne okno aplikacji.

```
def __init__(self, root):
    # Inicjalizacja głównego okna
    self.root = root

    # Inicjalizacja zakładek (Notebook)
    self.notebook = ttk.Notebook(root)
    self.root.title("Znak Wodny LSB z hasłem")

    # Inicjalizacja zakładek dla zakodowania i odczytania znaku wodnego
    self.encode_tab = ttk.Frame(self.notebook)
    self.decode_tab = ttk.Frame(self.notebook)
    self.src_filename = None
    self.src_filename_decode = None

    # Dodatkowy przycisk "Pomoc"
    self.info_button = tk.Button(
        root, text="Pomoc", command=self.show_help_box)
```

Metoda `Encode(self, src, message, dest, password)`

Metoda kodująca tekstowy znak wodny w obrazie. Wymaga podania pliku źródłowego (`src`), treści znaku wodnego (`message`), ścieżki docelowej (`dest`) i hasła (`password`).

```
def Encode(self, src, message, dest, password):
    # ...
```

Metoda `Decode(self, src, password)`

Metoda odczytująca tekstowy znak wodny z obrazu. Wymaga podania pliku źródłowego (`src`) i hasła (`password`).

```
def Decode(self, src, password):
    # ...
```

Metoda `pack_elements(self)`

Metoda umieszczająca elementy interfejsu graficznego na odpowiednich zakładkach.

```
def pack_elements(self):
    # ...
```

Metoda `show_help_box(self)`

Metoda wyświetlająca okno pomocy z opisem funkcji i sposobu użycia modułu.

```
def show_help_box(self):
    # ...
```

Użycie

Aby skorzystać z funkcji modułu `LSB_Watermark_Password`, należy utworzyć obiekt tej klasy i użyć metody `pack_elements()`. Następnie użytkownik może wybrać zakładkę "Dodaj znak wodny" lub "Odczytaj znak wodny" i dostosować ustawienia znaku wodnego, a po kliknięciu odpowiedniego przycisku przeprowadzić proces kodowania lub odczytywania.

Pomoc

Podprogram pozwala na ukrycie tekstowego znaku wodnego w obrazie PNG. <!> Program nie wspiera polskich znaków (ą, ś, ...).

Dodaj znak wodny

Opcje:

- Obraz źródłowy (PNG) - wybranie obrazu bazowego
- Znak wodny do ukrycia - tekstowy znak wodny do ukrycia w obrazie
- Hasło - hasło potrzebne do odkodowania znaku wodnego

Aby dodać niewidoczny tekstowy znak wodny:

1. Wybierz obraz źródłowy PNG
2. Wpisz znak wodny tekstowy do pola tekstowego

3. Wpisz hasło wymagane do odkodowania znaku wodnego
4. Kliknij "Zakoduj"
5. W nowym oknie wybierz nazwę i miejsce zapisu podpisanego obrazu

Odczytaj znak wodny

Opcje:

- Obraz źródłowy (PNG) - wybranie obrazu bazowego z ukrytym tekstem
- Hasło - hasło podane przy zakodowywaniu znaku wodnego

Aby odczytać ukryty tekstowy znak wodny:

1. Wybierz obraz źródłowy PNG
2. Wpisz hasło wymagane do odkodowania znaku wodnego
3. Kliknij przycisk "Zdekoduj"
4. Odczytaj znak wodny w nowym oknie

Wymagania

Moduł ten wymaga zainstalowanych bibliotek: `tkinter`, `PIL`, `numpy`. Można je zainstalować używając narzędzia do zarządzania pakietami w Pythonie, na przykład pip:

```
pip install tkinter
pip install pillow
pip install numpy
```

Ukryty znak wodny w tagu pliku MP3

Opis

Moduł `MP3_Watermarker` umożliwia dodawanie i odczytywanie ukrytych znaków wodnych w plikach MP3 poprzez edycję meta tagów.

Struktura Kodu

Importowanie Bibliotek

```
import tkinter as tk
import tkinter.ttk as ttk
from tkinter import filedialog, messagebox
from mutagen.id3 import ID3, TXXX
from shutil import copy2
import os
```

Klasa `MP3_Watermarker`

Klasa ta zawiera metody i elementy interfejsu graficznego potrzebne do dodawania i odczytywania tekstowych znaków wodnych w plikach MP3.

Metoda `__init__(self, root)`

Konstruktor inicjujący obiekt `MP3_Watermarker`. Przyjmuje jako argument główne okno aplikacji.

```
def __init__(self, root):
    # Inicjalizacja głównego okna
    self.root = root

    # Inicjalizacja zakładek (Notebook)
    self.notebook = tk.ttk.Notebook(self.root)
    self.root.title("MP3 Watermark")

    # Inicjalizacja zakładek dla dodawania i odczytywania znaku wodnego
    self.tab1 = tk.Frame(self.notebook)
    self.tab2 = tk.Frame(self.notebook)

    # Dodatkowy przycisk "Pomoc"
    self.info_button = tk.Button(
        root, text="Pomoc", command=self.show_help_box)
```

Metoda `pack_elements(self)`

Metoda umieszczająca elementy interfejsu graficznego na odpowiednich zakładkach.

```
def pack_elements(self):
    # ...
```

Metoda `open_file_tab1(self)`

Metoda otwierająca dialog wyboru pliku dla zakładki "Dodaj znak wodny".

```
def open_file_tab1(self):
    # ...
```

Metoda `open_file_tab2(self)`

Metoda otwierająca dialog wyboru pliku dla zakładki "Odczytaj znak wodny".

```
def open_file_tab2(self):
    # ...
```

Metoda `add_watermark_tab1(self)`

Metoda dodająca tekstowy znak wodny do pliku MP3. Znak wodny dodawany jest jako dodatkowy tag użytkownika z nazwą "Watermark".

```
def  add_watermark_tab1(self):
        filename  =  self.open_file_tab1()
        if  filename:
                original_file  =  filename
                original_file_copy  =  original_file  +  "_original"
                copy2(original_file, original_file_copy)

                audio  = ID3(original_file)
                if  "TXXX:Watermark"  in  audio:
                        audio["TXXX:Watermark"].text = [self.watermark.get()]
                else:
                        audio["TXXX:Watermark"] = TXXX(
                                encoding=3, text=self.watermark.get())
                audio.save()

                watermarked_file  =  original_file[:-4] +  "_watermarked.mp3"
                os.rename(original_file, watermarked_file)
                os.rename(original_file_copy, original_file)
```

Metoda `read_watermark_tab2(self)`

Metoda odczytująca tekstowy znak wodny z pliku MP3.

```
def  read_watermark_tab2(self):
        filename  =  self.open_file_tab2()
        if  filename:
                audio  = ID3(filename)
                watermark  =  audio.getall("TXXX")
                if  watermark:
                        messagebox.showinfo(
                                "Znak wodny", "Odczytany znak wodny:\n"  +  watermark[0].text[0])
                else:
                        messagebox.showinfo(
                                "Znak wodny", "Nie znaleziono znaku wodnego!")
```

Metoda `show_help_box(self)`

Metoda wyświetlająca okno pomocy z opisem funkcji i sposobu użycia modułu.

```
def show_help_box(self):
    # ...
```

Użycie

Aby skorzystać z funkcji modułu `MP3_Watermarker`, należy utworzyć obiekt tej klasy i użyć metody `pack_elements()`. Następnie użytkownik może wybrać zakładkę "Dodaj znak wodny" lub "Odczytaj znak wodny", dostosować ustawienia znaku wodnego, a po kliknięciu odpowiedniego przycisku przeprowadzić proces dodawania lub odczytywania znaku wodnego.

Pomoc

Ukryty znak wodny w tagu pliku MP3

Dodaj znak wodny

Zakładka pozwala dodać ukryty tekstowy znak wodny do pliku MP3.

Opcje:

- Tekst znaku wodnego: znak wodny tekstowy do ukrycia w pliku MP3.

Aby dodać ukryty znak wodny tekstowy do pliku MP3:

1. Wpisz tekst znaku wodnego.
2. Kliknij "Wybierz plik..."
3. W nowym oknie wybierz plik MP3.
4. Plik MP3 ze znakiem wodnym tekstowym zostanie zapisany w tym samym folderze co plik źródłowy.

Odczytaj znak wodny

Aby odczytać ukryty znak wodny z pliku MP3:

1. Kliknij "Wybierz plik..."
2. Odczytaj znak wodny w nowym oknie.

Wymagania

Moduł ten wymaga zainstalowanych bibliotek: `tkinter`, `mutagen`. Można je zainstalować używając narzędzia do zarządzania pakietami w Pythonie, na przykład pip:

```
pip install tkinter
pip install mutagen
```

Ukryty watermark w meta tagu HTML

Opis

Moduł `WebWatermarksApp` pozwala na dodawanie ukrytych znaków wodnych w plikach HTML poprzez modyfikację meta tagów.

Struktura Kodu

Importowanie Bibliotek

```
import tkinter as tk
import requests
from tkinter import messagebox, filedialog
from bs4 import BeautifulSoup
import hashlib
```

Klasa `WebWatermarksApp`

Klasa ta zawiera metody i elementy interfejsu graficznego potrzebne do dodawania ukrytych znaków wodnych do plików HTML.

Metoda `__init__(self, root)`

Konstruktor inicjujący obiekt `WebWatermarksApp`. Przyjmuje jako argument główne okno aplikacji.

```
def __init__(self, root):
    # Inicjalizacja głównego okna
    self.root = root

    # Inicjalizacja etykiety i pola tekstowego dla adresu URL
    self.url_label = tk.Label(root, text="URL:")
    self.url_entry = tk.Entry(root, width=30)

    # Inicjalizacja przycisku do generowania i zapisywania
    self.generate_button = tk.Button(
        root, text="Wygeneruj i zapisz", command=self.add_watermark_and_save)

    # Inicjalizacja przycisku pomocy
    self.info_button = tk.Button(
        root, text="Pomoc", command=self.show_help_box)
```

Metoda `pack_elements(self)`

Metoda umieszczająca elementy interfejsu graficznego w głównym oknie.

```
def pack_elements(self):
    # ...
```

Metoda `add_watermark_and_save(self)`

Metoda dodająca ukryty znak wodny do pliku HTML.

```
def add_watermark_and_save(self):
    # ...
```

Metoda `calculate_hash(self, data)`

Metoda obliczająca funkcję skrótu SHA-256 dla danych.

```
def calculate_hash(self, data):
    # ...
```

Metoda `show_help_box(self)`

Metoda wyświetlająca okno pomocy z opisem funkcji i sposobu użycia modułu.

```
def show_help_box(self):
    # ...
```

Użycie

Aby skorzystać z funkcji modułu `WebWatermarksApp`, należy utworzyć obiekt tej klasy i użyć metody `pack_elements()`. Następnie użytkownik może wprowadzić adres strony internetowej do pola "URL:", a po kliknięciu przycisku "Wygeneruj i zapisz" zostanie dodany ukryty znak wodny do pliku HTML. Użytkownik musi również wybrać miejsce i nazwę dla wygenerowanego pliku HTML.

Pomoc

Ukryty watermark w meta tagu HTML

Podprogram służy do dodania ukrytego znaku wodnego w tagu pliku HTML. <!> W adresie strony należy zawrzeć przedrostek http:// lub https://

Aby dodać ukryty znak wodny:

1. W polu "URL:" podaj adres strony internetowej, do której chcesz dodać znak wodny.
2. Kliknij w przycisk "Wygeneruj i zapisz".
3. Wybierz miejsce oraz nazwę dla wygenerowanego pliku HTML.

Wymagania

Moduł ten wymaga zainstalowanych bibliotek: `tkinter`, `requests`, `bs4`. Można je zainstalować używając narzędzia do zarządzania pakietami w Pythonie, na przykład pip:

```
pip install tkinter
pip install requests
pip install beautifulsoup4
```

Analiza stanu aplikacji

Mocne strony:

1. **Zróżnicowane funkcje:** Aplikacja oferuje różne metody dodawania znaków wodnych, takie jak tekst, ukryty tekst w obrazie, znak w pliku MP3 i znak w meta tagu HTML.
2. **Interfejs graficzny:** Wykorzystanie biblioteki Tkinter umożliwia stworzenie prostego interfejsu graficznego dla użytkownika, co ułatwia obsługę aplikacji.
3. **Komentarze i pomoc:** Kod zawiera komentarze, co ułatwia zrozumienie poszczególnych fragmentów programu. Istnieje funkcja "Pomoc", która dostarcza informacji użytkownikowi na temat każdej funkcji.
4. **Różnorodność znaków wodnych:** Aplikacja obsługuje zarówno widoczne (tekst) jak i niewidoczne (ukryte w obrazie, pliku MP3, meta tagu HTML) znaki wodne.

Słabe strony:

1. **Brak obsługi polskich znaków:** Istnieje komunikat w jednym z komentarzy, że program nie obsługuje polskich znaków (ą, ś, ...), co może być uważane za ograniczenie.
2. **Brak obsługi błędów:** W niektórych miejscach brakuje obsługi błędów, co może prowadzić do nieprzewidzianego zachowania aplikacji w przypadku niepoprawnych danych wejściowych.
3. **Powtarzający się kod:** Kod do tworzenia nowego okna i ustawiania jego położenia powtarza się dla każdej metody dodawania znaku wodnego. To może prowadzić do problemów z utrzymaniem i rozszerzaniem aplikacji w przyszłości.
4. **Wielkość kodu:** Kod wydaje się być dość długi i nieco rozbudowany, co może utrudniać zrozumienie i zarządzanie.
5. **Brak komentarza dokumentacyjnego:** Kod nie zawiera ogólnego komentarza dokumentacyjnego, który opisuje ogólną funkcję aplikacji, jej cel i sposób użycia.

Podsumowanie:

Aplikacja mogłaby być dalej rozwijana poprzez poprawę obsługi błędów, unikanie powtarzalności kodu i lepszą dokumentację. Wprowadzenie obsługi polskich znaków również mogłoby zwiększyć jej użyteczność dla polskojęzycznych użytkowników.