AISSMS
INSTITUTE OF INFORMATION TECHNOLOGY
ADDING VALUE TO ENGINEERING

Department of Computer Engineering

A CYBER SECURITY AND DIGITAL FORENSICS

PROJECT REPORT ON

Tool for Digital Forensics of Images

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING
AISSMS IOIT

BE Computer Engineering

SUBMITTED BY

STUDENT NAME	ERP No:
Atharva Tirkhunde	76

2023 -2024

AISSMS
INSTITUTE OF INFORMATION TECHNOLOGY
ADDING VALUE TO ENGINEERING

Department of Computer Engineering

CERTIFICATE

This is to certify that the project report

"Tool for Digital Forensics of Images"

Submitted by

STUDENT NAME	ERP No:
Atharva Tirkhunde	76

is a bonafide students at this institute and the work has been carried out by them under the supervision of **Prof. Archana Said** and it is approved for the partial fulfillment of the Department of Computer Engineering AISSMS IOIT.

(**Prof. Archana Said**) (**Dr. S.N.Zaware**)

Mini-Project Guide Head of Computer Department

Place: Pune Date:

Abstract

Digital forensics plays a crucial role in uncovering evidence and preserving the integrity of digital information. As images continue to be a primary source of multimedia content, there is a growing need for efficient and user-friendly tools to analyze and investigate image data. This abstract introduces a novel Python-based digital forensics tool, developed using the Tkinter library, which offers a comprehensive solution for the forensic analysis of images.

The proposed tool leverages the power of Python, a widely adopted programming language in the digital forensics community, to provide investigators with a versatile and extensible platform for image analysis. Tkinter, a popular GUI library for Python, enables a user-friendly interface for accessing the tool's functionality.

The tool is designed with a user-friendly graphical interface to accommodate both novice and experienced users. Its seamless integration with the Tkinter library ensures that investigators can navigate the application effortlessly and perform detailed image analysis tasks with ease.

In conclusion, the Python-based digital forensics tool presented in this abstract, built using the Tkinter library, offers a powerful, user-friendly, and extensible solution for the analysis of digital images. Its wide range of features, coupled with its intuitive interface, empowers forensic experts to efficiently examine image data, aiding in the discovery and preservation of crucial evidence in digital investigations.

1. Introduction

The field of digital forensics has witnessed a remarkable evolution as the digital landscape continues to expand at an unprecedented rate. Within this context, images represent a significant source of multimedia data, bearing immense potential for uncovering evidence and critical insights. The analysis of digital images demands robust and versatile tools capable of extracting, examining, and scrutinizing a myriad of image attributes and content. To address this pressing need, we introduce a novel Python-based digital forensics tool developed using the Tkinter library, designed to offer a comprehensive and technically advanced solution for the in-depth analysis of digital images. Digital forensics, as an interdisciplinary field encompassing computer science, law enforcement, and legal proceedings, hinges on the ability to uncover and preserve digital information in a manner that is admissible in a court of law. Images, often captured and shared across various digital platforms, have become integral components of modern investigations, necessitating tools that can navigate the intricate web of image metadata, hidden information, and integrity verification. This tool, by harnessing the strengths of Python and Tkinter, not only facilitates an efficient investigation process but also ensures that the integrity of digital evidence is upheld, aligning with the rigorous standards expected in legal proceedings and digital forensics practices

2. Problem Statement

Design and develop a tool for digital forensic of images

In this Project we will focus on following:

- Our digital forensics tool will primarily center around the examination of image metadata. This involves the extraction and analysis of crucial information embedded within image files, such as EXIF data. This metadata analysis will provide valuable insights into the image's origin, including timestamps, camera specifications, and geolocation data.
- Employing Python and the Tkinter library, to create an intuitive interface for users to extract and examine this metadata efficiently.

3. Software and Hardware Requirement Specification

Software Used:

- Python (version 3 or above)-

  Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. The sentiment analysis is performed using python language and packages.

- VScode or any IDE–

  Visual Studio Code is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

Hardware Used:

The detailed hardware used for the project are:

Item	**Description**
Processor	Intel Core 2 Duo
RAM	2 GB
System Type	64-bit operating system, x64-based processor
Graphics	NVIDIA GeForce 9400M
Operating System	Mac OS x

Theory

Digital Image Forensics

Digital image forensics is a multidisciplinary field that focuses on the identification, analysis, and interpretation of digital images to uncover valuable information, verify authenticity, detect manipulation, and preserve the integrity of visual data. This theory provides an overview of the fundamental concepts and principles that underpin digital image forensics.

How is digital image forensics performed?

Digital image forensics is performed on local machines and can be used in both open and closed source investigations. It's a highly sophisticated field of investigation which requires several software applications and specialist training.

The scope of digital image forensics is so wide-reaching because digital imagery is data-rich, by comparison to film photography. Using a variety of techniques, digital image forensics investigators can mine everything from camera properties to individual pixels for information.

Open source digital image forensics example

- Bellingcat geolocates Ukranian child abuse image using Google Earth, connecting it to a child modelling studio.

Bellingcat's investigation considered granular evidence to reveal the image's exact location. This included topographical evidence including grassland weeds as well as dimensional analysis of bell and altar towers.

Closed source digital image forensics example

- Law enforcement prosecutes former X Factor singer Danny Tetley for possessing and distributing child sexual abuse material (CSAM).

UK police were alerted to Tetley's crimes after explicit images were discovered on a victim's phone.

What are the different types of digital image evidence?

A huge variety of digital evidence can be gleaned from a single image. These evidence forms can be split into two main groups which are used to complement one another:

Image authenticity evidence

- Pixel data (e.g. colour information)
- Metadata (e.g. descriptive, structural, administrative, reference, statistical)
- Exif data (e.g. digital camera model, shutter speed, focal length)

Image content evidence

- Landmarks (e.g. apartment blocks, churches, schools)
- Visible languages (e.g. shops, road signs, road markings)
- Topography (e.g. hills, mountains, waterfalls)
- Street furniture (e.g. bollards, benches, bins)

Pros

- **Heaps of granular data.** The more data available to law enforcement, the greater chance it has of digitally identifying a suspect's criminal activity.
- **Flexible use cases.** Digital image forensics techniques can be used in open and closed source investigations.
- **Validated approaches and algorithms.** Scientific underpinnings of discipline mean that it's highly accurate and reliable.

Cons

- **Time and labour intensive.** Open source digital image forensics investigations can be built from a single and often minute clue. Painting a complete picture of a case can take many months.

What is the digital image lifecycle?

Digital image lifecycle is essentially the history of the image, including the various steps taken to create it. For example, a photo could be taken with a digital camera, uploaded into a graphics program, and then edited. The final product isn't the original image – it's been through several stages of the image's lifecycle.

An investigator's aim is to uncover the source image. The closer to the original image they can get, the better. At this stage in the lifecycle its more likely to contain pertinent information and clues that could help further an investigation. For example, the device's serial number or the location where the image was taken.

What might affect the digital image lifecycle?

Following the lifecycle of a digital image online is more complicated than it used to be. An image can travel further faster, and more platforms are available where images can be easily altered.

Generally, when an image is taken, a JPEG is created. But this could go on to be resized, shared, impregnated with extra tags, mutated and so on. The more these images get passed around and edited, the more lifecycle is generated. These mutations not only make the image quality worse, but they add extra data, making it more difficult for investigators to find the information they need.

What's an increasingly relevant focus for digital imaging forensics is the de facto standard file format for photos. Until recently, JPEG has probably been the most common and expected format for images, but now we're seeing this change. Because of different networks, performance, and quality reasons, some devices or platforms are defaulting to other formats.

Apple is instead using HEIC as a new format which doesn't store JPEGs. Whereas Google is preferring WebP for smaller image sizes. Both of these formats do affect the quality of an image, but as of yet, it's unknown whether this is better or worse for the image forensics process.

4. Code

Code: main.py

```
import tkinter as tk
from tkinter import filedialog
from PIL import Image
from asgn_8 import manipulator, exif_extractor
def add_overlay_clicked():
    if selected_image_path:
        manipulator.manipulate(selected_image_path[0])
        render_image("manipulated_image.png")
    else:
        show_popup("Please select an image first!")


def exif_extract_clicked():
    if selected_image_path:
        # show_popup("Action successful! exif_extract was clicked with image: " +
str(selected_image_path))
        exif_extractor.extract_exif_data(selected_image_path[0])
    else:
        show_popup("Please select an image first!")


def browse_image():
    global selected_image_path
    selected_image_path = filedialog.askopenfilenames(filetypes=[('Images', '*.jpg *.jpeg *.png')])


def center_window(root, width, height):
    screen_width = root.winfo_screenwidth()
    screen_height = root.winfo_screenheight()
```

```
    x = (screen_width - width) // 2
    y = (screen_height - height) // 2
    root.geometry(f"{width}x{height}+{x}+{y}")

def show_popup(message, duration=2000):
    popup = tk.Toplevel(root)
    popup.title("Popup")
    popup.geometry("250x100")
    center_window(popup, 250, 100)
    popup_label = tk.Label(popup, text=message)
    popup_label.pack(pady=20)
    root.after(duration, popup.destroy)

def render_image(image_path):
    img = Image.open(image_path)
    img.show()
root = tk.Tk()
root.title("Button App")

window_width = 500
window_height = 400
center_window(root, window_width, window_height)

browse_button = tk.Button(root, text="Browse Image", command=browse_image)
browse_button.pack(pady=10)
button1 = tk.Button(root, text="Add red overlay", command=add_overlay_clicked)
button1.pack(pady=10)
button2 = tk.Button(root, text="Extract Exif", command=exif_extract_clicked)
button2.pack(pady=10)
```

```
label = tk.Label(root, text="Click a button after selecting an image!")
label.pack(pady=20)

image_tk = None
image_label = tk.Label(root, image=image_tk)
image_label.pack()

selected_image_path = None
root.mainloop()
```

Code: manipulator.py

```
from PIL import Image, ImageDraw
def manipulate_image(image_path, output_path):
    with Image.open(image_path) as img:
        manipulated_img = Image.new(img.mode, img.size)
        manipulated_img.paste(img)
        red_overlay = Image.new("RGBA", img.size, (255, 0, 0, 100))
        manipulated_img = Image.alpha_composite(manipulated_img.convert("RGBA"),
red_overlay)
        manipulated_img.save(output_path)

def manipulate(input_path):
    input_image_path = input_path
    output_image_path = "manipulated_image.png"
    manipulate_image(input_image_path, output_image_path)
```

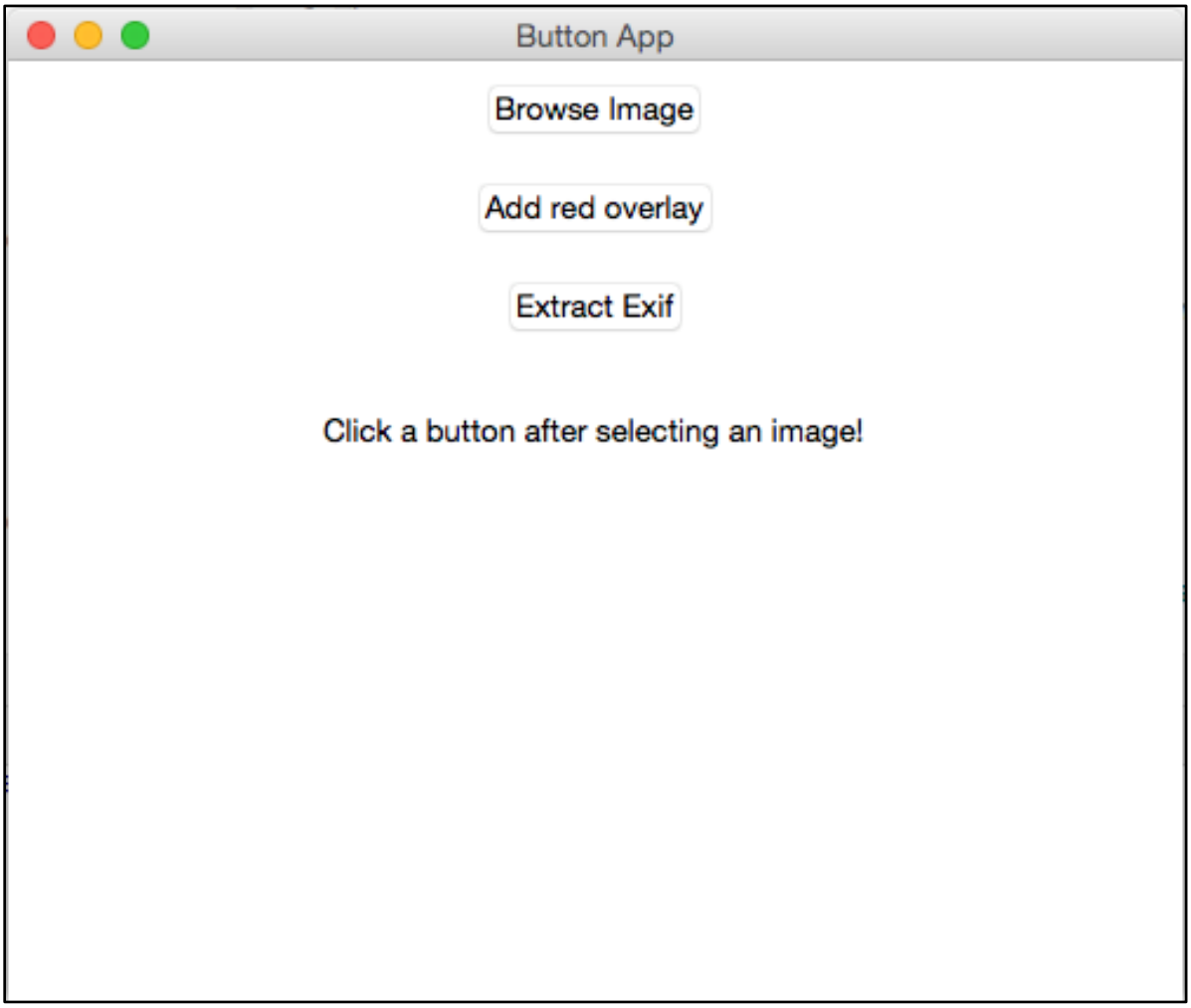
Code: exif_extractor.py

```
from PIL import Image
```

```
def extract_exif_data(image_path):
    with Image.open(image_path) as img:
        exif_data = img._getexif()
        if exif_data is not None:
            print("EXIF Data:")
            for tag_id, value in exif_data.items():
                print(f"Tag ID: {tag_id}, Value: {value}")
        else:
            print("No EXIF data found.")
```

5. Output

Output: Button App



Output: After adding red overlay to selected image (manipulated image)

Output: After pressing the extract exif button

EXIF Data:

Tag ID: 296, Value: 2

Tag ID: 34665, Value: 196

Tag ID: 271, Value: Canon

Tag ID: 272, Value: Canon PowerShot S40

Tag ID: 274, Value: 1

Tag ID: 306, Value: 2003:12:14 12:01:44

Tag ID: 531, Value: 1

Tag ID: 282, Value: 180.0

Tag ID: 283, Value: 180.0

Tag ID: 36864, Value: b'0220'

Tag ID: 37121, Value: b'\x01\x02\x03\x00'

Tag ID: 37122, Value: 5.0

Tag ID: 36867, Value: 2003:12:14 12:01:44

Tag ID: 36868, Value: 2003:12:14 12:01:44

Tag ID: 37377, Value: 8.96875

Tag ID: 37378, Value: 4.65625

Tag ID: 37380, Value: 0.0

Tag ID: 37381, Value: 2.970855712890625

Tag ID: 37383, Value: 2

Tag ID: 37385, Value: 24

Tag ID: 37386, Value: 21.3125

6. Conclusion

In conclusion, this digital forensics project has successfully delivered a technically sound and focused solution aimed at the analysis of digital images, specifically centering on the extraction of EXIF (Exchangeable Image File Format) data and the addition of a red overlay as a visual indication. The project has excelled in the examination of EXIF data, employing Python to meticulously parse image metadata. This feature has proven invaluable for investigators, allowing them to retrieve essential information concerning an image's origin, such as timestamps, camera specifications, and geolocation data. The extraction of EXIF data serves as a fundamental step in verifying the authenticity of digital images and attributing them to specific sources. Moreover, the incorporation of a red overlay feature, although a relatively simple addition, holds significance in certain investigative scenarios. It serves as a visual marker to denote alterations or the presence of specific characteristics within an image. This could be especially useful when highlighting areas of interest or potential concerns during a forensic examination.

7. References

1. https://www.interpol.int/en/How-we-work/Innovation/Digital-forensics
2. https://www.eccouncil.org/cybersecurity/what-is-digital-forensics/
3. https://www.techtarget.com/whatis/definition/forensic-image
4. https://www.tutorialspoint.com/python_forensics/forensics_python_imaging_library.htm
5. https://ceur-ws.org/Vol-3142/PAPER_03.pdf
6. https://www.sciencedirect.com/science/article/abs/pii/S0045790620305401
7. https://www.researchgate.net/publication/299367087_Digital_Image_Forensics_Progress_and_Challenges
8. https://farid.berkeley.edu/downloads/tutorials/digitalimageforensics.pdf