

PIXEL PUZZLE MASTER



**Karemah baker (4310154)**

**Ferdows Adam (4313204)**

**Basma Mokhles (4313216)**

**Ghofran Ghazi (4310675)**

Instructor: Dr. Budoor Alharbi

Python project

Cs111

[INTRODUCTION: 2](#_Toc153027130)

[detailed code analysis: 3](#_Toc153027131)

[1. Imported Libraries and Modules 3](#_Toc153027132)

[1. haslib library: 3](#_Toc153027133)

[2. Tkinter library: 3](#_Toc153027134)

[3. PIL` (Python Imaging Library): 3](#_Toc153027135)

[4. Numpy library: 3](#_Toc153027136)

[2. Functions 4](#_Toc153027137)

[5. Function `get\_sha\_key`: 4](#_Toc153027138)

[6. Function `process\_image`: 6](#_Toc153027139)

[Firstly: encoding 6](#_Toc153027140)

[secondly: decoding 8](#_Toc153027141)

[3. GUI Layout and Components 9](#_Toc153027142)

[7. Key Entry Field: 9](#_Toc153027143)

[8. Image Selection Button: 9](#_Toc153027144)

[9. Operation Selection: 10](#_Toc153027145)

[10. Process Button: 10](#_Toc153027146)

[11. Status Display: 10](#_Toc153027147)

[12. Output Display: 10](#_Toc153027148)

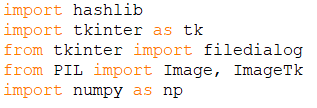
[13. Labels: 10](#_Toc153027149)

[how to use it 11](#_Toc153027150)

[concloution: 12](#_Toc153027151)

[references 13](#_Toc153027152)

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  | | --- | | PYTHON PROJECT | | PIXEL PUZZLE MASTER |     INTRODUCTION:  In the digital era, the proliferation of digital media, particularly images, has led to significant concerns regarding privacy and data security. Images often contain sensitive information, ranging from personal photographs to confidential documents. The unauthorized access and misuse of such images can lead to privacy violations and security breaches. This scenario underscores the importance of image encryption, so this project focuses on developing a Python-based application that addresses this need by providing a robust solution for image encryption and decryption.  The primary goal is to ensure that images can be encrypted (or encoded) and decrypted (or decoded) using a key, making the image readable only to those with the correct key.  Furthermore, this project creates a solution for secure image encryption and decryption. By using Python as the foundation, along with the SHA-256 hashing algorithm, tkinter for GUI, PIL for image processing, and Numpy for array manipulation, as well as all the requirements that were taken during the course (CS111) |

detailed code analysis:

## Imported Libraries and Modules

1. haslib library:

`hashlib` is used in password hashing, data integrity verification, and digital signatures. For example, storing hashed passwords in a database instead of plain text for enhanced security. In this project, `hashlib` is essential for generating a secure hash of the encryption/decryption key using the SHA-256 algorithm. It ensures that the key used in the image processing is unique and secure.

1. Tkinter library:

`tkinter` is used to create the user interface of the application, allowing users to interact with the program in a visual and intuitive way, such as selecting images and entering keys.

`filedialog` (from `tkinter`): It is utilized in the project to open a file explorer window, enabling users to select the image file they wish to encrypt or decrypt.

1. PIL` (Python Imaging Library):

`PIL` Adds image processing capabilities and it is crucial for opening, manipulating, and saving images. It handles the conversion of images into a format that can be processed (like arrays).

1. Numpy library:

`numpy` is used for efficient handling and manipulation of the image data, which is crucial for the encoding and decoding processes in the project.it adds support for large, multi-dimensional arrays and matrices.

However, Both Pillow (PIL fork) and NumPy are third-party libraries in Python, installing them by using the Python package manager, pip. Open a terminal or command prompt and enter the following commands.

A close-up of a black background

Description automatically generated

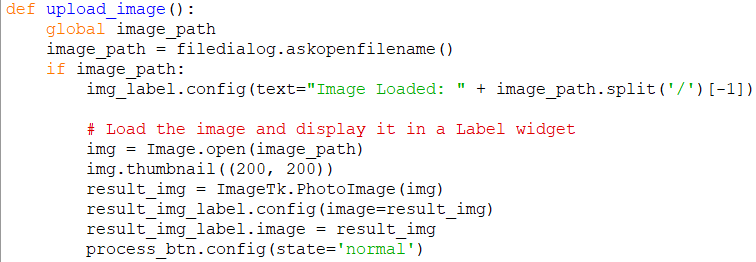
## A black text on a white background Description automatically generatedFunctions

1. Function `get\_sha\_key`:

- Purpose: The purpose of using SHA-256 is to create a unique and irreversible representation of the input data (the key) Even a small change in the input will produce a significantly different hash value

- Method:

* `key.encode()` This method converts the input string (key) into bytes. Hash functions typically operate on byte-like objects rather than strings.
* `hashlib. sha256(...)` This part initializes the SHA-256 hashing algorithm from the hashlib library in Python.
* `hexdigest()` This method is applied to the result of the SHA-256 hash. It converts the binary hash into a hexadecimal representation. The final output is a string containing the hexadecimal representation of the SHA-256 hash of the input key.

2. Function `upload\_image`:

- Purpose: The upload\_image function is designed for uploading an image file in a graphical user interface (GUI) using a file dialog

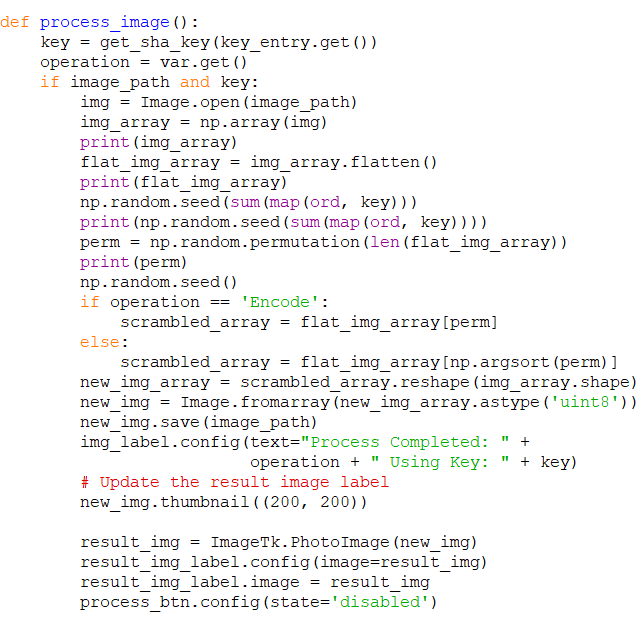
- Method:

* `global image\_path: `Declares a global variable image\_path to store the path of the uploaded image.
* `image\_path = filedialog.askopenfilename()` Opens a file dialog, allowing the user to select an image file. The selected file path is stored in the image\_path variable.
* `if image\_path: ` Checks if a valid image path has been selected.
* `img\_label.config(text="Image Loaded: " + image\_path.split('/') [-1])`Updates a label (img\_label) to indicate that an image has been loaded by displaying the filename.
* `img = Image.open(image\_path)` Opens the selected image using the Image module
* `img.thumbnail((200, 200))` Resizes the image to a thumbnail with a maximum siz of 200x200 pixels.
* `result\_img = ImageTk.PhotoImage(img)` Converts the resized image to forma compatible with Tkinter, assuming Image Tk is imported.
* result\_img\_label.config(image=result\_img): Updates a label (result\_img\_label) to display the resized image.
* `result\_img\_label.image = result\_img` Ensures that the Tkinter image reference is retained to prevent it from being garbage-collected.
* `process\_btn.config(state='normal')` Enables a button (process\_btn) to indicate that the image has been loaded and can now be processed.

1. Function `process\_image`:

- Purpose: Handles the main image processing logic, including encoding, and decoding the image.

It is divided into two parts. (If statement) where the encoding process happens and (else statement) Where the decoding process happens.



Firstly: encoding

- Method: it starts with retrieving the key from the GUI, then Calculates the SHA hash of the key. (operation variable) will hold the current state of the (var variable) which include ‘**StringVar’** method from the TK which changes depending on the user choose at the tk. **radiobutton.** Then if the (Image\_path) and the (key) are entered the statement will be True and the suits will be executed.

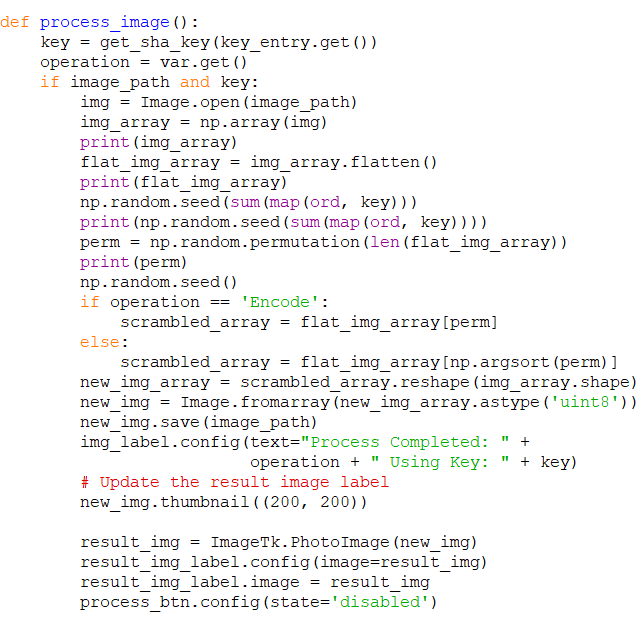
* `Image.open() `: This is a method provided by the Image class that opens an image file. The open () method takes a file path as an argument and returns an Image object representing the opened image.
* `np. array (image)`: is using the NumPy library to convert an image represented as a Pillow Image object (img) into a NumPy array (img\_array).
* The ‘print(img\_array)’, ‘print(flat\_img\_array)’ and ‘print (np. random.seed(sum(map(ord, key))))’ are used to show the process and the changes that happened on the array, for that ,deleting them will not affect the code
* ` flat\_img\_array = img\_array.flatten()` is using the NumPy library to flatten a multi-dimensional NumPy array (img\_array) into a one-dimensional array (flat\_img\_array) to be able to perform operations or analyses that require a one-dimensional representation of the image.
* ` np. random.seed (sum(map(ord, key)))` is setting the seed for the NumPy random number generator. The seed value is determined by summing the Unicode values of the characters in the provided key. This ensures that if the program is run again, it will produce the same sequence of random numbers, making the random process reproducible as long as the key remains constant.
* ` perm = np.random.permutation(len(flat\_img\_array))` the code generates a random permutation of indices, which is often useful for tasks like shuffling data samples

`. permutation (): `The specific usage in this code involves passing the length of the flat\_img\_array as an argument, and it returns a NumPy array representing a random ordering of integers from 0 to len(flat\_img\_array)-1

To make it clear, it is like making a new hotel. The hotel space depends on the ‘len(flat\_img\_array) ` and its room number are organized depending on the `random.seed` function.

* ` np. random.seed()`without any argument is used to reset the random seed of the NumPy random number generator to its default value, which is typically based on the system time.
* Now if the user chooses the encoding process ` if operation == 'Encode’: `

The suit `scrambled\_array = flat\_img\_array[perm]` is using the NumPy array indexing to create a new array (scrambled\_array) by rearranging the elements of the original array (flat\_img\_array) based on the random permutation stored in the array perm.



secondly: decoding

- Method: this part reshaping the **scrambled\_array** to the original image dimensions, and then saving and displaying the result.

* `scrambled\_array=flat\_img\_array[np.argsort(perm)]`

Reorder the elements of flat\_img\_array based on the sorted indices (np.argsort(perm)).

`. argsort ()` : is used to indirectly sort an array or to obtain the sorted indices

* `new\_img\_array = scrambled\_array.reshape(img\_array.shape):`Reshape the shuffled array (scrambled\_array) to match the original shape of the image (img\_array). it is preparing the reshaped array for creating a new image.
* `new\_img = Image.fromarray(new\_img\_array.astype('uint8')):` generally this part creates a new Pillow Image object (new\_img) from the reshaped array (new\_img\_array) with pixel values represented as unsigned 8-bit integers. `. astype('uint8’) `this method is used to convert the data type of new\_img\_array to an unsigned 8-bit integer. This is because the pixel values in an image are commonly represented as integers, and the 'uint8' (unsigned 8-bit integer) data type is suitable for representing grayscale images with pixel values ranging from 0 to 255. For visualizing or saving images, it's often common to use 'uint8', especially for standard color images, as it provides a good balance between memory efficiency and visual fidelity. That is why you should upload a small image only.

`Image.fromarray(...): `This is a method provided by the Pillow library (PIL) for creating a new image from a NumPy array.

* `new\_img.save(image\_path)` Saves the new image to the specified file path.
* img\_label.config(text="Process Completed: " + operation + " Using Key: " + key):

Update a label with information about the completed process, including the operation and key used.

* `new\_img.thumbnail((200, 200)):` Preparing a smaller version of the image for display (as at upload\_image function).
* `result\_img = ImageTk.PhotoImage(new\_img)` Create a Tkinter PhotoImage object from the new image.
* `result\_img\_label.config(image=result\_img) `Configure a Tkinter label widget to display the result image (as at upload\_image function).
* `process\_btn.config(state='disabled’) ` Preventing further processing until the user takes appropriate actions.

## GUI Layout and Components

1. Key Entry Field:

key\_entry = tk.Entry(root, bg='white')

A text box where the user enters a secret key for encryption or decryption. These keys form the basis of security for the image processing operation.

1. Image Selection Button:

upload\_btn = tk.Button(root, text="Upload Image", command=upload\_image, \*\*style)

A button that, when clicked, opens a file dialog allowing the user to select the image they wish to encode or decode.

1. Operation Selection:

process\_btn = tk.Button(root, text="Process Image", command=process\_image, \*\*style, state='disabled')

process\_btn = tk.Button(root, text="Process Image", command=process\_image, \*\*style, state='disabled')

Radio buttons or a dropdown menu for the user to select the desired operation: either 'Encode' or 'Decode'. This choice determines how the application processes the image.

1. Process Button:

process\_btn = tk.Button(root, text="Process Image", command=process\_image, \*\*style, state='disabled')

A button to initiate the encoding or decoding process once the key is entered and the image is selected.

1. Status Display:

img\_label = tk.Label(root, text="No Image Loaded", wraplength=200, \*\*style)

A label or text area that displays messages to the user. It be configured during the process. From "No Image Loaded" to "Process Completed” as an example

1. Output Display:

result\_img\_label = tk.Label(root, \*\*style)

An area to display the resulting image after the process or provide a button to save the processed image.

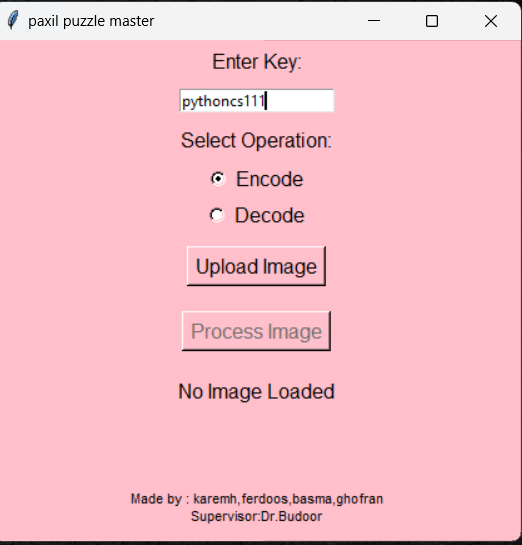
1. Labels:

key\_label = tk.Label(root, text="Enter Key:", \*\*style)

name\_label = tk.Label(root, text="""Made by : karemh,ferdoos,basma,ghofran)

It is used to clarify the parts of the interface to help the user to understand and interact with it.

how to use it:

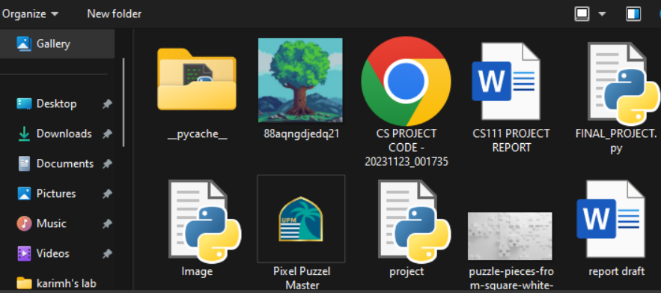
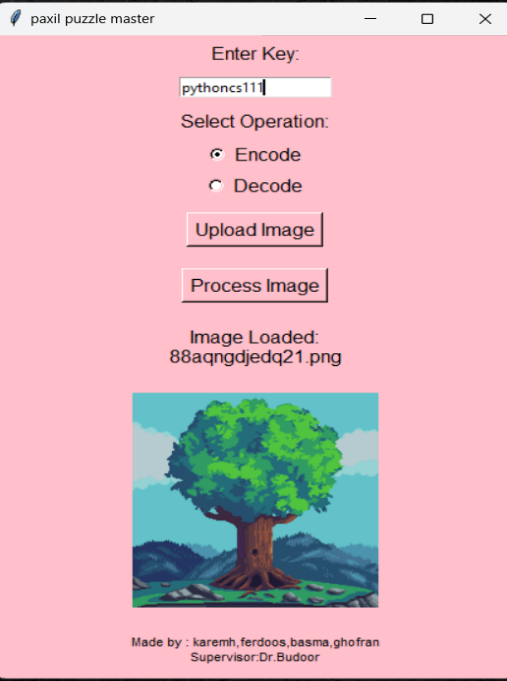


1. Enter your key (it accepts any data type)

and choose the operation that you want. Then press upload image to upload your photo.

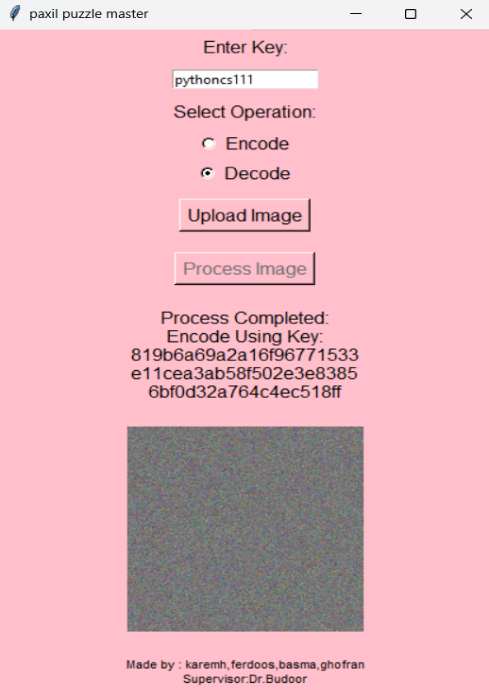


1. Open the program from this icon (an Executable Window)

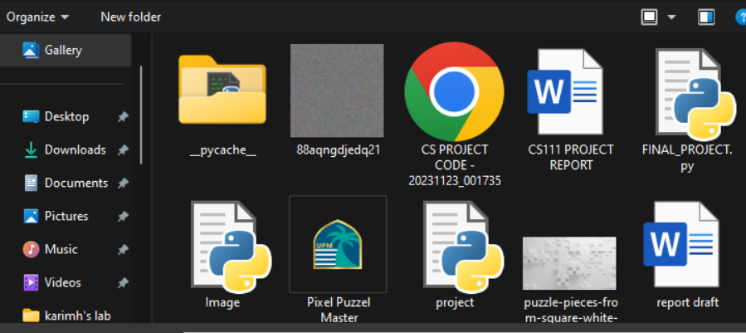


1. Choose the photo that you want to encode.
2. After loading the image

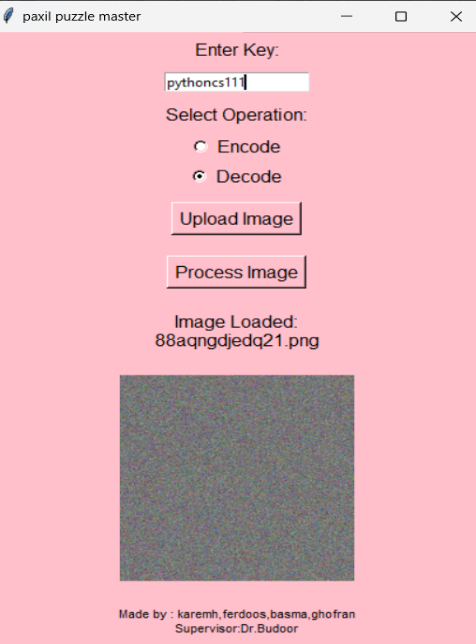
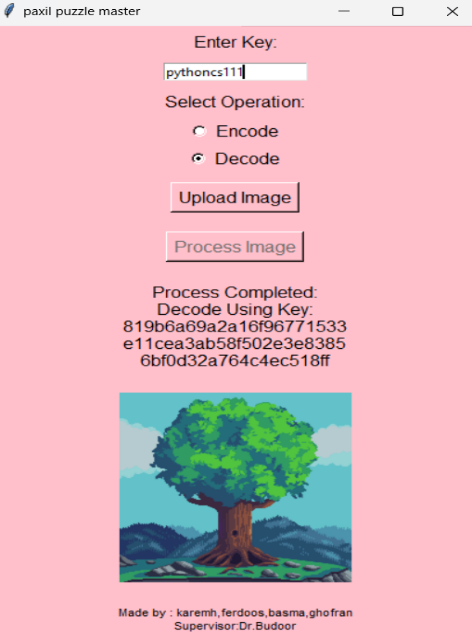
Press process image



5. you will see the encrypted image, to decode it change the operation to decoding and press upload image again.



6. you will find your original photo encrypted too, so choose it again



7. after uploading the encrypted photo enter the same key that you entered before then press again process image button to start the decoding process

8. finally, your image is decrypted and come back to its original state

concloution:

In conclusion, this program offers a practical solution to the growing demand for secure image processing tools. In a digital landscape dominated by the exchange of visual information, the ability to encode and decode images securely is of paramount importance. This program, with its user-friendly interface and robust cryptographic mechanisms, provides a valuable tool for individuals and professionals alike.

Simply, it is using the Tkinter library for creating a GUI application. The application allows users to upload an image, select an operation (either encoding or decoding), provide a key, and then process the image accordingly. The image processing involves shuffling and reshaping the pixel values based on a key using NumPy and Pillow libraries.

references:

* <https://ecampusontario.pressbooks.pub/commbusprofcdn/chapter/1-2/>
* <https://docs.python.org/3/library/hashlib.html?highlight=hashlib#module-hashlib>
* <https://www.geeksforgeeks.org/sha-in-python/>
* <https://docs.python.org/3/library/tk.html>
* <https://pypi.org/project/Pillow/>
* <https://numpy.org/doc/stable/reference/arrays.html>
* <https://www.cisco.com/c/en/us/products/security/what-is-information-security-infosec.html>