**Data Compression System for Backbone Network**

**Overview:**

This tool is designed to compress, encrypt, decompress, and decrypt files efficiently, making it ideal for backbone network applications where data size, speed, and security are crucial. It provides a simple GUI interface using Tkinter, making it user-friendly even for non-technical users.

**Python Libraries used for Compression and Decompression:**

1) Zlib – used to compress and decompress files

Methods used:

Zlib.compress()

Zlib.decompress()

2) Os : allows the program to access the operating system, access the files present in the system etc.

Methods used:

os.path.dirname() – gets the directory path / folder path from any path

os.path.join() – creates a new path for the new file in the same folder

os.path.getsize() – gets the size of a file from the system

os.path.basename() – returns the name of a file

3) Time : used for profiling in the code

Methods used:

Time.time() – storing the current time in seconds

4) Tkinter : Standard GUI (Graphical User Interface) library for Python. It lets you build windows, buttons, text boxes, menus, popups etc.

Modules used:

Filedialog – displays or opens a file dialog box containing all the files of the user system

Messagebox – displays or opens a messagebox that usually represents the results

Methods used:

Filedialog.askopenfilename() – allows user to select a file from the system

Messagebox.showinfo() – creates a pop up message box displaying information

Messagebox.showerror() – creates a pop up message box displaying error

**Algorithm used for encryption and decryption:**

AES(Advanced Encryption Standard):

KEY = get\_random\_bytes(16) : Generates a random 16-byte encryption key

AES.MODE\_CBC : A secure mode of AES that requires padding and an IV

iv : Initialization vector . It is a type of an initial input that ensures that the same plaintext, when encrypted multiple times with the same key, produces different ciphertexts.

**Challenges faced in the tool:**

1) Overwriting of the file – when encrypting files one after the other, they get overwritten with the same name i.e.-compressed\_file.zlib or decompressed\_file.txt

Solution : added string formatting to overcome the problem

f”decompressed\_{file\_path}”

2) Extension changes during decompression – While decompression, every file gets saved to a .txt format (since, we save the file as decompressed\_file.txt)

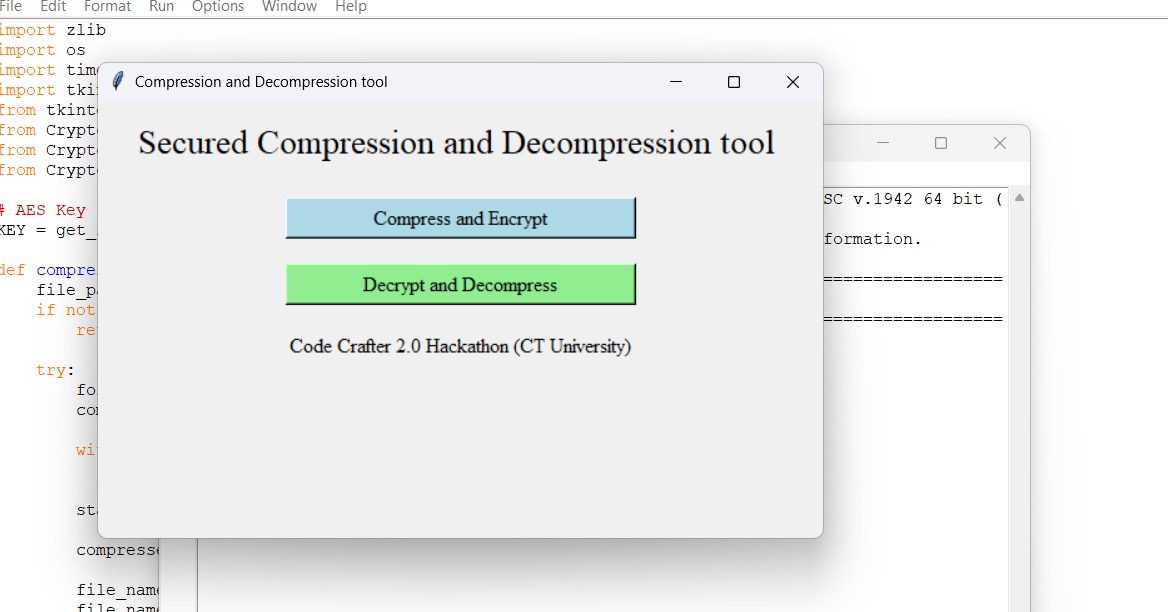
Solution : stored the file name and extension separately so that during decompression, program recognises to recover the exact file name (without changing its extension).

**Key Benefits:**

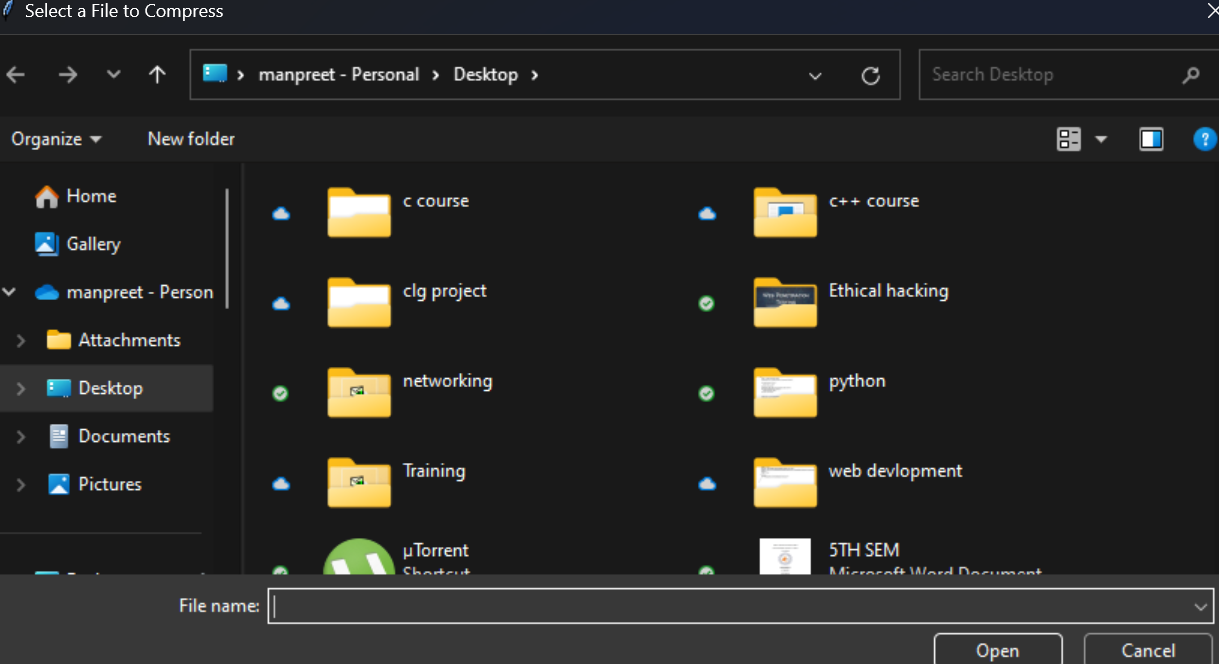
* **Reduced File Size:** Compresse data to minimize transmission time and storage space.
* **Enhanced Security:** AES encryption ensures secure transmission of sensitive files.
* **User-Friendly Interface:** Tkinter GUI makes file selection and operation intuitive.
* **Efficient File Handling:** Supports large files with time profiling for performance tracking.
* **Flexible File Support:** Maintains original file names and extensions during decompression.

**Screenshots of the tool**

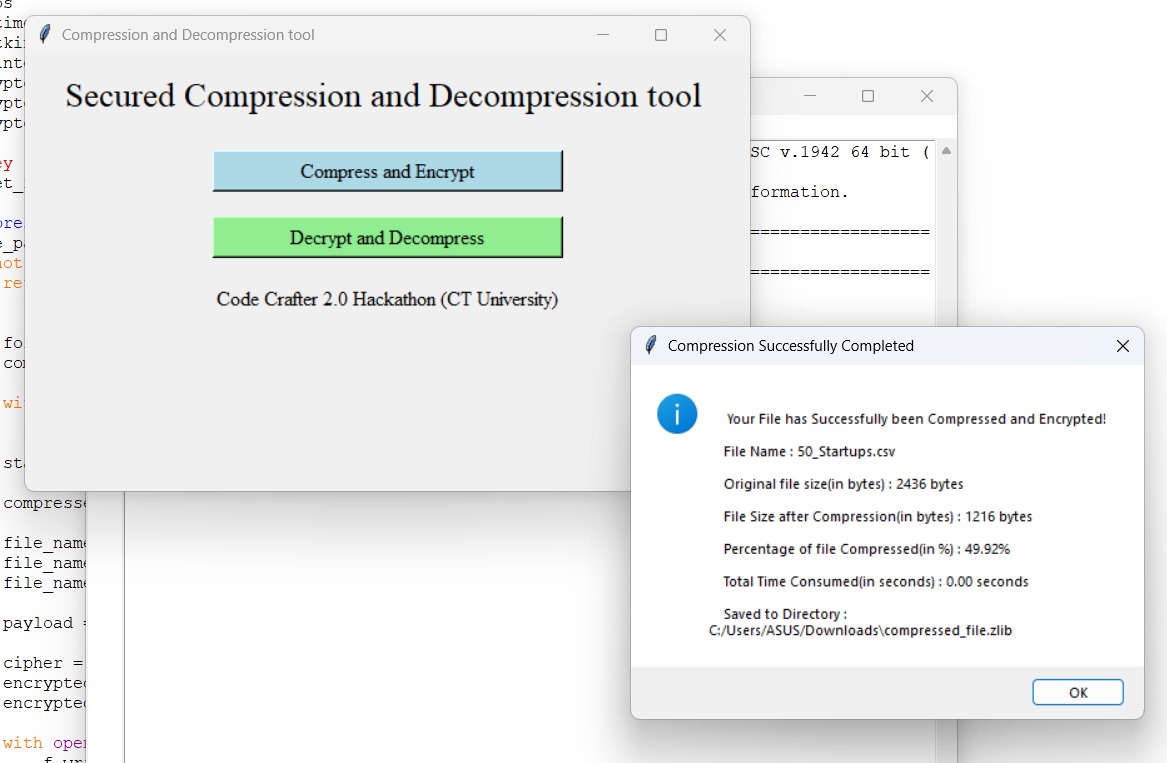
With the help of python code we made a tool with which we can securely compress a file irrespective of its format, and then decompress it back to its original format.



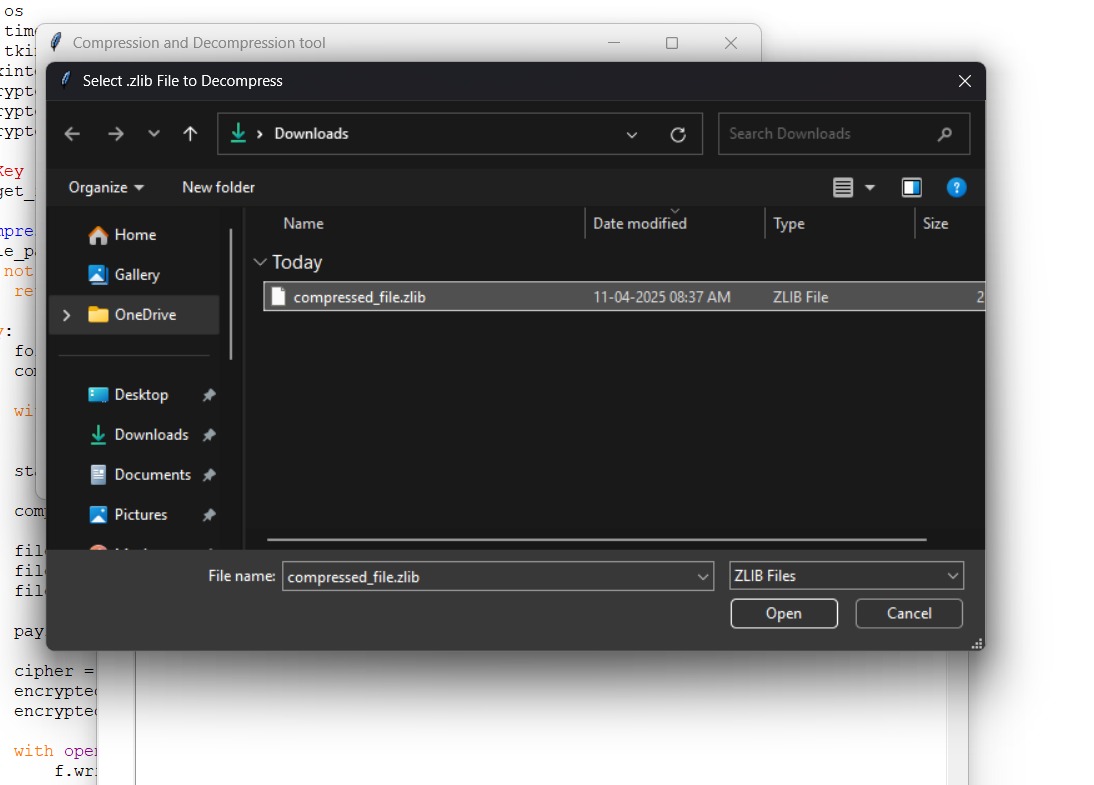
Using Python built-in GUI tool , Tkinter, user can firstly select any file of their choice for its compression and encryption



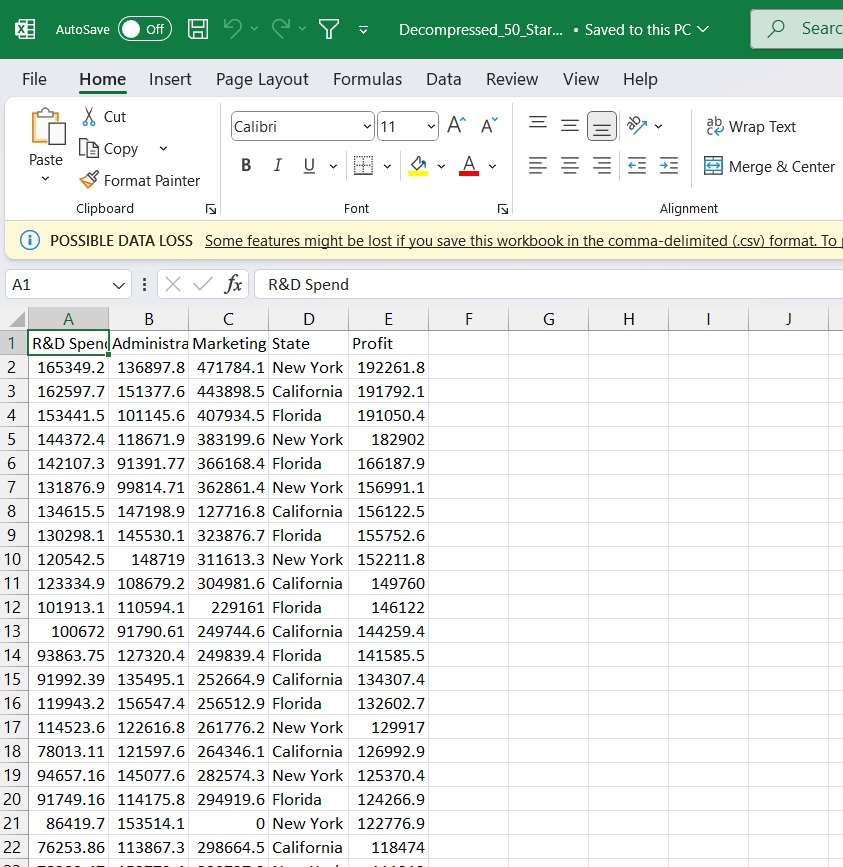
After succesfully compressing the files, a pop up box displaying file statistics shows up. It consists information like file name, original file size, reduced file size after compression, total time consumed to compress the file.



The new compressed file is saved in the same directory as the original file.



After compression, we can decompress the compressed file that is in .zlib format



Using cisco packet tracer, we have created a real-time simulation for the same file.

