CLOUDCrypt Source Code:

```
import tkinter as tk

import tkinter.ttk as ttk

import tkinter.messagebox as messagebox

from tkinter import filedialog

import base64

import dropbox

from Crypto.Cipher import AES

import os

from tkinter import simpledialog


class CloudCryptGUI:

    def __init__(self, master):

        self.master = master

        master.title("CloudCrypt")

        master.geometry("400x300")


        # Create tab control

        self.tab_control = ttk.Notebook(master)

        self.tab_control.pack(expand=1, fill="both")


        # Encryption tab

        self.encryption_tab = ttk.Frame(self.tab_control)

        self.tab_control.add(self.encryption_tab, text='Encryption')

        self.init_encryption_tab()


        # Decryption tab
        self.decryption_tab = ttk.Frame(self.tab_control)
        self.tab_control.add(self.decryption_tab, text='Decryption')
        self.init_decryption_tab()

        # Dropbox access token
        self.access_token = "sl.B1KRmfFvBKpnO9ndO-
_u6Q8fERpEm1aIKtDFjxntmnbeHzxcpnN8rnsH0uleMY4iK6gw1ymJ3E6jVuN6kZSBsHUpgUHAFRZIfgA7RK
UBzOkoVxjXe9dkeePmVPcjfFawxixXAxof5lBCpifreLMw0w8"

    def init_encryption_tab(self):
        # Create frames
        self.frame1 = tk.Frame(self.encryption_tab, bg="lightblue")
        self.frame1.pack(fill="x", padx=10, pady=5)

        self.frame2 = tk.Frame(self.encryption_tab, bg="lightblue")
        self.frame2.pack(fill="x", padx=10, pady=5)

        self.frame3 = tk.Frame(self.encryption_tab, bg="lightblue")
        self.frame3.pack(fill="x", padx=10, pady=5)

        # Create labels and entries
        self.label1 = tk.Label(self.frame1, text="Select Encryption Algorithm:", bg="lightblue")
        self.label1.pack(side="left")

        self.algorithm_var = tk.StringVar()
        self.algorithm_var.set("AES")  # default value

        self.aes_radio = tk.Radiobutton(self.frame1, text="AES", variable=self.algorithm_var, value="AES",
bg="lightblue")
        self.aes_radio.pack(side="left")
        self.label2 = tk.Label(self.frame2, text="Select File:", bg="lightblue")
        self.label2.pack(side="left")

        self.file_path_label = tk.Label(self.frame2, text="", bg="lightblue")
        self.file_path_label.pack(side="left")

        self.browse_button = tk.Button(self.frame2, text="Browse", command=self.browse_file,
bg="lightgray")
        self.browse_button.pack(side="left")

        # Create entry for encryption key
        self.label3 = tk.Label(self.frame3, text="Enter Encryption Key:", bg="lightblue")
        self.label3.pack(side="left")

        self.key_entry = tk.Entry(self.frame3, width=20, show="*")
        self.key_entry.pack(side="left")

        # Create encrypt button
        self.encrypt_button = tk.Button(self.frame3, text="Encrypt and Upload",
command=self.encrypt_button_clicked, bg="lightgray")
        self.encrypt_button.pack(side="left")

    def init_decryption_tab(self):
        # Create frame
        self.decryption_frame = tk.Frame(self.decryption_tab, bg="lightgreen")
        self.decryption_frame.pack(fill="both", expand=True, padx=10, pady=5)

        # Create decrypt button
        self.decrypt_button = tk.Button(self.decryption_frame, text="Download and Decrypt",
command=self.decrypt_button_clicked, bg="lightgray")
        self.decrypt_button.pack(side="top", padx=10, pady=5)

    def browse_file(self):
        file_path = filedialog.askopenfilename()
        if file_path:
            self.file_path_label.config(text=file_path)
        else:
            self.file_path_label.config(text="")

    def encrypt_button_clicked(self):
        key = self.key_entry.get()
        algorithm = self.algorithm_var.get()
        file_path = self.file_path_label.cget("text")

        if not file_path:
            messagebox.showerror("Error", "Please select a file.")
            return

        if not key:
            messagebox.showerror("Error", "Please enter an encryption key.")
            return

        try:
            # Encrypt the file
            with open(file_path, "rb") as file:
                data = file.read()
            if algorithm == "AES":
                cipher = AES.new(key.encode(), AES.MODE_EAX)
                ciphertext, tag = cipher.encrypt_and_digest(data)
                nonce = cipher.nonce
            else:
                messagebox.showerror("Error", "Unsupported encryption algorithm.")
                return

            file_name = os.path.basename(file_path)

            self.upload_dropbox(file_name, ciphertext)

            self.save_encryption_info(file_name, key, tag, nonce)

            messagebox.showinfo("Success", "File encrypted and uploaded successfully.")
        except Exception as e:
            messagebox.showerror("Error", f"An error occurred: {str(e)}")

    def decrypt_button_clicked(self):
        algorithm = self.algorithm_var.get()

        try:
            file_name, key, tag, nonce = self.load_encryption_info()

            ciphertext = self.download_dropbox(file_name)

            if algorithm == "AES":
                cipher = AES.new(key.encode(), AES.MODE_EAX, nonce=nonce)
                decrypted_data = cipher.decrypt_and_verify(ciphertext, tag)
                with open("decrypted_" + file_name, "wb") as file:
                    file.write(decrypted_data)

                messagebox.showinfo("Success", "File downloaded and decrypted successfully.")
            else:
                messagebox.showerror("Error", "Unsupported encryption algorithm.")
                return
        except Exception as e:
            messagebox.showerror("Error", f"An error occurred: {str(e)}")

    def upload_dropbox(self, destination_file_name, data):
        """Uploads a file to Dropbox."""
        dbx = dropbox.Dropbox(self.access_token)
        dbx.files_upload(data, "/" + destination_file_name)

    def download_dropbox(self, source_file_name):
        """Downloads a file from Dropbox."""
        dbx = dropbox.Dropbox(self.access_token)
        _, response = dbx.files_download("/" + source_file_name)
        return response.content

    def save_encryption_info(self, file_name, key, tag, nonce):
        file_namee = simpledialog.askstring("Input", "Enter the Name of User you want to share with")
        with open(file_namee+".txt", "a") as info_file:
            info_file.write(f"File Name: {file_name}\n")
            info_file.write(f"Key: {base64.b64encode(key.encode()).decode('utf-8')}\n")
            info_file.write(f"Tag: {base64.b64encode(tag).decode('utf-8')}\n")
            info_file.write(f"Nonce: {base64.b64encode(nonce).decode('utf-8')}\n\n")
        def load_encryption_info(self):
            file_namee = simpledialog.askstring("Input", "Enter the file name you have provided by the sender
    ex: (xxxxxx.txt):")
            with open(file_namee, "r") as info_file:
                lines = info_file.readlines()
                for i in range(len(lines)):
                    file_name = lines[i].strip()[11:]
                    key = base64.b64decode(lines[i + 1].strip()[5:]).decode('utf-8')
                    tag = base64.b64decode(lines[i + 2].strip()[5:])
                    nonce = base64.b64decode(lines[i + 3].strip()[7:])


                    print(file_name)
                    return file_name, key, tag, nonce


    root = tk.Tk()
    app = CloudCryptGUI(root)
    root.mainloop()
```