**Dropbox Encryption Project**

**Introduction**

Dropbox is an online file storage provider founded in 2007 by Drew Houston and Arash Ferdowsi. The file hosting service provides cloud storage and file synchronization. Dropbox works by creating a folder on the user’s computer and synchronizing the contents of the folder to its servers and other computers or devices that the user has installed Dropbox, thereby keeping the files up-to-date on all devices.

Dropbox also implements various security mechanisms to ensure that data is secure. According to its website, Dropbox files at rest are encrypted using 256-bit Advanced Encryption Standard (AES). Moreover, Secure Sockets Layer (SSL)/Transport Layer Security (TLS) is used to transmit data between Dropbox’s app and servers. The channel is secured by 128-bit or higher AES encryption. Dropbox also provides user’s the option to use two-step verification at login.

Despite these security features, Dropbox does not provide client-side encryption to users (encrypting data on the sender’s side before the transmission of data to a server). Client-side encryption improves security because it prevents service providers and third parties from gaining access to data. The current project seeks to remedy the shortcoming of Dropbox by adding client-side encryption to Dropbox. More specifically, this project will implement client-side encryption for shared files on Dropbox.

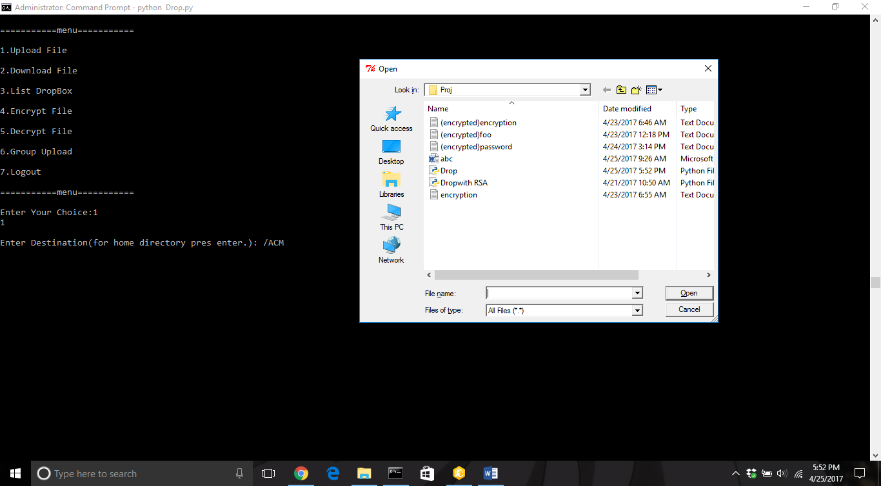
**Project Implementation**

There are several requisites needed to implement the project. First, we assume that the user can run Python on Windows. Second, the user must also have Tkinter and PyCrypto support for Python. The program used in this project imports methods from these modules. Third, the user must receive an access token from Dropbox. This can be done by going to the Dropbox developer’s website and creating an app. The access token is necessary to use Dropbox’s API. The API allows the user to upload and download files through the command line; there is no need to manually log into Dropbox and download or upload files. Fourth, each person who seeks to share or view a file must have a Dropbox account. This is necessary because Dropbox only allows its users to upload and download files. Without the ability to do so, the user cannot download or upload keys or encrypted files. Fifth, we assume that there already exists a shared folder with the appropriate access controls. Our program does not add or remove the permissions of a shared folder. These permissions must be adjusted through Dropbox’s website itself.

Instead, each time a user signs up in our application a folder is created with his username and a file named again as username.txt ic created inside it with public key of the user as its content. The program creates “Proj” folder in “C:/” and “downloads” folder inside “C:/Proj” where it stores all the downloads. When the user runs the programs for the first time, he/she is given an option of Login or Sign UP. The user then chooses signup option and enters his details and app token. While signing up the program creates a file named “PUPR\_username of the user.txt” where it stores the public private keys generated, encrypted and uploaded into Dropbox so that the user can use them from any system and also creates a file named “PWD.txt” which stores all user info and password are stored. Then after signing up the user is asked to login with the details he provided and if the user is verified then he is given with a menu filled with tasks which he can perform.

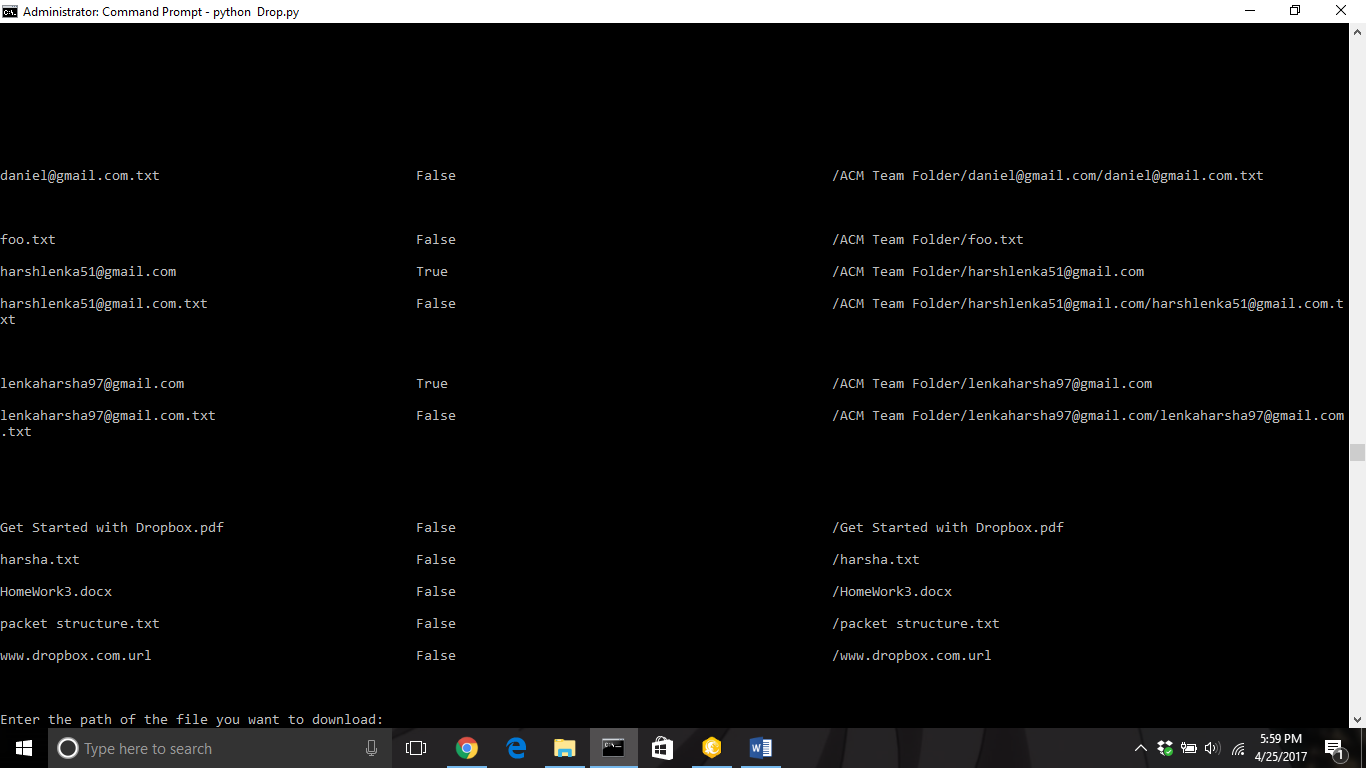
Our program consists of various modules (or options), which together allow the user to encrypt and decrypt a file on Dropbox. The modules are as follows:

**Upload Module:** This module allows a user to upload files of different formats to a desired location in Dropbox. For convenience, it provides a dialog box that allows you to browse through files on the local system. Note: This module does not encrypt the files that are uploaded.

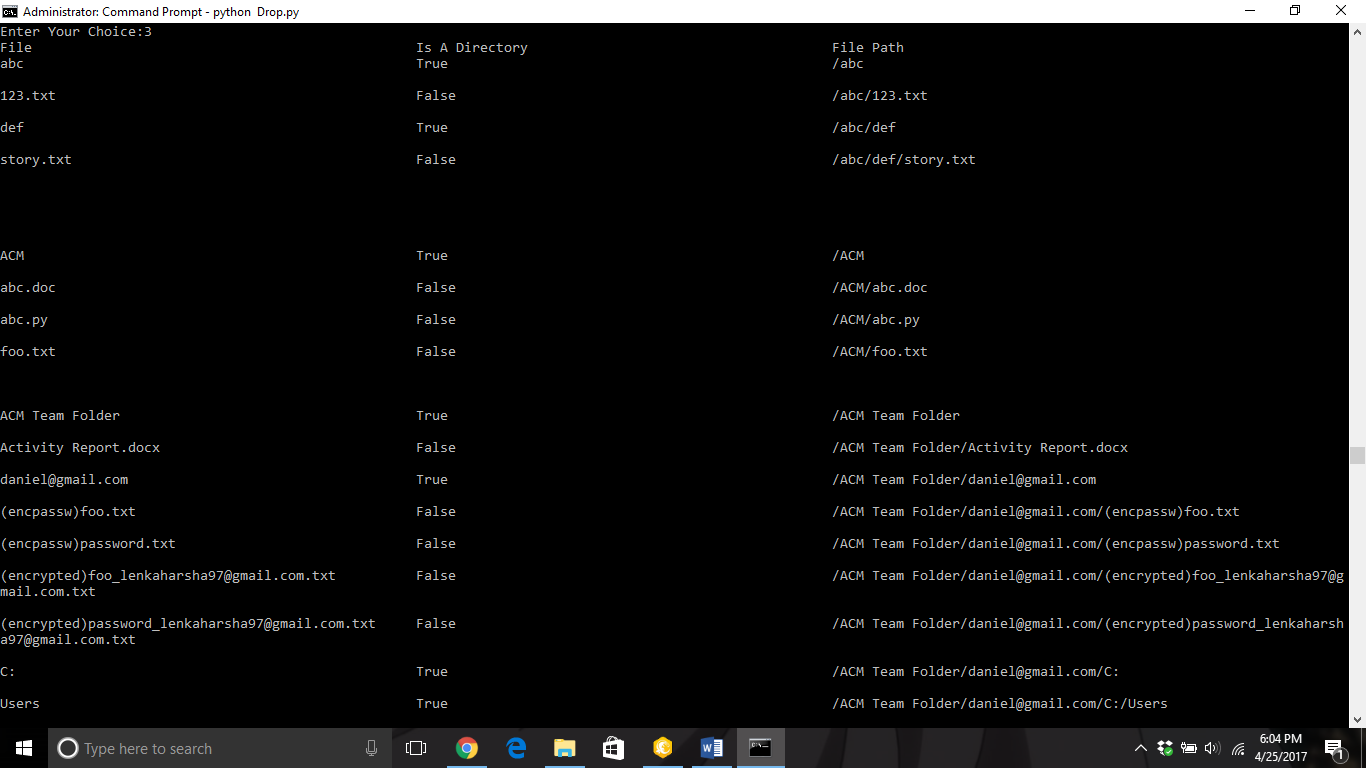
**Screenshot:**

**Download Module:** This module gives the user a list of files and folders and their paths. Then the user is asked to choose the path to his desired file. It downloads the file into “C:\Proj\downloads”.

**Screenshot:**

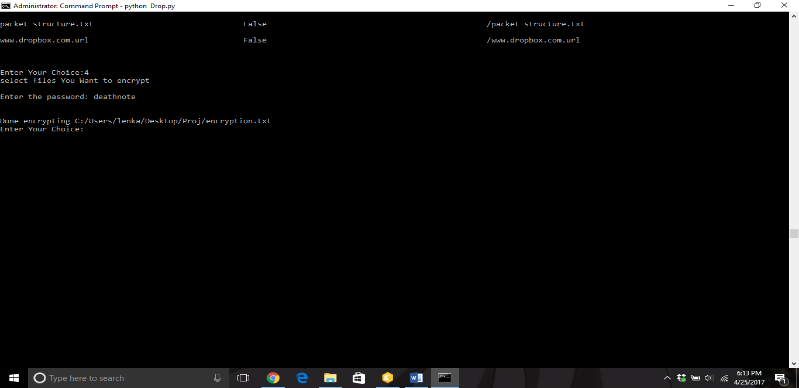


**List Files:** This module gives out the list of files and folders available on Dropbox. It also lists their paths.

**Screenshot:**

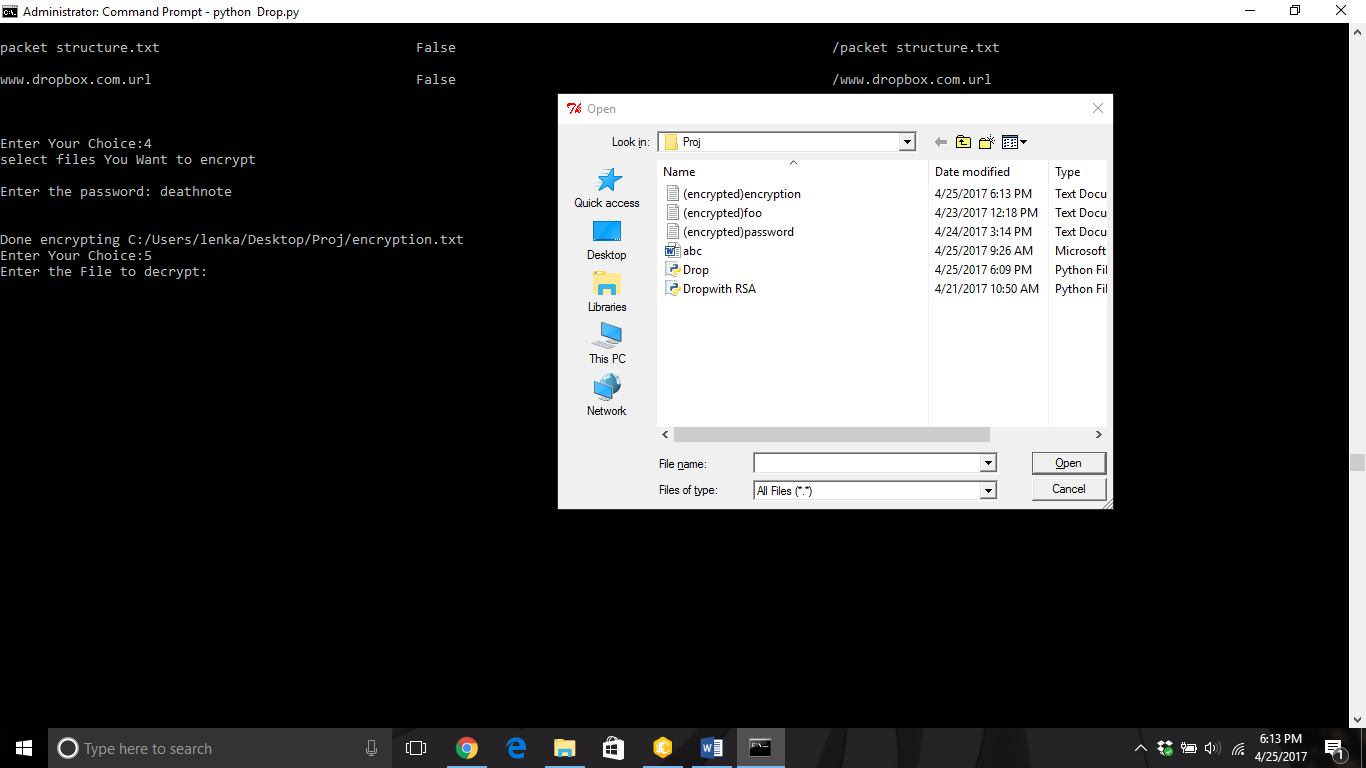
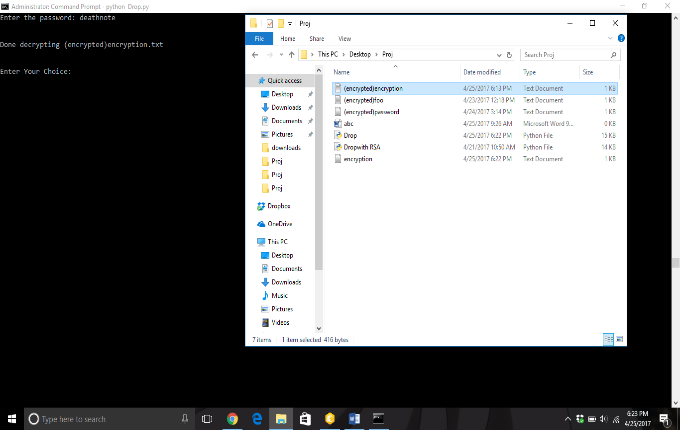
**Encrypt Module:** This module allows the user to encrypt his/her files using AES-256 in CBC mode. The user directs the application to the file he or she wants to encrypt. The user is then asked for a password. The password is hashed with SHA-256 and is used as the secret key for AES. The secret key and a randomly generated initialization vector is used to encrypt the file. All encrypted files are attached with the prefix “(encrypted)”.

**Screenshot:**



**Decrypt Module:** This module accepts the files with “(encrypted)” as prefix and decrypts them.

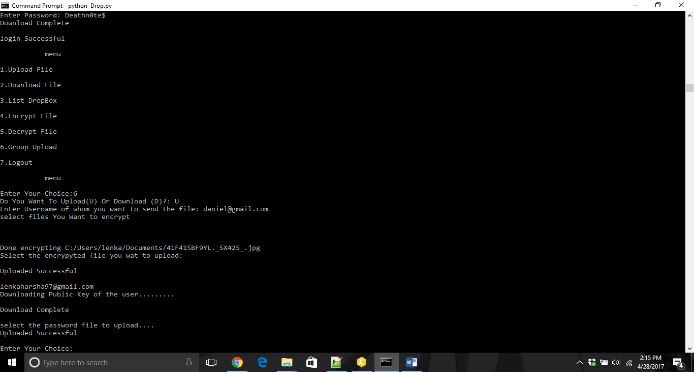
**Screenshot:**

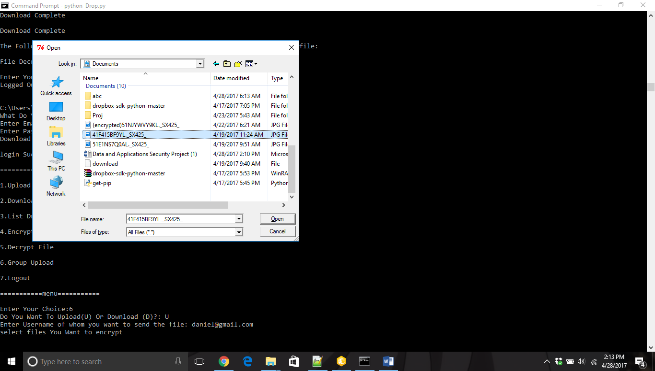


**Group Upload:** This module is specifically designed to encrypt and decrypt files on a shared folder in Dropbox. This module has two parts: the first module encrypts and uploads the file and the second module decodes a file.

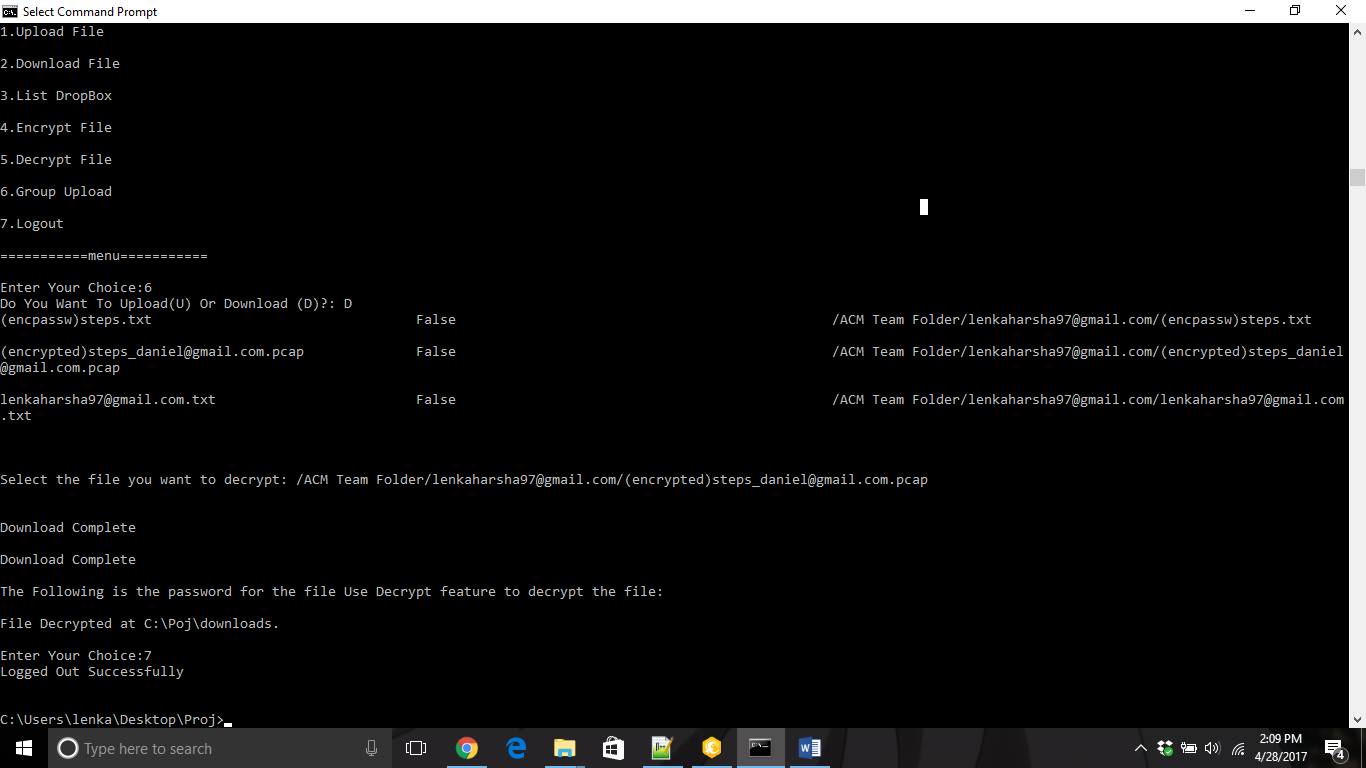
**Upload:** This module asks the user for the username of the person whom you want to send the file and the file you want to send. Then the file is encrypted using the encryption module and the public key of the user whom you want to send the file is download from Dropbox. The public key is used to encrypt the password. The encrypted password is written in a format that cannot be written to a file. It is encoded using base64 encoding and saved to a file with a prefix “(encpassw)” followed by the file name. The file is then sent to its destination in the user’s folder in Dropbox.

**Screenshot:**





**Download:** The download module asks you to enter the path of the file you wish to download. It then extracts the data from the file and derives the password. Then it downloads both the files and decrypts the password file with base 64 and then with RSA private key of the user and print the file password which can then be used to decrypt the file downloaded.

**Screenshots:**

**Difficulties in the Implementation**

Our group faced several difficulties in implementing the program. First, it took our group some time to become acquainted with Dropbox’s API. We needed to learn what the API allowed us to do and what it did not allow us to do. It was also somewhat difficult to understand the parameters that each method required. We needed to test the methods often to determine how to make it fully functional. The second problem we faced was determining the project’s design. We were often uncertain how to manage the public and private keys. We also had difficulty determining how to balance security with usability. The third difficulty we faced was implementing RSA. We found that small mistakes resulted in a number of errors that were hard to correct. The fourth difficulty was designing how to store user information and keys in such a way as to make the program implementation easier.

**Division of Labor**

The division of labor is hard to describe as we both worked on the program together in many aspects of its design and implementation. Each of us have taken up a module at a time and discussed if we had come across any errors that stalled us. Finally, we also put all the pieces of the puzzle together to get the desired output.