CS5002 Final Project

Team:

Ruotian Zhang,

Zhiliang Yu

Context:

For modern people living in the information age, life cannot be separated from accounts and passwords: email, shopping, banks, school accounts, Facebook, Instagram, and so on. The use of these passwords, on the one hand, ensures that we can have our own digital space, but on the other hand, it also poses risks of privacy breaches and the inconvenience of forgetting passwords. We choose to research the ways of encoding, decoding, and storing passwords. Our goal is to develop a password manager for local password management.

- The requirements for setting new passwords have become increasingly stringent, from simple six-digit codes to today's inclusion of uppercase, lowercase, special characters, and even fingerprint unlocking and facial recognition. I used to entrust all my passwords to Chrome until my it crashed last month, and I lost all my stored account information. I had to spend a long time remembering or resetting them, and that is also the reason why I think that locally stored passwords may offer more stability and security, just like how I used to write down passwords in my diary when I was a child. Thus, developing a local password manager comes into my mind – this will be a practical and useful application for both my understanding of encryption and Python programming skills.

- Recently, in our Number Theory module, I learned about the application of modular arithmetic in encryption technologies, which deeply sparked my interest in cryptography and information encryption. Modular arithmetic plays an important role in many encryption algorithms, especially in public-key cryptography and hash functions. Understanding these encryption principles made me realize that by correctly applying these techniques, we can effectively protect digital data. Even if someone sees the encrypted passwords, without knowing the key, they cannot access the actual password, thus ensuring the security of our accounts. Therefore, I want to integrate modular arithmetic and other cryptographic principles to design a more powerful and secure password manager. Such a tool would not only store passwords but also encrypt user information, ensuring that even in the event of a data breach, the information remains secure.

Application:

Our application will be a password manager, designed to help people store passwords locally and enhance the security and privacy of passwords using advanced cryptographic techniques.

So far we have built a Manager module, which serves as the main entry point of the application, responsible for initializing and managing the entire application workflow. It encompasses creating the main interface, handling user login and registration, and integrating other modules such as List, Add, and Search. The List module is tasked with storing, managing, and displaying saved passwords, while the Add module allows users to input new password records. Moreover, a significant focus has been placed on developing an Encryption module, which involves not just basic password encryption but also converting strings into binary numbers and encoding and decoding using various formats like UTF-8 or UTF-16. This module is designed to ensure that stored information remains secure, even in the event of a data breach.

To enhance user experience, we have employed the Tkinter library to create graphical user interfaces, constructing a range of interfaces including the main login window, add window, and list window. These interfaces are designed to make the process of password management more intuitive and user-friendly.

Currently, we have built a basic Caesar cipher encryption method for testing purposes, but it is not secure or complex enough. To address this issue, we plan to explore more advanced encryption algorithms such as AES and RSA.

Scope:

Our project aims to develop a comprehensive local password manager. In terms of functionality, we intend to implement basic password storage and management features, including adding, listing, searching, and deleting passwords. These functions will be supported by a secure local database. Moreover, we will focus on the development of the encryption module to ensure the security of stored passwords. Currently, our plan is to upgrade from simple Caesar-style encryption to more advanced encryption algorithms such as AES and RSA, to enhance security and complexity.

We will also conduct exploratory analysis to study how users interact with the password manager, including analyzing patterns of password storage and access. In addition, we will consider developing additional features like autofill and multi-user support to increase the application's utility and flexibility.

Due to time and resource constraints, we may not be able to explore advanced features related to cloud synchronization or online security. Our focus will be on providing a stable, secure, and easy-to-use local password management solution. The ultimate goal of this project is to create a tool that can effectively protect user passwords while being easy to use in everyday life.

Description:

What we have done:

1. Separate the project into modules:

   To make our developing clear, we split it into different parts to accomplish it step by step. Below is a brief structure of the project, while some additional functions might be added after the fundamental skeleton has been built.

   （1） Manager

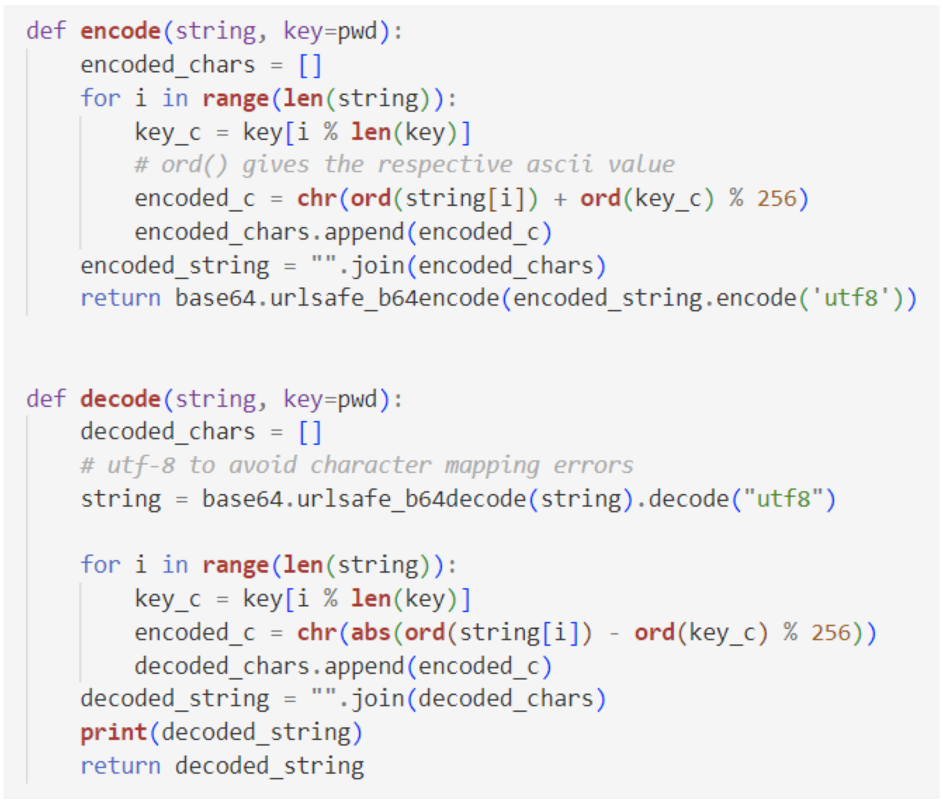
   The purpose of this module is to serve as a main entry of the application, to initialize and manage the entire application work flow. This module contains the creation of the main interface, handling user login and registration, and integrating other modules (such as *List*, *Add*, *Search*). It deals with user inputs, navigation, and state management of the entire application.

   （2） List

   The purpose of this module is to store, manage and display the stored passwords. It is responsible to read the user's input password, pass it to the *Encrypt* and *Add* module and parse the stored data. It also provides the user with an interface to view the saved password list and browse and copy it.

   （3） Encrypt

   This is the most significant module, because we don't want to totally use the built-in encryption algorithms in Python and just import and refer to them. This part will manage the encryption and decryption of passwords, and ensure the stored data is not easily accessible by unauthorized parties. This process includes not only the encryption but also the string converting into binary numbers, as well as encoding and decoding strings in formats like UTF-8 or UTF-16. We also use hash function to check whether the password for the manager is correct.



```python
def encode(string, key=pwd):
    encoded_chars = []
    for i in range(len(string)):
        key_c = key[i % len(key)]
        # ord() gives the respective ascii value
        encoded_c = chr(ord(string[i]) + ord(key_c) % 256)
        encoded_chars.append(encoded_c)
    encoded_string = "".join(encoded_chars)
    return base64.urlsafe_b64encode(encoded_string.encode('utf8'))


def decode(string, key=pwd):
    decoded_chars = []
    # utf-8 to avoid character mapping errors
    string = base64.urlsafe_b64decode(string).decode("utf8")

    for i in range(len(string)):
        key_c = key[i % len(key)]
        encoded_c = chr(abs(ord(string[i]) - ord(key_c) % 256))
        decoded_chars.append(encoded_c)
    decoded_string = "".join(decoded_chars)
    print(decoded_string)
    return decoded_string
```

Figure 1 Encrypt method for now

（4） Add

This module is to enable user to add new password records, including new keywords, usernames and passwords. Then the passwords and related information will be stored in a local database after encrypted. We might want to add the website URL afterwards if possible to see if we can realize the Autofill function.

（5） Search

This module is designed for user to quickly find specific password in the stored records. The user can enter the keyword to search the particular result, and the search result will be displayed.

The modules listed above will be able to store passwords for a single user and search existing passwords for the user. Apart from this basic function, we may add functions like Autofill or Multiusers. So far, we have built the *List*, *Add* and simple *Encrypt* part for test, *Search* and a more complicated *Encrypt* module in processing.

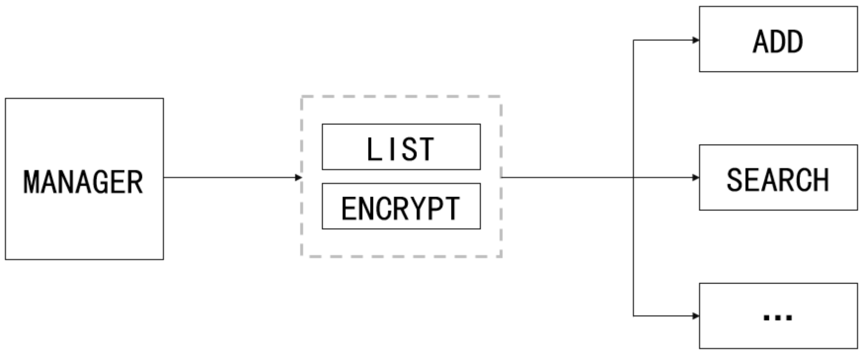


Figure 2 Structure of the Password Manager

2. <u>GUI</u>

   It doesn't look great to just use the command line to control the manager, so we use the *Tkinter* library to create graphical user interfaces. We use is as the basic class to build our main *Login* window, *Add* window, *List* window and so on.

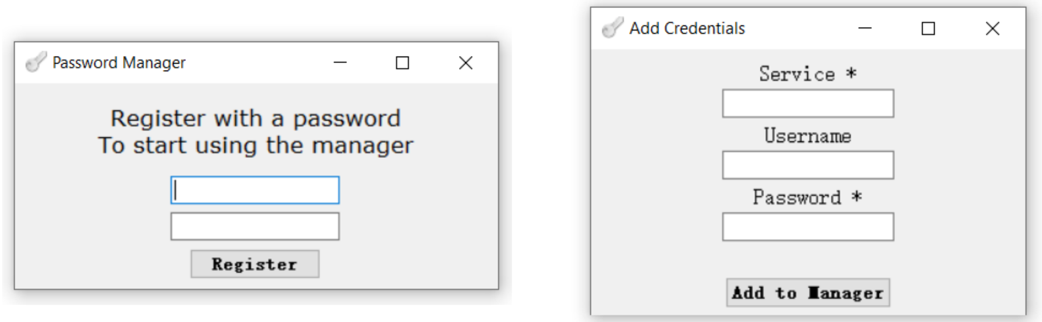


Figure 3 GUI implementation

What we plan to do:

1. Details about the *Encrypt* module

   Now we have just built a Caesar-like encryption for testing, while it is easy to be decrypted and not safe or complicated enough, not only for the password but also for this course. We want to explore more kinds of encryption algorithms like AES, RSA and so on, which will be the main part of our project. Along with the application in our program, discussion and comparison on several kinds of encryption algorithms will also be included in the following research.

2. Complete the other modules and add additional functions like *Autofill* and *Multiusers* if time permitted.

3. Test the separate modules and the whole program to make the project easier to maintain, extendable, and to ensure the reliability and stability of the application.