

Password Manager Documentation

COMP.SEC.300 Secure Programming project work, programming project

Henri Rantala

Table of contents

[1. General description 1](#_Toc165994912)

[1.1 Current stage and functionalities 1](#_Toc165994913)

[1.2 Structure of the program 1](#_Toc165994914)

[1.3 User interface 2](#_Toc165994915)

[2. programming solutions 3](#_Toc165994916)

[2.1 Security aspects 3](#_Toc165994917)

[2.2 Testing and security threats 4](#_Toc165994918)

[2.3 AI usage 4](#_Toc165994919)

[3. Future development and ideas 5](#_Toc165994920)

.

# General description

The purpose of the program is to function as password manager running locally on a device, which is capable of storing and reading encrypted passwords with chosen format. Passwords are saved to file, which can be safely distributed or saved to a cloud as passwords are encrypted and thus cannot be read without decrypting the file first. The encryption is done via master password which is used to generate the symmetric key that is used to encrypt and decrypt the file.

## Current stage and functionalities

Current version of the software is simple and straightforward in its functionality. It can encrypt and decrypt files of set formatting. At this stage only supported file format is .txt files, which must store the passwords in expected format. The software also stores the service and username related to the password used. The master password of existing file can be changed as well as individual passwords along with related information can be changed.

Besides storing passwords and related information, the program is capable of generating new passwords by generating a random combination of letters, numbers and special characters to form a secure password. The program has a simple GUI, which makes the program easier to use and helps in displaying the relevant and desired information.

## Structure of the program

Code itself is divided into two different files. These are gui.py, which is responsible for visual representation and UI, and main.py, which is responsible for general logic and function of the software like encryption and decryption. As this implementation is relatively simple and the scope quite small, there is little need for further division of the code. The program functions by starting the GUI from which the necessary functions are called based on user input.

## User interface

User interface is kept simple with essentially two different sections. On the left side of the window there is menu options as buttons from file opening to adding new password entries. These should be easy to understand with accordingly labelled buttons. The right side will display the saved passwords when file is opened and allows the editing of the information.

# programming solutions

The program was implemented using Python, with Tkinter as GUI library and PyCryptodome library for encryption. Also libraries included in Python like secrets and hashlib were utilized for generating cryptographically random sequences and hashing respectively.

## Security aspects

In order to safely store the password, encryption is used. The chosen method for this was AES256 encryption algorithm as AES is widely used and is practically the de facto standard method for symmetric encryption. The encryption key is derived from master password through key derivation algorithm PBKDF2, which widely used key derivation algorithm. While the encryption and key derivation algorithms are cryptographically secure, they cannot protect the encrypted passwords if master password is weak.

Choice to make locally run software without accounts or online features can make the software require more effort from the user and be less modern and clunkier solution but it also reduces some security threats. Security problems which are present in web software are greatly reduced as well as software complexity, like there is no need for session management or other attacks which need to be taken account in web environment. Also saving encrypted passwords to a file instead of database prevents injection attacks.

Of course, these solutions and design choices come with their own issues. One large is that these place the responsibility for the security to the user of the software. The security of the encryption relies greatly on the user to make a secure master password, which must not be easily guessed, or brute forced as if someone else gains access to the password file they have essentially all the time in the world to try to crack the encryption. While the file containing the encrypted passwords is safe to share through multiple devices through the internet, some consideration should be given how the user should do it, as careless sharing with weak or guessable password can lead to others decrypting the encrypted file contents.

## Testing and security threats

Testing focused mainly on the functionality of the software the most major security threats are user dependant. The testing has been manual, and all of the main functionalities have been tested to work. The main threats to security are all related to user behaviour and as such testing them is difficult.

The main threat to security in this system is weak master password. As of yet there is no policy to judge the strength of the master password or enforce any standards on what it contains. As such the password is as strong or weak as the user makes it and for now user is responsible for ensuring that the password is secure. An immediate priority is to implement some type of policy to ensure that at least some basic standards must be met by the master password before it is accepted.

## AI usage

Both copilot and chatgpt 3.5 were used during the implementation of this project. Especially in the implementation of the GUI, the AI proved to provide meaningful assistance in providing what components of the library should be used, how to use them and even generating some individual elements or their parts. The process of utilizing AI generated code was usually iterative, with using AI to add new components to the the code already made either by itself or myself, adding functionality or new elements incrementally instead of all in one as doing the latter often resulted in not getting the functionality or the visual result that was wanted.

Also, AI was used to find or explain how to use components with Tkinter to speed up the process as I did not have that much familiarity or experience with it. AI was used throughout the project to help with the syntax and sometimes helped in solving errors, mostly when already having a clue what was wrong but not how to solve it, like how to have the execution wait user input in another popup window.

However, with things like implementing the encryption and matters important to security, like implementing the encryption, the AI use was very limited to non-existent as documentation proved far more effective and detailed than what the AI could provide. Also, the inability to completely trust the AI and need to confirm things from documentation and provided examples in it made it not that useful as compared to thing like UI.

# Future development and ideas

List of possible new items:

* The software could in future integrate way to autofill fields in a browser, which was abandoned quite early in the development process of this project as it was deemed too time consuming.
* Adding compatibility with other open-source password managers like KeePass.
* Additional file format support
* Force users to make secure password, as currently anything is accepted.