Personal Password Manager

CS303 Final Project

Buster Shick
Aerospace Engineering Dept
ERAU
Daytona Beach, FL
shickb@my.erau.edu

I. This project presents the development of a Personal Password Manager, a lightweight Windows-based application designed to securely store and manage user login credentials. Implemented in Python, the system provides an intuitive graphical user interface that allows users to add, view, update, and delete password entries with ease. All sensitive information is protected using the Fernet symmetric encryption scheme from the *cryptography* library, ensuring that passwords are never stored in plaintext. Encrypted passwords and associated metadata are stored locally in a SQLite database, providing a simple yet reliable data management structure. The application also incorporates secure key-handling practices by separating the encryption key from the database file, thereby reducing the risk of credential compromise if local storage is accessed by unauthorized users. This project demonstrates how strong cryptographic techniques and user-focused interface design can be combined to create an accessible, secure, and efficient password management solution without the complexity of server-based systems or third-party services.

II. Introduction

Online security has become increasingly important as users manage a growing number of accounts across various platforms. Password reuse, weak password choices, and insecure storage methods continue to place users at heightened risk of unauthorized access. Many commercial password managers exist to mitigate these issues; however, some users prefer lightweight, offline, and transparent solutions that do not rely on cloud services or subscription models

The Personal Password Manager presented in this project addresses this need by providing a secure, Windows-based graphical application for storing, retrieving, and managing user credentials. Developed in Python with a graphical user interface, the application offers essential features such as adding, viewing, updating, and deleting password entries. To ensure confidentiality, all passwords are encrypted before being stored in a local SQLite database. This project demonstrates how cryptographic tools, user-focused interface design, and secure data handling can be combined to deliver a simple yet effective personal cybersecurity tool.

III. Background

Password mangers are designed to reduce the cognitive burden of memorizing numerous strong, unique passwords. They commonly use database storage combined with encryption techniques to protect sensitive data. In this project, secure password handling is achieved using the *cryptography* library's Fernet encryption protocol. Fernet is a symmetric encryption system built on AES and HMAC authentication, guaranteeing both confidentiality and integrity. Only users with the correct encryption key can decrypt the stored values, and tampering attempts are reliably detected

SQLite serves as the storage backend due to its low overhead and self-contained file-based architecture. It requires no external server and is widely used in desktop and mobile applications. However, because SQLite does not encrypt data by default, the application performs encryption before writing passwords to the database.

A graphical user interface was implemented using Tkinter, enabling the application to operate as a Windows desktop program rather than a command-line tool. This improves usability, accessibility, and overall user experience, particularly for users who may be unfamiliar with terminal-based software.

Password managers are designed to reduce the cognitive burden of memorizing numerous strong, unique passwords. They commonly use database storage combined with encryption techniques to protect sensitive data. In this project, secure password handling is achieved using the *cryptography* library's Fernet encryption protocol. Fernet is a symmetric encryption system built on AES and HMAC authentication, guaranteeing both confidentiality and integrity. Only users with the correct encryption key can decrypt the stored values, and tampering attempts are reliably detected.

SQLite serves as the storage backend due to its low overhead and self-contained file-based architecture. It requires no external server and is widely used in desktop and mobile applications. However, because SQLite does not encrypt data by default, the application performs encryption before writing passwords to the database.

A graphical user interface was implemented using Tkinter, enabling the application to operate as a Windows desktop program rather than a command-line tool. This improves usability, accessibility, and overall user experience, particularly for users who may be unfamiliar with terminal-based software.

IV. Methodology

The development approach for the Personal Password Manager involved several stages:

A. System Setup and Dependencies

- *cryptography* for Fernet encryption
- *sqlite3* for database operations
- *tkinter* for the graphical user interface
- *os* and *pathlib* for local key and file management

B. Encryption Key Management

A single Fernet key is generated upon first launch and stored in a secure file, separate from the SQLite database. This key:

- Required to decrypt all stored passwords
- Remains persistent to maintain consistent encryption across sessions
- Never stored in plaintext within the database

C. Database Architecture

The SQLite database includes one table containing:

- An auto-incremented ID
- Service name
- Username
- An encrypted password field stored as binary data

Only the password column is encrypted to facilitate readable searching and indexing by service.

D. Graphical User Interface Design

The GUI was developed using Tkinter to enable:

- Input fields for service, username, and password
- Buttons for adding, updating, and removing entities
- Table-like display for saved entries
- Selective decryption of passwords on demand

The interface emphasizes simplicity, clarity, and ease of use.

E. Functional Integration

Backend logic for encryption, database operations, and key handling was linked to GUI event functions. Each user action triggers validation, database interaction, and display updates. This ensures secure and seamless operation throughout the application workflow.

V. Results and Testing

Testing confirmed that the Personal Password Manager meets it functional and security requirements.

A. Function Verification

- Entries were successfully added, update, displayed and deleted.
- ID numbering remained consistent and auto incrementing
- Passwords displayed as ciphertext when viewed directly in the database

B. Encryption Validation

- Passwords were encrypted using Fernet prior to storage
- Attempting to decrypt entries with an incorrect key produced authentication failures, confirming the integrity protection
- Passwords remained unreadable when the database file was viewed independently

C. GUI Testing

- All interface buttons performed the intended actions
- Input validation prevented incomplete submissions
- Decrypted passwords were displayed only when explicitly requested

D. Security Evaluation

- Copying the database to another system without the key rendering all passwords inaccessible
- Removal or corruption of the key file prevented decryption, demonstrating proper dependency on the encryption key
- The separation of the key from database reduced attack surface risks

Overall, the system performed reliably and demonstrated strong data protection.

VI. Conclusion

The Personal Password Manager successfully delivers a secure, offline, and user-friendly solution for managing personal login credentials. By integrating Fernet encryption, SQLite storage, and a Tkinter-based graphical interface, the project combines modern security practices with accessible design. All user password data is stored securely, encrypted at the application layer, and protected through strict key-handling procedures.

The project highlights the importance of encryption-by-default, minimal reliance on external systems, and clear interface design in personal cybersecurity applications. Future enhancements could include password generation features, master-password protection, encrypted database formats such as SQLCipher, or secure cloud synchronization with end-to-end encryption. Nevertheless, the current implementation provides a strong foundation for secure and convenient password management on Windows systems.