Krypta – Encrypted Password Manager

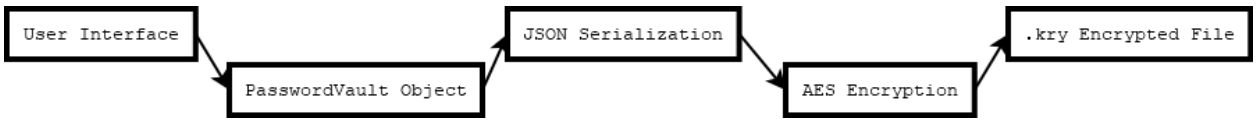
Balázs Krisztián-Zsombor

Contents

Introduction

Krypta is a Windows desktop password manager that securely stores and organizes user credentials in encrypted local vaults. It combines modern security practices with an accessible, lightweight interface built using **C#** and **Windows Forms**.

The application allows users to create new password vaults, open existing ones, edit stored entries, and save them in a custom **.kry** format. Each vault file is fully encrypted using the **AES (Advanced Encryption Standard)** algorithm with keys derived from a master password through **PBKDF2**. This ensures that all sensitive data remains protected even if the file is accessed directly.



Krypta is designed with both **security** and **ease of use** in mind. The main interface displays credentials in a clean, table-based view where users can add, modify, or remove entries such as website names, usernames, passwords, and notes. It also includes practical tools like a built-in **password generator**, a **strength meter** for evaluating password quality, and a **search bar** for quickly locating stored entries.

To improve safety, Krypta includes an **auto-lock feature** that secures the vault when the application loses focus, as well as manual locking and unlocking options.

The program also provides clear visual feedback, including a color-coded status bar and masked passwords to help users stay aware of the vault's state.

Overall, Krypta demonstrates how strong encryption, responsible data handling, and thoughtful interface design can work together to deliver a secure and intuitive password management experience.

Objectives and Motivation

Krypta is built as a **self-contained desktop password manager** that offers complete local control over stored data. It provides a familiar environment similar to professional password managers while remaining lightweight, transparent, and easy to use.

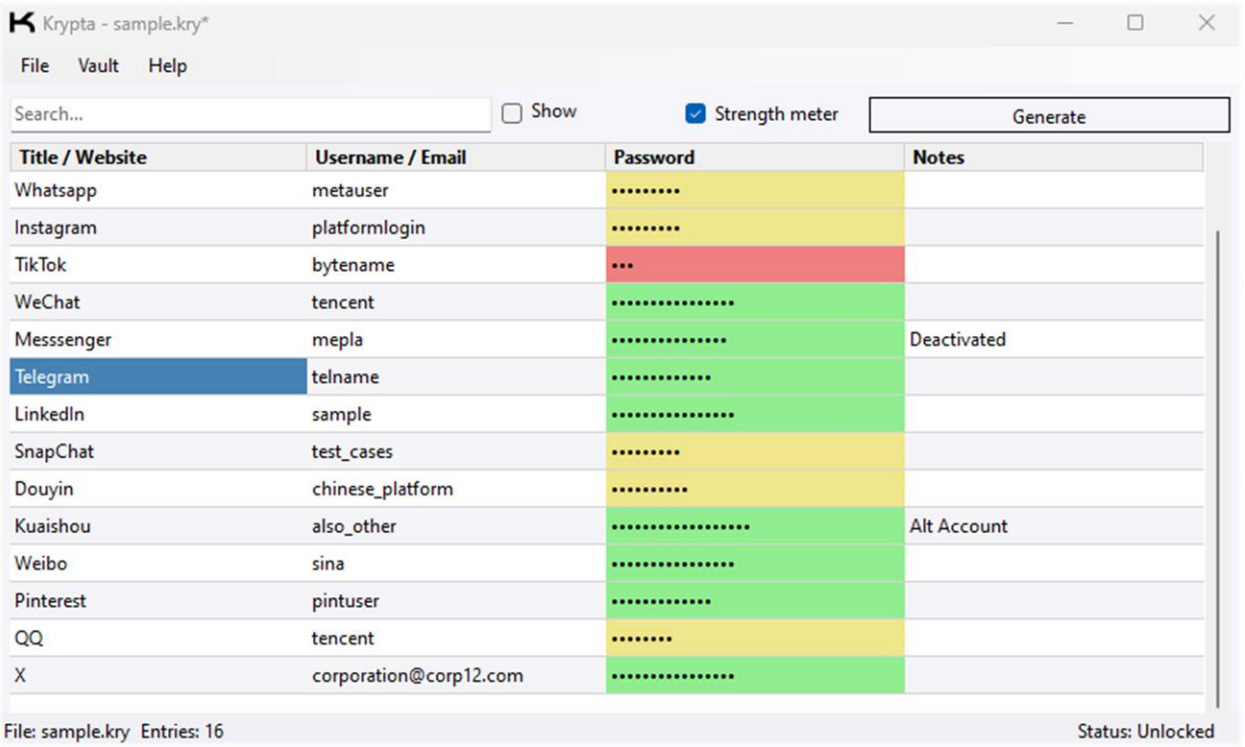
At its core, Krypta maintains a collection of **password entries**, each consisting of a title, username, password, and notes. Users can freely create, edit, delete, and search through these entries. The application ensures that all changes are tracked: unsaved modifications trigger visual indicators and prompts to prevent accidental data loss.

Every saved vault is encrypted end-to-end. The program never writes unencrypted passwords to disk, and the vault is cleared from memory whenever it locks or closes. This approach prioritizes **data confidentiality** without introducing unnecessary complexity.

In addition to essential security functions, Krypta emphasizes a practical and user-friendly design:

- A **menu system** for all common file and vault actions (New, Open, Save, Lock, Unlock, etc.).
- A **status bar** displaying the current file name, number of entries, and lock status, with the lock indicator highlighted in red when active.
- Keyboard **shortcuts** for faster access to key commands (e.g., Ctrl+S to save, Ctrl+L to lock).
- A clear, flat-styled **DataGridView** for readability.
- A responsive layout that resizes cleanly with the window.

By combining encryption, usability, and a focused interface, Krypta delivers a practical solution for securely storing and managing passwords offline, entirely under the user's control.



System Overview

The program structured around a simple yet robust architecture that separates **data management**, **encryption**, and **user interface** logic. The design focuses on clarity, modularity, and maintainability, each part of the system handles a distinct responsibility.

Component	Responsibility
Main Form (Form1)	The graphical interface; manages user interaction, menus, and display of password entries.
PasswordVault	Represents the collection of stored passwords. Acts as the root data model.
PasswordEntry	Defines a single credential record (Title, Username, Password, Notes).
VaultFileService	Handles reading and writing of vault files; performs encryption and decryption.
Encryption Layer (AES + PBKDF2)	Secures vault data before saving and decrypts it upon loading.
UI Components	Include the menu system, DataGridView, search bar, checkboxes, and status bar.

The main workflow revolves around loading, editing, and saving encrypted vaults:

1. **Vault Creation or Opening**
    - When a user creates a new vault or opens an existing one, the application initializes a PasswordVault object.
    - If a file is opened, the data is decrypted and deserialized into in-memory entries.
2. **Editing**
    - Users can add, modify, or delete entries directly within the DataGridView.
    - Any change marks the vault as “dirty” (unsaved), reflected by an asterisk in the title bar and in the status bar.
3. **Saving**
    - When saved, the vault is serialized to JSON, encrypted using AES, and written to disk as a .kry file.
    - Encryption keys are derived from the master password using PBKDF2, ensuring consistent and secure file protection.
4. **Locking and Unlocking**
    - Krypta locks automatically when the window loses focus (if saved), or manually through the Vault menu.
    - When locked, the vault data is cleared from memory, the grid is emptied, and the status bar turns red (“Status: Locked”).
    - Unlocking prompts for the master password and reloads the decrypted data.
5. **Search and Filtering**
    - The search box dynamically filters entries in real time.
    - Matches are checked against the Title, Username, and Notes fields.
6. **Password Strength and Generation**
    - Each entry’s password field is color-coded based on estimated strength.
    - The “Generate” button produces a random, secure password using the system’s cryptographic RNG.

Features

Krypta provides a complete set of tools for securely storing, organizing, and protecting passwords in a local, encrypted vault.

Core

Category	Features
Vault Management	Create, open, save, save as, close, and exit vaults (.kry files).
	Tracks unsaved changes with an asterisk (*) and save prompts.
	Auto-locks on losing focus and clears vault data.
	Manual lock/unlock available
Encryption	AES-256 encryption with PBKDF2 key derivation.
	Encrypted .kry format containing serialized JSON data.
	Master password required to open and decrypt vaults.

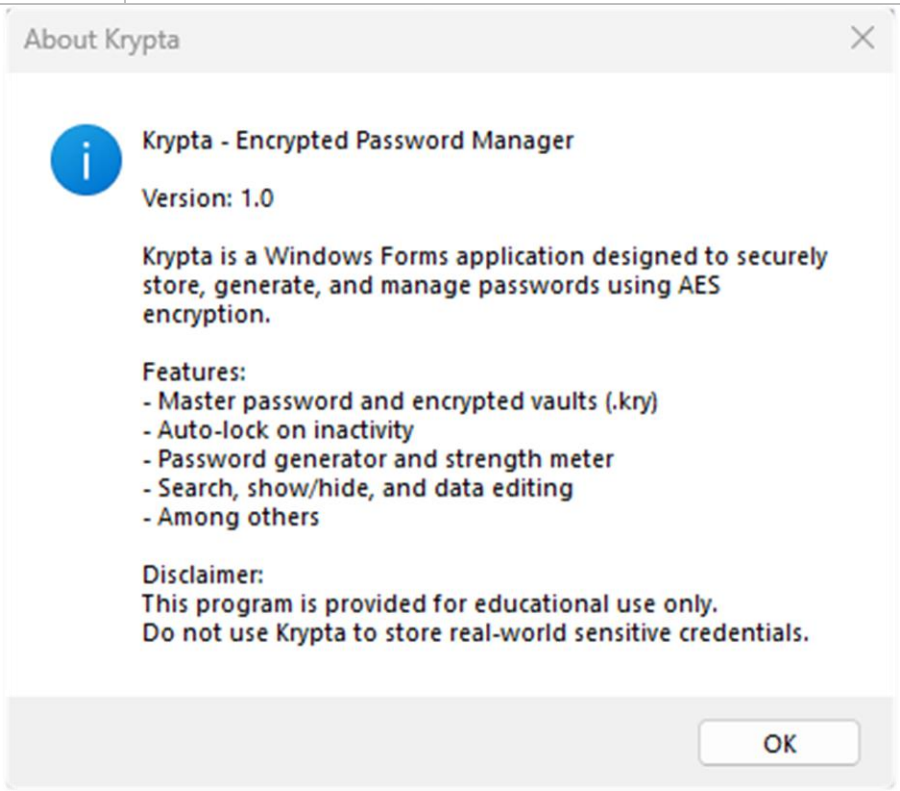
Password Handling

Category	Features
Entry Management	Add, edit, or delete password entries directly in the table.
	Fields: Title / Website, Username / Email, Password, Notes.
	Deletion confirmed before removal to prevent mistakes.
Search & Filtering	Real-time filtering through the search box (Title, Username, Notes).
	Clears automatically when the search field is empty.
Password Tools	Show/Hide passwords toggle.
	Password strength meter with color coding (Weak–Red, Medium–Yellow, Strong–Green).
	Password generator with automatic clipboard copy.

User Interface

Category	Features
Layout	Clean, flat-styled DataGridView with auto-sizing columns.

	Alternating row colors and readable fonts.
Menus	Organized File / Vault / Help structure.
	Status bar shows file name, number of entries, and lock state (red when locked).
Dialogs & Messages	Confirmation dialogs prevent unintended actions.
	About Krypta dialog (Help - About) shows version and developer info.
Behavior	Auto-lock avoids data loss by skipping lock if unsaved changes exist.
	Fully offline – no network communication.



Keyboard Shortcuts

Action	Shortcut
New Vault	Ctrl + N
Open Vault	Ctrl + O
Save Vault	Ctrl + S
Save As	Ctrl + Shift + S
Close Vault	Ctrl + W

Lock Vault	Ctrl + L
Unlock Vault	Ctrl + U
Delete Entry	Delete
Search / Focus Filter	Ctrl + F
About Krypta	F1

Limitations and Future Improvements

While all of this combined provides a secure and practical approach to local password management, it remains a lightweight, single-user application with deliberate design simplifications. The following points outline its current limitations and potential areas for future development.

Current Limitations

- **Local-only storage:** Krypta operates entirely offline; vaults are stored locally without cloud sync or multi-device support.
- **Single-user vaults:** The program does not support multiple user profiles or shared vaults.
- **Basic encryption integrity:** Vault files are encrypted but do not include checks for tampering or data integrity verification.
- **Limited password analysis:** The strength meter uses simple heuristic scoring and does not perform advanced entropy or dictionary checks.
- **Clipboard handling:** Generated passwords are copied to the clipboard but are not automatically cleared afterward, which may expose temporary data if left unattended.
- **Manual backup:** The application does not include automated backup or versioning of vault files.
- **No dark/light theme switching:** Interface customization is limited.

Possible Future Improvements

- **Enhanced password scoring:** Introduce entropy-based or zxcvbn-style strength evaluation with detailed feedback.
- **Cloud synchronization:** Optional secure cloud backup using encrypted transmission and remote authentication.
- **Automatic clipboard clearing:** Clear copied passwords after a configurable timeout for better privacy.

- **Integrity verification:** Add authenticated encryption or checksum validation to detect tampering.
- **User preferences:** Implement theme support, configurable auto-lock timeout, and custom generator rules.
- **Database or cloud backend:** Migrate to SQLite or encrypted cloud-based storage for scalability.
- **Password expiry reminders:** Notify users optionally when passwords exceed a defined age threshold.
- **Multi-language interface:** Add localization support for broader accessibility.

These enhancements would improve both security depth and user experience, allowing Krypta to evolve from a personal learning project into a more production-ready password manager.

Conclusion

Krypta demonstrates how a desktop application can combine usability with data protection through encryption and thoughtful design. By integrating AES-based encryption, file serialization, auto-locking, and clear visual feedback into a single cohesive environment, it provides a realistic example of secure application architecture within the Windows Forms framework.

The project successfully achieves its goals of protecting user credentials, providing intuitive management tools, and maintaining simplicity without relying on external dependencies. Although designed primarily for educational and demonstration purposes, Krypta embodies key principles of modern secure software: confidentiality, usability, and maintainability.