PROJECT ABSTRACT

Secure Password Manager with AES Encryption

Password reuse is the root cause of most account takeovers, yet people still memorize a handful of weak passwords.

01 Title

Secure Password Manager with AES Encryption

02 Introduction / Background

Password reuse is the root cause of most account takeovers, yet people still memorize a handful of weak passwords. A password manager solves this — one strong master password protects a vault of unique, generated credentials. Building one is an ideal final-year security project: it forces the student to implement real cryptography correctly (key derivation, IVs, authenticated encryption) and to reason about threat models.

03 Problem Statement

Users reuse passwords across sites, so one breach compromises everything. Commercial managers are closed-source black boxes the user must trust. This project builds VaultKey — an open, examinable password manager with an AES-256 encrypted vault, a zero-knowledge design where the server sees only ciphertext, a strong generator, breach warnings and a browser extension for autofill.

04 Objectives

- AES-256 encrypted credential vault with per-entry random IVs.
- Master-password key derivation (PBKDF2) — password never stored.
- Zero-knowledge architecture: client-side encryption, ciphertext-only server.
- Cryptographically strong password generator with entropy display.
- Breach-check warnings against compromised-password datasets.
- Browser extension with login-form autofill.

05 Proposed System / Working

Built in Python or Node.js with AES-256 (Fernet/WebCrypto, GCM preferred) and PBKDF2-SHA256 key derivation with a random salt and 600k+ iterations. Each vault entry is encrypted with a fresh random IV; ciphertext blobs are stored in SQLite locally and optionally synced to a server that never receives keys or plaintext. The generator uses a cryptographically secure RNG with

configurable charset and an entropy estimate in bits. The breach checker compares hash prefixes against known-compromised datasets without exposing passwords. A vanilla-JS Chrome/Firefox extension detects login forms and autofills after unlock. Auto-lock and clipboard clearing limit exposure; the report documents the cryptography choices, threat model and test cases.

The demo creates a vault, saves logins, generates a high-entropy password, flags a weak reused password in the health dashboard, and autofills a demo login page via the extension — then inspects the server store to show it holds only undecryptable ciphertext. The full flow runs in one session.

06 Technologies / Components Used

Python/Node.js · AES-256 (Fernet/WebCrypto) · browser extension (JS) · SQLite

07 Key Features

- AES-256 (GCM) vault with per-entry random IVs.
- PBKDF2 key derivation; master password never stored or transmitted.
- Zero-knowledge server design — ciphertext only.
- CSPRNG password generator with entropy readout.
- Breach-check warnings and password-health dashboard.
- Browser extension autofill with auto-lock and clipboard clearing.

08 Applications / Use Cases

- Personal credential management
- student and developer workflows
- small-team shared secrets (with the sharing extension)
- as a teaching artifact for applied-cryptography courses

10 Conclusion

VaultKey is a complete, examinable security application — real AES-256 encryption, zero-knowledge design and a working browser extension — with a threat model the student can defend confidently in the viva.