A Secure Password Manager Platform for Sensitive Data Protection

Teodor-Mihail GULUȚĂ and Ion BICA

***Abstract*—This paper presents the design and implementation of a modular password manager platform with end-to-end encryption, aimed to protect sensitive information such as passwords, banking information, and personal notes. The novelty of the solution lies in its architecture that integrates a secure web application with a browser extension to offer seamless usability, password strength evaluation, and automated form completion. Key contributions include a zero-knowledge encryption model, password health monitoring, integration with breach detection services, and compatibility across platforms. The proposed system addresses both cybersecurity threats and usability challenges, meeting industry standards and ensuring high level of protection of users sensitive data.**

***Index Terms*—browser extension, password manager, secure storage, zero-knowledge encryption.**

I. Introduction

In today's digital landscape, authentication credentials are a frequent target of cyberattacks, often being the weakest link in security chains. Despite growing awareness, many users still tend to reuse passwords across multiple platforms or choose weak and easy-to-remember combinations [1-2]. Conversely, complex passwords—while more secure—are often forgotten, leading to frustration and security workarounds [3]. These behaviors increase the risk of unauthorized access and data breaches, especially in environments that require stringent data protection, such as critical infrastructure systems [4-5].

To address these challenges, we present EnginePassword, a modular password manager platform designed to provide robust end-to-end security while remaining intuitive and efficient for daily use. The solution is composed of two integrated modules: a secure web application for managing and organizing sensitive data, and a browser extension for seamless auto-completion, password generation, and threat detection during navigation [6]. The platform adheres to modern cryptographic standards and aims to balance security with usability, helping users adopt safer password practices without compromising convenience.

The main contributions of this paper are summarized as follows.

1. We designed and implemented a modular password manager platform composed of a web application and a browser extension, that are using end-to-end encryption (AES-GCM with PBKDF2 key derivation).
2. We implemented strong password generation and evaluation mechanisms using zxcvbn (realistic password strength estimation) [7], along with breach detection via HIBP integration.
3. We introduced a novel PIN-based session key mechanism to decrypt locally stored data without exposing the master key in memory.
4. We enabled auto-fill capabilities for login forms and password management directly within the browser, with breach and phishing detection.
5. We supported secure storage and retrieval of cards, notes, and SSH credentials, following strict compartmentalization principles.
6. We implemented a feature for remote SSH access using PuTTY directly from the web application, allowing users to initiate secure terminal sessions without manual credential entry.

The implementation discussed in this paper is publicly available on GitHub at the following repository: https://github.com/teodorguluta16/Password_Manager.git.

II. Related Work

Numerous commercial solutions exist (e.g., Bitwarden, 1Password, NordPass, Keeper), each offering a set of features aimed at balancing security and usability [8]. While these platforms provide secure password storage, most rely on centralized architectures and offer limited client-side control over encryption and key management workflows. For instance, Keeper offers advanced breach detection and compliance tools, but its password sharing mechanism is primarily tied to account-level permissions and lacks fine-grained, client-controlled encryption per shared item.

This work draws from principles outlined in NIST SP 800-63B [9] and extends upon findings from [6] and [10] by integrating advanced usability with zero-knowledge security. Unlike many commercial solutions, EnginePassword performs fully local encryption and decryption, never exposing the master key to the server or transmitting decrypted content.

Moreover, our platform introduces asymmetric encryption mechanisms (similar to PGP) for secure item sharing, allowing users to share passwords, notes, and SSH credentials individually—each encrypted with the recipient's public key and optionally signed with the sender's key. This ensures end-to-end confidentiality and integrity, even in collaborative or multi-user environments.

A notable example that highlights the importance of local cryptographic control is the LastPass breach [11], where the attackers exfiltrated encrypted vaults but took advantage of weak key derivation parameters—PBKDF2 with only 100,000 iterations for some users. This incident

T. M. GULUȚĂ is with the Military Technical Academy "Ferdinand I", Bucharest, Romania (e-mail: teodor.guluta@mta.ro).

I. BICA is with the Military Technical Academy "Ferdinand I", Bucharest, Romania (e-mail: ion.bica@mta.ro).

Digital Object Identifier 10.32754/JMT.2026.1.02

demonstrates that inadequate client-side key strengthening and server exposure can critically weaken a system's resilience to offline attacks. EnginePassword enforces high iteration counts and performs all cryptographic operations in the client's browser.

III. METHODS

EnginePassword is structured as a two-module system: a web application backend with a secure frontend client, and a browser extension for in-page interaction. The architecture prioritizes zero-knowledge encryption, local decryption, and secure key handling on the client side.

A. Cryptographic Architecture

Sensitive user data—including passwords, notes, cards, and SSH credentials—is encrypted locally in the browser using AES-256-GCM. The encryption key is derived from the user's master password via PBKDF2 with 500,000 iterations and a cryptographically secure random salt. The derived key is never stored or transmitted. Each item (password, note, card, etc.) is encrypted with its own unique AES-GCM data key, which is itself encrypted using the derived master key. To additionally ensure password integrity, each encrypted password field is accompanied by an HMAC signature generated using a key derived from the master key. This HMAC is verified on decryption to detect any unauthorized tampering with the encrypted content. This layered approach supports forward secrecy and facilitates selective sharing. Encrypted items are stored in the database in JSON format, containing the encrypted data (encData), the initialization vector (iv), and the authentication tag (tag) for each field. An example of a stored item is shown in Listing 1.

This format ensures field-level encryption and authentication, offering high confidentiality and integrity guarantees even if the database is compromised.

B. Session Management with PIN-Derived Key

To improve usability, EnginePassword supports a PIN-based mechanism to temporarily unlock the session. At login, the master key is encrypted using a session key derived from a user-defined PIN and stored locally (e.g., in IndexedDB). Upon page refresh, users only need to re-enter the PIN to re-unlock the session. The session key is cleared from memory when the user logs out or the session times out.

C. Password Generation and Evaluation

The application includes a cryptographically secure password generator that uses *crypto.getRandomValues()* to ensure strong randomness. Users can customize the password length, character sets (uppercase, lowercase, digits, symbols), and apply exclusion rules (e.g., avoid ambiguous characters). To evaluate password strength, we integrate the zxcvbn library[2], which provides an entropy-based score from 0 (weak) to 4 (strong). This score reflects how easily the password could be cracked using known patterns, dictionary attacks, or user-specific guesses. Passwords with 12 or more characters from diverse character sets consistently reach scores of 3.5–4.0. The browser extension includes an embedded version of the zxcvbn evaluator and a local copy of compromised passwords extracted from Have I Been Pwned's downloadable dataset [12]. This enables real-time password evaluation and breach detection directly on the client side, even offline, without leaking password hashes to external services.

```
{
  "metadata": {
    "created_at": "2025-05-19T06:11:14.280Z",
    "modified_at": "2025-05-19T06:11:14.281Z",
    "modified_parola": "2025-05-19T06:11:14.281Z",
    "version": 1,
    "meta": {"lungime": 32,
          "charset":
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz01
23456789!@#$%^&*()_+-=[]{}|;:,.<>?"
  }
  },
  "data": {"tip": {
      "iv": "3175...d182", "encData": "44e5...b327",
      "tag": "7692...d6454"
    }, "nume": {
      "iv": "00e7e3...4469", "encData": "499040",
      "tag": "1c1c...aaab"
    }, "url": {
      "iv": "bea9...1992", "encData": "6357...41c9",
      "tag": "3b1f...58052"
    }, "username": {
      "iv": "830b...df86", "encData": "4181...2ff7",
      "tag": "6746...8646"
    }, "parola": {
      "iv": "ebc1...41b3", "encData": "0d5e...22bf",
      "tag": "9fdb...c5bc4"
    },"semnatura": {
      "iv": "deec...08a3","encData": "999f07...39a9",
      "tag": "c43e...fac1"
    }, "comentariu": {
      "iv": "0284...7801","encData": "cf43...6abf",
      "tag": "d918...4142"
    }
  }
}
```

Listing 1. Example of a stored item showing field-level AES-GCM encryption with unique IVs and authentication tags for each field

D. Breach Detection and Password Health

The browser extension interacts in real-time with the web application. It detects forms on visited pages and suggests filling in credentials. The detection mechanism scores each form based on a heuristic evaluation of input fields (type, id, name, placeholder) and checks for keywords like "username", "email", "password", "confirm", etc. Based on the score:

- Score ≥ 4 → Registration form
- Score < 4 and presence of keywords like "old", "confirm", "reset" → Reset password form
- Otherwise → Login form

This scoring allows accurate classification and autofill behavior. The extension and web app are synchronized bidirectionally. Any item changes (add, edit, delete) are reflected in both components with minimal delay.

E. Auto-Fill and Content Script Integration

The browser extension injects content scripts into login and registration pages, identifying input fields based on heuristics and semantic cues. It supports secure auto-fill of credentials, form detection, and in-field password generation. All operations are performed client-side, with

[2] https://github.com/dropbox/zxcvbn

 Digital Object Identifier 10.32754/JMT.2026.1.02

strict origin and context validation to prevent injection attacks.

To further strengthen security, the extension enforces a strict Content Security Policy (CSP) that blocks execution of unsafe scripts by disallowing the use of eval, Function, and assignments to innerHTML. This minimizes the attack surface for Cross-Site Scripting (XSS) attacks. Additionally, when the web application communicates with the extension via *window.postMessage()*, messages are filtered and accepted only if they originate from a predefined trusted origin. This prevents other websites or tabs from sending unauthorized messages to the extension.

F. Secure Sharing via Asymmetric Encryption

EnginePassword allows users to share individual items securely using PGP-like asymmetric encryption. Each user has a keypair (public/private), and shared items are encrypted with the recipient's public key and signed with the sender's key for authenticity. The private key is stored encrypted with the master key and only decrypted locally.

G. Remote Access Integration with PuTTY

The platform supports launching PuTTY sessions from within the web interface. A lightweight helper server (a small executable) is downloaded and run locally on the user's device to handle remote session initialization. SSH credentials are decrypted locally within the browser and passed securely to the helper server via a local API or IPC mechanism. These credentials are stored temporarily in memory during the session and are securely erased once the connection is closed or the session expires. This setup automates secure login to remote systems without requiring manual credential entry while maintaining local control and high confidentiality. This modular and layered approach ensures data confidentiality, integrity, and usability while maintaining compatibility across modern browsers and devices.

H. Item History and Synchronized Modification

EnginePassword maintains a complete history of all modifications made to each stored item, allowing users to track changes and restore previous versions when needed. Additionally, users can edit or update stored items directly from either the web application or the browser extension. Both modules are synchronized in real time to ensure consistency and immediate propagation of changes, regardless of where the operation is performed. This modular and layered approach ensures data confidentiality, integrity, and usability while maintaining compatibility across modern browsers and devices.

IV. RESULTS

The main interface of the EnginePassword web application is shown in Fig. 1, illustrating the layout used for managing and navigating encrypted items.

The EnginePassword platform was evaluated in a local deployment with emphasis on performance, functionality, and security guarantees.

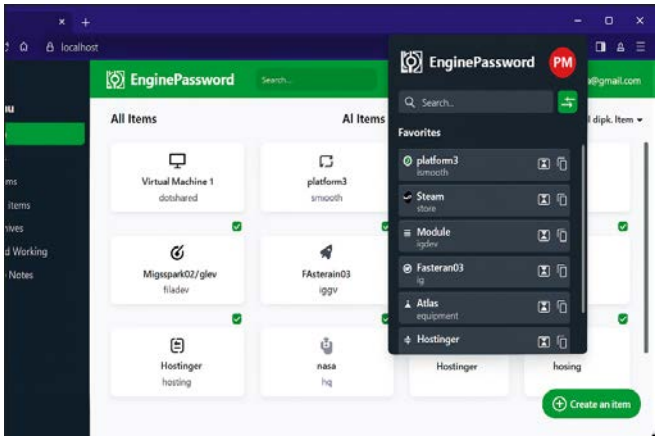

Figure 1. The main interface

- **Encryption/Decryption Performance:** AES-256-GCM encryption/decryption with HMAC verification consistently took under 5ms per item on modern desktop browsers.
- **Session Handling:** The PIN-based session unlock mechanism successfully re-derived the master key in memory, using the locally stored encrypted key in IndexedDB. The decryption via PBKDF2 (20,000 iterations) was completed in under one second, and the key was not exposed post-session. Refresh persistence and logout behavior were handled securely.
- **Form Detection Accuracy:** The form classification algorithm successfully identified nearly all login, registration, and password reset forms in test cases.

  Forms were scored using a heuristic-based mechanism that inspected input attributes such as type, name, id, and placeholder. Based on the presence of keywords like "username", "password", "email", and secondary fields like "confirm" or "old password", each form was categorized with high accuracy. The autofill module used this classification to offer targeted suggestions and complete login or registration fields automatically.
- **Password Generation**: Generated passwords with 12+ characters and full character sets consistently scored 3.5–4.0 on the zxcvbn scale.

  An example of the weak-password detection workflow is presented in Fig. 2.
- **Breach Detection**: Have I Been Pwned integration successfully flagged all test credentials present in its dataset.
- **Extension Synchronization**: Updates made in either the extension or the web application were reflected in both components with high consistency.
- **SSH Integration**: The PuTTY helper launched correctly from the web interface. Temporary decrypted SSH credentials were used securely and deleted from memory after session termination.

  The integration of the PuTTY-based remote SSH

workflow is shown in Fig. 3, where decrypted credentials are securely injected into a temporary local session.

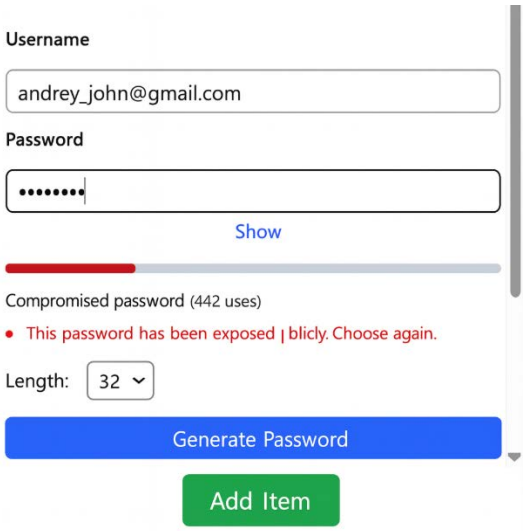

Figure 2. Weak password detection

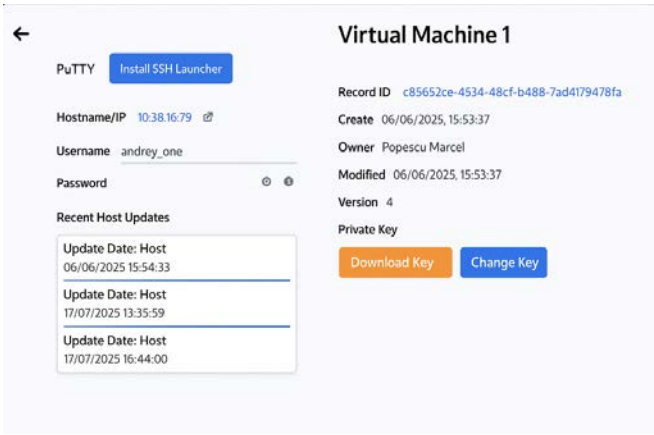

Figure 3. Remote SSH connection

- **Item History**: Modifications to items were tracked and could be reviewed in-app, supporting transparency and rollback.

These results confirm that the platform upholds secure design principles while offering responsive, reliable, and synchronized user interactions across all modules.

V. Conclusion

EnginePassword demonstrates that it is possible to build a password manager that ensures a high level of security while offering excellent usability and cross-platform flexibility. The integration of strong client-side encryption, PIN-based session key recovery, and real-time form detection through heuristics allows for a smooth yet highly secure experience. By combining a web interface with a browser extension, the system provides seamless autofill and synchronization without compromising privacy.

Unlike many commercial alternatives, EnginePassword ensures that the master key is never exposed or stored in plaintext and leverages modern cryptographic practices, including AES-GCM, HMAC verification, and PBKDF2-based key derivation. The platform also promotes password hygiene through strength scoring and breach detection. The proposed solution proves the feasibility of decentralized, zero-knowledge password management systems and lays the groundwork for future expansions involving biometric authentication, multi-device sync, or encrypted cloud backup—all under strict user control and cryptographic guarantees.

References

[1] R. Wash and E. Rader, "Prioritizing security over usability: Strategies for how people choose passwords," *J. Cybersecur.*, vol. 7, no. 1, pp. 1–17, Feb. 2021. doi:10.1093/cybsec/tyab012

[2] A. Mathews and S. M. Taiabul Haque, "Exploring the risks of password reuse across websites of different importance," in *Human Interaction and Emerging Technologies (IHIET 2024)*, vol. 157, 2024, pp. 105–115. doi:10.54941/ahfe1005469

[3] R. Shay *et al.*, "Designing password policies for strength and usability," *ACM Trans. Inf. Syst. Secur.*, vol. 18, no. 4, pp. 1–34, May 2016. doi:10.1145/2891411

[4] M. Dowd, J. McDonald, and J. Schuh, *The Art of Software Security Assessment: Identifying and Preventing Software Vulnerabilities*. Boston, MA: Addison-Wesley Educational, 2006.

[5] P. Langlois, "2020 Data breach investigations report," *Verizon*, 2020. [Online]. Available: https://www.cisecurity.org/wp-content/uploads/2020/07/The-2020-Verizon-Data-Breach-Investigations-Report-DBIR.pdf.

[6] C. Herley and P. Van Oorschot, "A research agenda acknowledging the persistence of passwords," *IEEE Secur. Priv.*, vol. 10, no. 1, pp. 28–36, Jan. 2012. doi:10.1109/MSP.2011.150

[7] D. L. Wheeler, "zxcvbn: low-budget password strength estimation," in *Proceedings of the 25th USENIX Security Symposium (SEC'16)*, Austin, TX, Aug. 10-12, 2016, pp. 157–173.

[8] P. Gallus, D. Stanek, and I. Klaban, "Security evaluation of password managers: A comparative analysis and penetration testing of existing solutions," *ICCWS*, vol. 20, no. 1, pp. 105–113, Mar. 2025. doi:10.34190/iccws.20.1.3330

[9] D. Temoshok *et al.*, "Digital identity guidelines: Authentication and authenticator management," National Institute of Standards and Technology, Gaithersburg, MD, Jul. 2025. doi:10.6028/NIST.SP.800-63B-4

[10] M. S. Turan, E. B. Barker, W. E. Burr, and L. Chen, "Recommendation for password-based key derivation: Part 1: Storage applications," National Institute of Standards and Technology, Gaithersburg, MD, Dec. 2010. doi:10.6028/NIST.SP.800-132

[11] J. Gentles, M. Fields, G. Goodman, and S. Bhunia, "Breaking the vault: A case study of the 2022 LastPass data breach," *arXiv [cs.CR]*, 2025. doi:10.48550/arXiv.2502.04287

[12] T. Hunt, "Introducing 306 million freely downloadable pwned passwords," *Troy Hunt*, Aug. 2017. [Online]. Available: https://www.troyhunt.com/introducing-306-million-freely-downloadable-pwned-passwords/.