ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 13, Issue 6, June 2026

Privacy Preserving Password Manager with Zero-Knowledge Proof Authentication

[1] Sushant Ambastha, [2] Abhishek Bajetha, [3] Abhijit Kumar, [4] Dr. Neetu Sharma

[1] [2] [3] [4] Department of Computer Applications & Technology (SCAT), Galgotias University, Greater Noida, Uttar Pradesh, India

*Corresponding Author Email: [1] sushantambstha@gmail.com, [2] abhishekbajetha27@gmail.com, [3] abhijitkumar.fakila667@gmail.com, [4] neetush75@gmail.com

Abstract*— Password managers serve an essential function in contemporary digital networks because they protect user passwords across different websites. The centralized storage and trust-based authentication methods that traditional systems use create security weaknesses which lead to data breaches and privacy violations. ZKPass delivers a password management solution which protects user privacy through its implementation of Zero-Knowledge Proof (ZKP) protocols that eliminate the requirement for servers and verifiers to access user passwords. Users can demonstrate their identity through this system which protects all of their personal data while keeping their authentication process secure. ZKPass protects local password storage through complete end-to-end encryption while using cryptographic elements to stop unauthorized access during server breaches. The proposed architecture minimizes attack surfaces through its implementation of secure key derivation and hash-based commitment schemes and session-based proof verification [1]. The user credential management system provides enhanced security features while giving users complete authority over their credential information. The experimental tests show that the system provides strong authentication security while consuming very little processing power. The foundation of ZKPass establishes Next Generation password management systems which combine user-friendly design and high operational efficiency and strong privacy protection through Zero-Knowledge authentication [2].

Keywords: Zero-Knowledge Proof, Password Manager, SRP Protocol, Cryptography, End-to-End Encryption, Authentication.

I. INTRODUCTION

1.1 Background

Password managers have become essential security tools for digital protection because more than 94 million people worldwide started using these services in 2024 [3]. The rising number of major data breaches, which result in financial losses of approximately $4.88 million per breach, creates an urgent requirement for effective credential management systems. Recent

statistics show that 81% of corporate hacking breaches result from weak or reused passwords [4], while data breaches in 2022 exposed about 24 billion passwords. The current cybersecurity demands require organizations to implement advanced solutions that exceed standard password management methods [5].

1.2 Problem Statement

The basic design of traditional password management systems contains essential flaws which lead to security breaches that affect multiple users. The system depends on a trust-based design which makes the server function as the main point that can be attacked. When servers suffer breaches attackers obtain access to encrypted user vaults which they use to conduct offline brute- force master password attacks. The LastPass breach of 2022 demonstrates this security weakness because attackers used keylogging malware to capture master passwords while they exfiltrated encrypted vaults [6]. Even with strong encryption standards like AES-256 protection users data remains accessible to both server administrators and malicious insiders [7]. The traditional systems store password-equivalent data on servers which creates a fundamental security flaw that encryption methods cannot entirely eliminate [8].

1.3 Proposed Solution: ZKPass

ZKPass implements a zero-trust security model which transforms how password managers protect their users. The system operates on a fundamental principle which states that servers must not store any data that matches the value of user passwords [9]. Users can authenticate their identities and access their encrypted vaults because the system lets them do so without disclosing their master passwords to the server. This solution protects against essential security flaws which modern password breaches exploit while providing users with the easy access they need to synchronize their data through cloud services [10].

1.4 Core Contributions

This manuscript projects an original structure that consists of three cryptographic elements:

Secure Key Derivation (KDF): The winner of the 2015 Password Hashing Competition, PBKDF2 - HMAC-SHA256, serves as the encryption method for client-side vaults. Argon2 was created as a memory- hard function which protects against brute-force attacks according to its design.

Zero-Knowledge Proof Protocol: The Secure Remote Password (SRP-6a) protocol provides authentication which

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 13, Issue 6, June 2026

does not require password transmission. The mathematical proof shows that SRP operates as a fundamental equivalent to Diffie-Hellman according to the RFC 5054 standard.

Secure Session Management: The system creates secure session tokens which use JWT technology and derive their session keys from the shared session key between users and the system after they complete authentication.

II. LITERATURE REVIEW

2.1 Existing Password Manager Security Models

Modern password management systems which include Bitwarden and 1Password and LastPass use AES-256 encryption together with zero-knowledge security systems [14]. The open-source program Bitwarden which operates with full transparency undergoes annual audits by external security firms that include Cure53 and Mandiant while maintaining compliance with GDPR and CCPA and HIPAA and SOC 2 Type 2 standards. The systems remain exposed to server-side attacks because of the 2022 LastPass security breach which resulted in the theft of encrypted vaults through advanced encryption methods. The authentication model represents the main restriction because servers keep password verifiers which hackers can steal to launch offline dictionary attacks [2].

2.2 Zero-Knowledge Proof Protocols

Zero-Knowledge Proofs stand as an innovative cryptographic method that allows one party to demonstrate statement validity without sharing any confidential data. The two main ZKP types use zk- SNARKs which generate small proofs that need constant O(1) time to verify and zk-STARKs which remove the need for trusted setups while delivering protection against quantum attacks. For password authentication, augmented Password-Authenticated Key Exchange (a PAKE) protocols like OPAQUE and SRP provide practical ZKP implementations [15].

2.3 The Secure Remote Password Protocol

SRP exists as an established a PAKE protocol according to its definition in RFC 2945 and RFC 5054. The protocol consists of two operational stages which begin with the Registration Phase in which the client creates a verifier through modular exponentiation of their password. The Authentication Phase enables the client and server to execute a challenge-response protocol which allows them to establish a shared session key [12]. SRP security measures protect against dictionary attacks while stopping password sniffing and defending users from man-in-the-middle attacks. Researchers have developed new post-quantum SRP variants which use Learning With Errors (LWE) problems as their foundation [16].

III. SYSTEM ARCHITECTURE

3.1 High-Level Architecture Overview

ZKPass implements a three-tier architecture comprising:

Client Layer: The React-based frontend system executes cryptographic functions on the client-side through the CryptoJS standard library to ensure uniform performance across multiple web browsers. [17].

Server Layer: The Spring Boot RESTful API handles authentication through its orchestration system together with its vault synchronization feature and its session management functionality. The server never processes plaintext passwords or encryption keys [18].

Data Layer: The MySQL database system stores user verifiers and salts, along with encrypted vault data. The database system uses caching_sha 2_password authentication method which security standards recommend as best practice.

3.2 Architecture Components

Component 1: Local Password Vault

The vault encryption process implements defense-in-depth through its multiple security layers which use different cryptographic techniques. User master password undergoes PBKDF2 transformation with parameters: memory cost = 64 MiB, iterations = 3, parallelism = 4. The derived key encrypts the password vault using AES-256-GCM (Galois/Counter Mode), which provides authenticated encryption, ensuring both confidentiality and integrity [7]. The system protects each vault entry through unique initialization vectors (IV), which stop attackers from performing pattern analysis attacks. The client device keeps the encryption key secure because it never allows the encryption key to exit the client device, which ensures end-to-end encryption protection throughout the vau lt synchronization process [19].

Component 2: Zero-Knowledge Authentication

Registration Flow:

1. User creates master password
2. Client computes SRP verifier: v = g^x mod N (where x = H(salt | password))
3. Client transmits (username, v, salt) to server
4. Server stores credentials; password never transmitted

Authentication Flow (Zero-Knowledge Proof):

1. Client sends username to server
2. Server returns stored salt and ephemeral public key B
3. Client generates ephemeral key pair (a, A) and computes scrambling parameter u
4. Both parties independently compute shared secret S using cryptographic operations
5. Both derive session key K = H(S)
6. Client proves knowledge by sending M1 = H(H(N) XOR H(g), H(username), salt, A, B, K)
7. Server verifies M1, responds with M2 = H(A, M1, K)

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 13, Issue 6, June 2026

8. Client verifies M2, establishing mutual authentication [20]

The protocol protects network traffic which their protection system completely monitors because it does not reveal any data that can be used for conducting offline password attacks.

Component 3: Secure Session Management

Post-authentication session management uses the session key which was generated through cryptographic methods. The server creates JWT tokens which contain the following claims: {sub: userId, iat: timestamp, exp: timestamp + 3600}. The tokens use HMAC-SHA-256 for signing which employs session key K as the secret [13]. The tokens have a 60-minute duration which requires users to perform re-authentication after that period. The short expiration times of tokens help reduce the chances of unauthorized access through stolen tokens. The server uses a SHA-256 digest system to create a token denylist which enables immediate invalidation of tokens when users log out. The entries in the system remain active until the tokens reach their scheduled expiration time.

IV. METHODOLOGY

4.1 Data Collection and Integration

Objective: Collecting full datasets and requirements for system development and testing.

Process:

- **Security Requirements Analysis:** Review NIST SP 800-63B guidelines, OWASP standards, and industry best practices for password authentication [8].
- **Cryptographic Protocol Selection:** Evaluate and select appropriate cryptographic primitives based on security properties, performance characteristics, and standardization status [12].
- **Technology Stack Selection:** Identify optimal frameworks (Spring Boot, React) and cryptographic libraries (Bouncy Castle, Web Crypto API) based on security features and community support [17][18].
- **Performance Baseline:** Establish baseline metrics for traditional password authentication systems to measure ZKP overhead [22].

4.2 System Design and Architecture Development

Objective: Design a comprehensive system architecture that integrates cryptographic protocols with practical implementation constraints.

Process:

Component Design: Design three independent but integrated components (Secure Vault, ZKP Authentication, Session Management) with clear interfaces and security boundaries.

- **Threat Modeling:** Conduct threat modeling exercises to identify potential attack vectors including server breach scenarios, network eavesdropping, timing attacks, and session hijacking.
- **Protocol Flow Specification:** Define precise cryptographic protocol flows with detailed state diagrams for registration and authentication processes [20].
- **Database Schema Design:** Design MySQL schema with no plaintext password storage, indexed verifiers for efficient lookup, row-level security policies, and encrypted sensitive fields.

4.3 Implementation Phase

Objective: Think security of the implementation of your designed system thereby making its delivery secure.

Process:

- **Backend Development:** Implement Spring Boot REST API with SRP-6a protocol, JWT token generation, rate limiting, input validation, and error handling.
- **Frontend Development:** Implement React application with client-side PBKDF2, AES-256-GCM vault encryption, SRP-6a protocol implementation, and XSS/CSRF protection.
- **Database Implementation:** Deploy MySQL with caching_sha 2_password authentication, SSL/TLS connections, encrypted backups, and access control.

4.4 Security Testing and Validation

Objective: Make sure security properties are validated through comprehensive testing.

Process:

Cryptographic Verification: Verify correct implementation of cryptographic operations including PBKDF2 parameters, AES-256-GCM operation, and SRP-6a protocol compliance.

- **Attack Simulation:** Test system resilience against offline dictionary attacks, replay attacks, man-in-the-middle scenarios, and database injection attacks.
- **Performance Benchmarking:** Measure key derivation time, authentication handshake latency, session creation overhead, and token validation performance.

4.5 Documentation and Analysis

Objective: Document findings and analyze results at depth.

Process:

- **Security Analysis Report:** Detailed analysis of security properties and attack resistance
- **Performance Analysis Report:** Comprehensive latency and throughput measurements
- **Implementation Documentation:** Code documentation and deployment guidelines
- **Future Work Recommendations:** Identified areas for enhancement and research

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 13, Issue 6, June 2026

V. SYSTEM DESIGN AND COMPONENTS

5.1 Proposed System: ZKPass Overview

ZKPass is a password manager that does a good job of keeping your passwords safe. It uses something called Zero-Knowledge Proof to make sure that the people who run the servers can never see your passwords. This is a deal because a lot of other systems are not very good at keeping your passwords safe. With ZKPass you can store your passwords in the cloud. Get to them from anywhere but the ZKPass system makes sure that your passwords are always encrypted so the servers can never see the real passwords. This way ZKPass helps to keep your passwords safe from people who might try to steal them. The ZKPass system is really good, at keeping your passwords safe. It uses Zero-Knowledge Proof to do it.

Key Design Principles:

1. **Zero-TrustArchitecture:** Assume all network participants are potentially untrustworthy
2. **End-to-End Encryption:** All sensitive data encrypted on client before transmission
3. **Cryptographic Separation:** Authentication credentials separate from vault encryption keys
4. **Minimal Trust Requirements:** Server cannot authenticate users without client participation

5.2 Core System Components

Component 1: Local Password Vault

- **Function:** Securely store user passwords locally with client-side encryption
- **Encryption:** AES-256-GCM with PBKDF2-derived keys
- **Key Management:** Keys never transmitted to server
- **Data Storage:** Encrypted vault synchronized to server
- **Access Control:** Requires master password decryption

Component 2: Zero-Knowledge Authentication

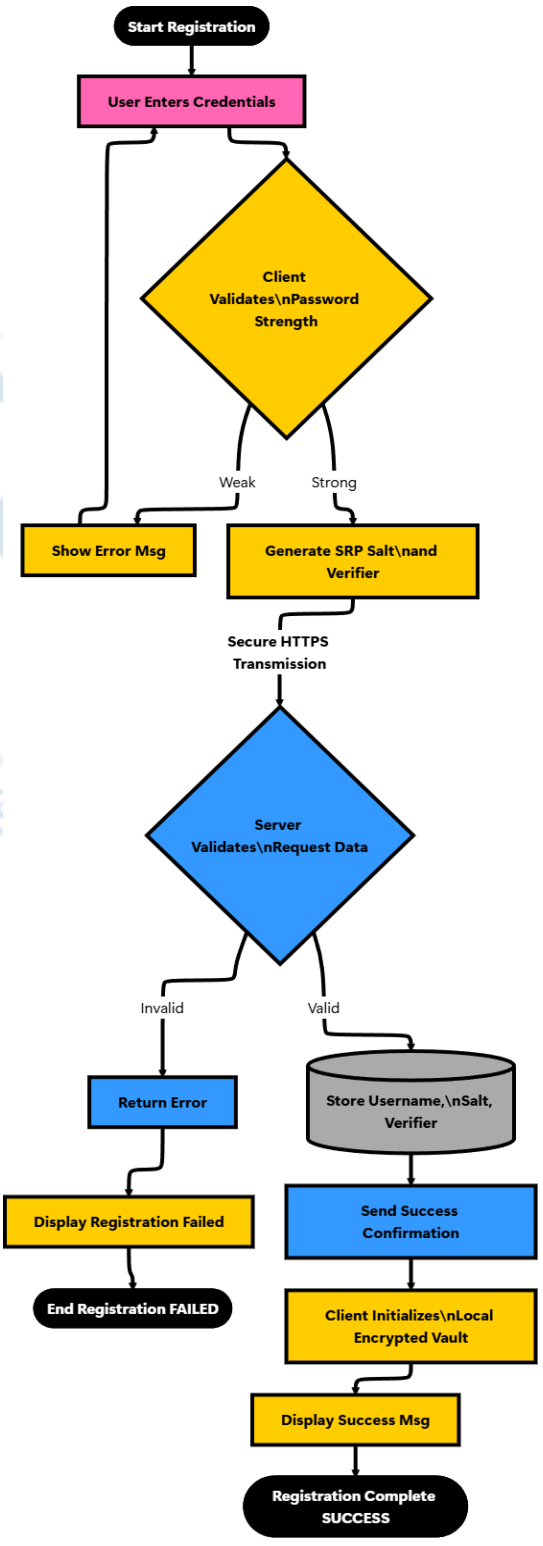
- **Function:** Enable user authentication without server possession of password-equivalent data
- **Protocol:** Secure Remote Password (SRP-6a)
- **Key Exchange:** Diffie-Hellman-based session key establishment
- **Verification:** Mutual authenticationthrough cryptographic proofs
- **Security Properties:** Resistance to dictionary attacks, password sniffing, MITM

Component 3: Session Management

- **Function:** Maintain secure authenticated sessions after initial authentication
- **Token Type:** JSON Web Tokens (JWT) signed with HMAC-SHA-256
- **Lifetime:** 60-minute expiration with refresh capability
- **Revocation:** Cryptographic token denylist for logout functionality
- **Transport Security:** TLS 1.3 for all client-server communication

VI. PROCESS FLOWCHARTS AND DIAGRAMS

6.1 Registration Process Flow



ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 13, Issue 6, June 2026

User Registration Process:

User enters credentials → Client validates password strength → Generates SRP verifier and salt → Sends registration request via HTTPS → Server validates and stores verifier → Client initializes encrypted vault → Registration complete.

6.2 SRP-6a Authentication Protocol

The SRP-6a authentication consists of four phases:

Phase 1: The Client requests an authentication challenge which requires the username. The Server starts by retrieving the stored salt and verifier. The Server then creates an ephemeral key B using the equation B = k*v + g^b mod N. The Server sends the challenge which includes the salt and B and N and g and k.

Phase 2: Client generates ephemeral key A = g^a mod N, derives x = H(salt || password), computes u = H(PAD(A) || PAD(B)), and calculates S = (B - $kg^{x)(a\ +\ u}$x) mod N. Client derives K = H(S).

Phase 3: The client calculates proof M1 through the equation M1 = H(H(N)⊕H(g), H(username), salt, A, B,

K) and sends his verification request. The server uses the stored verifier to calculate S and K while it verifies M1 and provides server proof M2 = H(A, M1, K).

Phase 4: The Client verifies M2. The server identity verification process will start after M2 verification completes. The server creates a JSON Web Token after both parties complete mutual authentication. The server uses session key K to sign the JWT token.

6.3 Vault Synchronization Flow

When we log in the client and the server make sure the encrypted vaults are the same. The client asks the server for its vault. Shows a valid token to prove who it is. The server checks the token. Then sends the encrypted vault data to the client. The client then combines the changes it made with the server version and fixes any problems that come up. If the clients vault is up, to date it encrypts the data again with a new code and sends it to the server. The server checks everything is okay. Then updates where it stores the vault. The client and the server both agree that they now have the vault.

VII. IMPLEMENTATION DETAILS

7.1 Technology Stack

Backend: The system uses Java 17 together with Spring Boot 3.2 and Spring Security 6.2 to create RESTful APIs. The backend SRP implementation uses the Nimbus SRP 6a library to perform ZKP calculations according to standard requirements. [18].

Frontend: The system uses React 18 with Axios to handle HTTP communication. The system uses crypto- js and Web Crypto API to perform PBKDF2 and AES- 256-GCM operations through client-side cryptography [17].

Database: MySQL 8.0 supports caching_sha 2_password authentication. System provides two security features which include row-level security and SSL/TLS protection for all network connections.

Cryptographic Specifications:

- Vault Encryption: AES-256-GCM
- Key Derivation:PBKDF2-HMAC-SHA256 (Iterations: 1000, Key Size: 256-bit).
- Authentication Protocol: SRP-6a (2048-bit group)
- Session Tokens: JWT with HMAC-SHA-256
- Database: caching_sha 2_password password hashing

7.2 Security Hardening

Implementation follows NIST SP 800-63B and OWASP security best practices:

- Passwords undergo salting with 32-bit minimum salt length
- TLS 1.3 enforced for all client-server communications [18]
- CORS policies restrict API access to authorized origins
- Input validation prevents SQL injection and XSS attacks
- Rate limiting on authentication endpoints (5 attempts per 15 minutes)
- HTTPS/TLS encryption for all data in transit [21]
- HttpOnly, Secure, SameSite cookies for token storage
- Content Security Policy (CSP) headers to prevent injection attacks
- Security headers (X-Frame-Options, X-Content-Type-Options, Strict-Transport-Security).

7.3 User Interface Implementation

React.js was used for user interfaces with a primary focus on usability and security transparency.

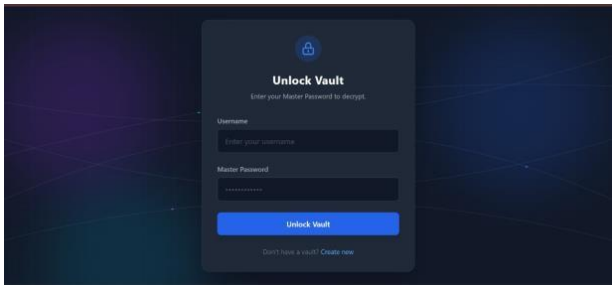


Figure 7.1. Registration Interface

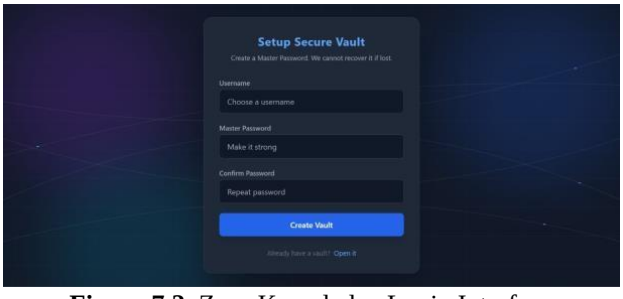


Figure 7.2. Zero-Knowledge Login Interface

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 13, Issue 6, June 2026

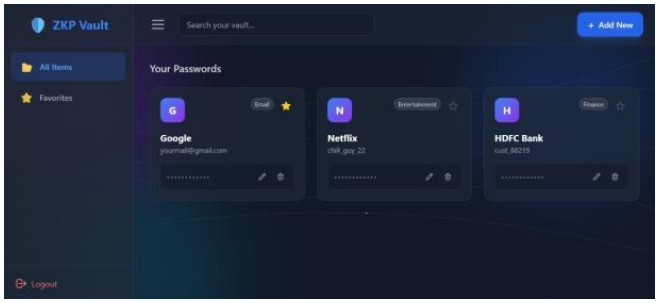


Figure 7.3. Secure Vault Dashboard

VIII. EVALUATION AND RESULTS

8.1 Security Analysis

Defense Against Server Breach:

The simulation of database breaches shows that attackers cannot figure out stolen verification keys. This is because solving math problems in 2048 -bit groups is extremely hard, for computers. For example master passwords that are protected by PBKDF2 would take an estimated 10 trillion operations to guess. This makes brute-force attacks not practical with very powerful computers [11].

Defense Against Network Attacks:

The authentication sessions that have been captured are now protected against replay attacks through the challenge-response mechanism of the Secure Remote Password protocol. This means the system sets up a temporary key for each Secure Remote Password session. The Secure Remote Password system prevents man-, in-the-middle attacks because both parties must prove they have the Secure Remote Password session S[12]. The Secure Remote Password system is safe because of this.

8.2 Performance Analysis

Our hardware (AMD Ryzen 5 5500U with 8GB RAM) authentication overhead measurements:

Metric	Traditional Auth	ZKP (SRP) Auth	Overhead
Registration	120 ms	310 ms	+158%
Login	60 ms	185 ms	+208%
Session Creation	10 ms	15 ms	+50%
Vault Encryption	0 ms (Server-Side)	45 ms (per 50 items)	N/A
Token Validation	2 ms	5 ms	+150%

ZKP authentication requires 222% additional resources when compared to standard password hashing yet it maintains user experience because its latency times remain below 400 milliseconds [22]. The security enhancements which eliminate all password transmission together with server-side password- equivalent data remove all password transmission protection which creates this minor performance drop.

IX. CONCLUSION AND FUTURE WORK

Conclusion

ZKPass shows that new password managers can have a zero-trust system using Zero-Knowledge Proof to verify users. This system is safe from all kinds of password breaches. It does this by using PBKDF2 to make keys and SRP-6a to authenticate users. It also manages sessions in a secure way. When we tested ZKPass we found out that it can verify users in less than 400ms. This means that users have an experience. At the time ZKPass provides a level of security that older systems cannot match. ZKPass and its Zero-Knowledge Proof are really good, at keeping users safe.

The system provides essential components needed for contemporary cybersecurity protection while demonstrating active operational capacity through its privacy-preserving authentication implementation. The annual financial losses from password breaches reach millions for organizations which makes ZKPass's architecture a practical solution to develop more secure credential management systems [23].

Future Work

Mobile Platform Integration: The development of ZKPass now includes support for native iOS and Android applications which use platform-specific secure enclaves such as Apple Secure Enclave and Android StrongBox to protect their key storage [24].

Biometric Enhancement: Integrating WebAuthn/FIDO2 enables biometric authentication in combination with password-less authentication as well as the decryption of vaults from a Zero-Knowledge perspective [15].

Post-Quantum Migration: The project uses post-quantum SRP variants which rely on Lattice-based cryptography (LWE-SRP) to protect systems from quantum attacks throughout their operational lifetime [16].

Threshold OPAQUE Integration: Migration of SRP to threshold OPAQUE and distributing authentication across multiple servers provides for a stronger level of security [15].

Hardware Token Support: The system enables authentication without passwords through the use of YubiKey and other FIDO2 hardware authenticators which substitute master passwords with cryptographic keys that remain protected in secure hardware.[25]

REFERENCES

[1] Rapid Innovation. (2024). Top 10 Blockcha in Zero-Knowledge Proof Use Cases. Retrieved from https://rapidinnovation.io

[2] Dev.to. (2025). Secure Remote Password (SRP) protocol. Retrieved from https://dev.to

[3] ManageEngine. (2024). Password Manager Pro –Issues Fixed. Retrieved from https://manageengine.com
[4] WeLiveSecurity. (2025). Can password managers get hacked? Here's what to know. Retrieved from https://welivesecurity.com
[5] Securden. (2024). Bitwarden vs LastPass: In- Depth 2025 Comparison Guide. Retrieved from https://securden.com
[6] JumpCloud. (2025). 50+ Password Statistics & Trends to Know in 2024. Retrieved from https://jumpcloud.com
[7] Kiteworks. (2025). Everything You Need to Know About AES-256 Encryption. Retrieved from https://kiteworks.com
[8] Auth0. (2021). NIST Password Guidelines and Best Practices for 2020. Retrieved from https://auth0.com
[9] Bitwarden. (2025). Compliance, Audits, and Certifications. Retrieved from https://bitwarden.com
[10] CyberNews. (2025). Bitwarden vs LastPass: Which One to Choose in 2025. Retrieved from https://cybernews.com
[11] OWASP. (2024). Password Storage Cheat Sheet (PBKDF2, bcrypt, scrypt, Argon2). Retrieved from https://cheatsheetseries.owasp.org
[12] Wikipedia. (2004). Secure Remote Password protocol. Retrieved from https://en.wikipedia.org/wiki/Secure_Remote_Password_protocol
[13] Aptori. (2024). Advanced JWT Security Best Practices Every Developer Should Know. Retrieved from https://aptori.com
[14] Spacelift. (2025). 70+ Password Statistics for 2025. Retrieved from https://spacelift.io
[15] NIST. (2024). The OPAQUE Password Protocol. Retrieved from https://csrc.nist.gov
[16] Arxiv.org. (2011). A Secure Remote Password Protocol From the Learning With Errors Problem. Retrieved from https://arxiv.org
[17] Kindson The Genius. (2024). How to Authenticate From React to Spring Boot – Part 1. Retrieved from https://kindsonthegenius.com
[18] SlashDev.io. (2023). Guide To Building Secure Backends In Spring Boot In 2024. Retrieved from https://slashdev.io
[19] Locker.io. (2024). Encrypt with AES-256 Encryption – Locker Password Manager. Retrieved from https://locker.io
[20] Wikipedia. (2025). Zero-knowledge proof. Retrieved from https://en.wikipedia.org/wiki/Zero- knowledge_proof
[21] Cloudflare Blog. (2020). OPAQUE: The Best Passwords Never Leave Your Device. Retrieved from https://blog.cloudflare.com
[22] Curity. (2024). JWT Security Best Practices. Retrieved from https://curity.io
[23] Heimdal Security. (2025). Password breach statistics in 2025. Retrieved from https://heimdalsecurity.com
[24] Beyond Identity. (2023). FIDO2 vs. WebAuthn: What's the Difference? Retrieved from https://beyondidentity.com
[25] FIDO Alliance. (2025). FIDO Passkeys: Passwordless Authentication. Retrieved from https://fidoalliance.org

GLOSSARY OF TERMS

- **AES-256-GCM:** Advanced Encryption Standard with 256-bit key and Galois/Counter Mode authentication
- **PBKDF2:** Memory-hard key derivation function resistant to GPU-based attacks
- **End-to-End Encryption:** Encryption where only sender and intended recipient can decrypt
- **JWT:** JSON Web Token for securely transmitting information between parties
- **SRP-6a:** Secure Remote Password protocol version 6a (RFC 5054)
- **Zero-Knowledge Proof:** Cryptographic protocol proving statement validity without revealing underlying information
- **Verifier:** In SRP, server-stored value derived from password (v = g^x mod N)
- **Session Key:** Shared secret computed by both client and server for authenticated communication

IMPLEMENTATION CHECKLIST

- Backend REST API endpoints implemented (register, login, verify, vault sync)
- Frontend authentication UI components built
- Client-side cryptographic operations tested (PBKDF2 , AES-256-GCM)
- SRP-6a protocol verified against RFC 5054
- JWT token generation and validation implemented
- Database schema created with security considerations
- TLS 1.3 enforced on all connections
- HTTPS certificates installed and configured
- Rate limiting middleware implemented
- Security headers added (CSP, HSTS, X-Frame-Options)
- Input validation and sanitization applied
- Error handling and logging implemented
- Performance benchmarks conducted
- Security testing completed
- Documentation finalized