IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

# Design and Performance Evaluation of a Post-Quantum Password Manager Using Argon2id, AES-256-GCM, and ML-KEM

**Sanjana Pawar[1], Rama Pandey[2], Mrs. Soly Zacharish[3]**

IT/CS Department, Pillai College of Arts, Commerce & Science Navi Mumbai, India[1-3]

**Abstract:** Quantum computing. It also requires that we think about post-quantum methods to keep sensitive credentials safe for a long time. This paper presents a post-quantum password manager that uses Argon2id to derive keys, from passwords AES-256-GCM to encrypt the vault with authentication and ML-KEM to set up post-quantum keys. The versions ML-KEM-512, ML-KEM-768 and ML-KEM-1024 were. Tested under identical conditions using twenty benchmarks. We examined generation, encapsulation, decapsulation, the size of cryptographic objects and how the password vault works. The computational cost and storage space grow as the ML-KEM parameter set becomes larger. ML-KEM-768 turned out to balance security, performance and storage so it was chosen for the password-manager prototype.

**Keywords:** Post-Quantum Cryptography, Password Manager, Argon2id, AES-256-GCM, Password Security, Cryptographic Performance

## 1. INTRODUCTION

1. Password managers keep user passwords. Login details safe and secure not in plain text. They use encryption with methods. As quantum computers get better there is now a need to check encryption methods that can handle attacks from computers.

2. Post-quantum cryptography has algorithms that can stand up to attacks from computers. In 2024 NIST announced the version of FIPS 203. This document describes ML-KEM as a -quantum key-encapsulation method and lists three options (each based on the NIST estimate of symmetric security strength with options for ML-KEM-512, ML-KEM-768 and ML-KEM-1024). (NIST Computer Security Resource Center)

3. This article talks about a password manager that uses the Argon2id password-based derivation function the symmetric authenticated encryption with associated data (AEAD) algorithm AES-256-GCM and the post-quantum key establishment method called Module-Lattice-based Key-Encapsulation mechanism (ML-KEM). All three ML-KEM options are studied with the setup for speed, size and password vault features.

4. The goal of this experiment is to understand the balance between security, performance and storage, for each of the three ML-KEM options and to pick one ML-KEM configuration to use in the password manager.

## 2. LITERATURE REVIEW

Previously research already focused on post-quantum password managers and password-authentication. Aremu et al. [5] introduced a post-quantum password manager, and Ren et al. [4] and Li et al. [6] have examined post-quantum password-authentication solutions. NIST FIPS 203 [1] has standardized the ML-KEM-512, ML-KEM-768, and ML-KEM-1024; the password-based key derivation functions Argon2id [2] and AES-GCM [3] have been standardized for authentication and encryption. While they have different architecture and experiments conditions, this motivates a controlled evaluation in a common password- manager architecture.

## 3. RESEARCH GAP

Earlier studies on post-quantum password managers, password-authentication protocols and secure vault systems show designs and ways to mix cryptographic methods. They also show how to run these designs in test environments. Because of this it is very hard to compare them. In particular there is a small amount of evidence and there is no data on how the three standard ML-KEM parameters—ML-KEM-512, ML-KEM-768 and ML-KEM-1024—perform, in terms of speed and storage when used in the same password manager architecture. This study fills that gap.

 This work is licensed under a Creative Commons Attribution 4.0 International License 

## 4. RESEARCH OBJECTIVE

1. Objective to develop and implement a post-quantum password manager based on Argon2id, AES-256-GCM, and ML-KEM.
2. To compare ML-KEM-512, ML-KEM-768, and ML-KEM-1024 under the identical experimental setup.
3. For measuring the key-generation, encapsulation, and decapsulation performance for each set of parameters.
4. To compare the public-key and the size of the ciphertext for all three parameter sets.
5. To explore the security-performance-storage trade-offs and to select an appropriate ML-KEM configuration for our password manager.

## 5. THREAT MODEL

I look at three threats to the password manager. An attacker might download the vault.json file. Try to read the stored passwords. The passwords are encrypted so they are not in text. The attacker might also try to corrupt the encrypted vault. The AES-256-GCM encryption will detect that. Finally an attacker might try to read the vault with a master password and the application should block that.
This study does not cover attacks such, as malware, memory extraction, side-channel attacks or compromised devices.

## 6. PROPOSED ARCHITECTURE

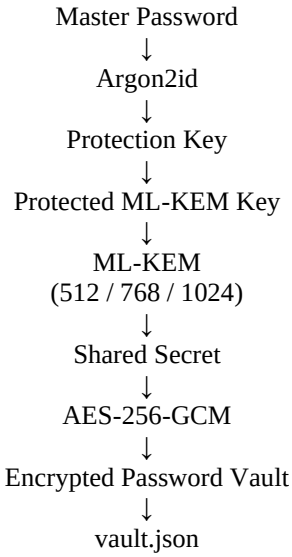



Figure 1. Proposed architecture of the post-quantum password manager.

The proposed password manager uses Argon2id, AES-256-GCM and ML-KEM as cryptographic components. The Argon2id component derives material, from the master password. The AES-256-GCM component is used for authenticated encryption of the password vault. The ML-KEM component handles post-quantum establishment. The password manager considers ML-KEM-512, ML-KEM-768. Ml-kem-1024 for comparison. The encrypted vault is saved in vault.json.

## 7. IMPLEMENTATION AND EXPERIMENTAL METHOD

I designed this password manager as a Python program with Argon2id, AES-256-GCM, and ML-KEM. Its features include vault creation, storing, and pulling passwords, locking, and unlocking vaults, saving encrypted vaults, and detecting vault tampering.

The tests were conducted with the ML-KEM-512, ML-KEM-768, and ML-KEM-1024 under identical conditions. All the configurations were used for 20 iterations for each of key generation, encapsulation, and decapsulation. The public key, cipher text and secret text size was measured as well.

Testing of the integrated password-manager setup was performed with ML-KEM-768 with Argon2id, and AES-256-GCM. Functional tests: creation opening closing storing loading a vault, loading a corrupted vault, opening with the wrong password, opening with the correct password, tampering with the vault.

The password manager was developed and tested in a Python environment with the cryptographic libraries needed for Argon2id, AES-256-GCM, and ML-KEM. The ML-KEM-512, ML-KEM-768, and ML-KEM-1024 parameter sets were also tested using the same implementation and test settings. Each parameter set was tested 20 times for key generation, encapsulation, and decapsulation, and the average execution time was recorded. Sizes of cryptographic objects and successful shared-secret verification were also recorded.

| Metric | Evaluation |
|---|---|
| ML-KEM variants | 512, 768, 1024 |
| Benchmark runs | 20 per variant |
| KEM operations | KeyGen, Encapsulation, Decapsulation |
| Size metrics | Public key, Ciphertext, Shared secret |
| Encryption | AES-256-GCM |
| Key derivation | Argon2id |
| Functional tests | Vault, unlock, retrieval, persistence, tamper detection |

## 8. RESULTS AND ANALYSIS

### 8.1 ML-KEM Performance

The evaluation of computational performance of ML-KEM-512, ML-KEM-768, and ML-KEM-1024 was conducted on key-generation, encapsulation, and decapsulation over 20 benchmark runs.

| ML-KEM Parameter Set | Key Generation (ms) | Encapsulation (ms) | Decapsulation (ms) |
|---|---|---|---|
| ML-KEM-512 | 0.034865 | 0.037985 | 0.040720 |
| ML-KEM-768 | 0.050340 | 0.052255 | 0.060840 |
| ML-KEM-1024 | 0.073030 | 0.077420 | 0.085565 |

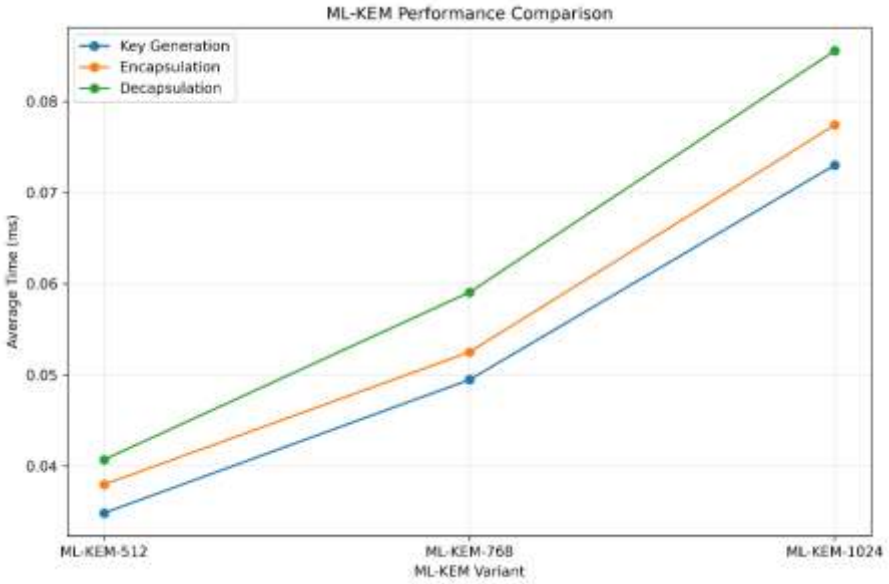



Figure 2. Performance comparison of ML-KEM-512, ML-KEM-768, and ML-KEM-1024 over 20 benchmark runs.

The results indicate that ML-KEM-512 has the lowest execution time for the three operations and ML-KEM-1024 has the highest execution time. ML-KEM-768 has a performance that is between ML-KEM-512 and ML-KEM-1024.

### 8.2 Cryptographic Size Comparison

The cryptographic object sizes of ML-KEM-512, ML-KEM-768, and ML-KEM-1024 were presented and used to compare public-key size, ciphertext size and shared-secret size.

| ML-KEM Parameter Set | Public Key Size | Ciphertext Size | Shared Secret Size |
|---|---|---|---|
| ML-KEM-512 | 800 bytes | 768 bytes | 32 bytes |
| ML-KEM-768 | 1184 bytes | 1088 bytes | 32 bytes |
| ML-KEM-1024 | 1568 bytes | 1568 bytes | 32 bytes |

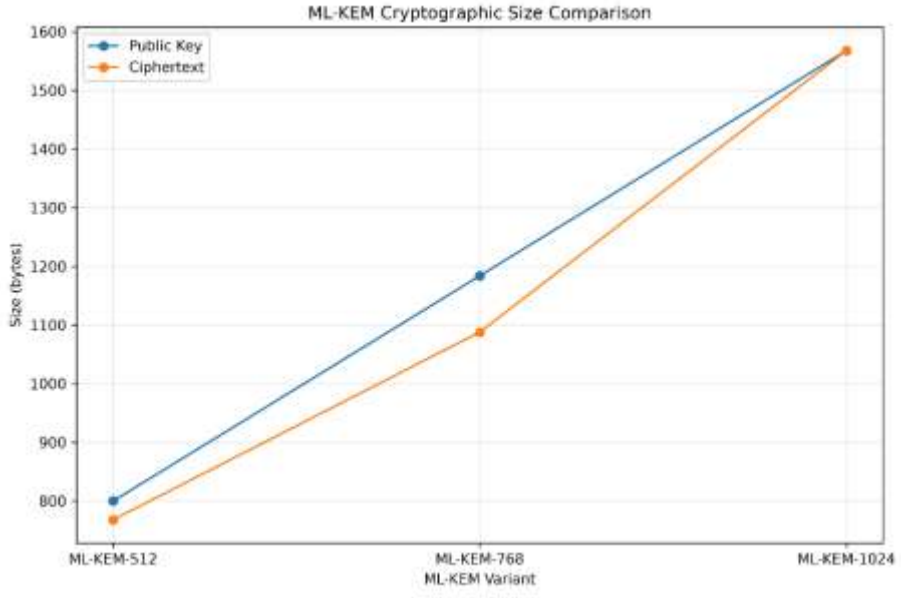



Figure 3. Public-key and ciphertext size comparison of the three ML-KEM parameter sets.

The results indicate that both public-key and ciphertext sizes grow as the ML-KEM parameter set increases. ML-KEM-512 features the smallest cryptographic objects, while ML-KEM-1024 has the largest. The shared-secret size stays the same at 32 bytes for all three parameter sets.

### 8.3 Password Manager Functional Results

Functional testing of the password manager is described below. The password manager was subjected to various vault and security tests, like creation of a vault, generation of the ML-KEM-768 key, encryption of data using the AES-256-GCM encryption algorithm, storing, and retrieving passwords, acceptance, and rejection of the master-password, vault duration, vault durability, detection of suspicious vault state, and rescue. Each functional test was successful.

| Functional Test | Result |
|---|---|
| Vault creation | Passed |
| Password storage and retrieval | Passed |
| Correct master password unlock | Passed |
| Incorrect master password rejection | Passed |
| Vault persistence after locking and reopening | Passed |
| Tamper detection | Passed |

The functional tests confirmed that the proposed password manager successfully performed the main vault operations. The system created and encrypted the vault, stored, and retrieved passwords, and successfully locked and reopened the vault. An incorrect master password was rejected, while modification of encrypted vault data was detected. Plaintext

 This work is licensed under a Creative Commons Attribution 4.0 International License 

password visibility was not observed in the stored `vault. json` file. These results demonstrate successful functional operation of the implemented prototype under the tested conditions.

## 9. DISCUSSION

The experimental results show that there is an evident trade-off between the three ML-KEM parameter sets. Among the three tested ML-KEM, ML-KEM-512 had the best performance on lowest computational time and smallest cryptographic object sizes. As for the worst performance, ML-KEM-1024 had the highest computational and storage overhead. ML-KEM-768 exhibits as a middle ground in all metrics under study.

The results show that given the ML-KEM parameters, as the parameter set increased, the cryptographic object size increased plus the amount of computation required. This is a trade-off, which would need to be considered when selecting a parameter set for real-world password-management usage.

Based on the results of our experiments, we selected ML-KEM-768 to build our integrated prototype of the password-manager. It provided a balanced compromise between performance and storage, and it was easily attachable to both Argon2id and AES-256-GCM. Our functional tests were successfully run including vault creation encryption password management, persistence, and tampering detection.

## 10. LIMITATIONS

This study has several limitations. First, the evaluation was performed on a Python-based prototype and under a specific experimental environment; therefore, the measured execution times may vary with different hardware, operating systems, and cryptographic libraries. Second, the experimental comparison was limited to the three standardized ML-KEM parameter sets: ML-KEM-512, ML-KEM-768, and ML-KEM-1024, and did not include other post-quantum KEMs or hybrid key-establishment approaches. Third, the performance evaluation used 20 benchmark runs for each parameter set, which provides a controlled comparison but does not represent large-scale deployment conditions. Finally, the security evaluation focused on functional checks such as incorrect-password rejection, vault persistence, plaintext visibility, and tamper detection. These tests demonstrate the functionality of the implemented prototype but do not constitute a formal cryptographic security proof or a comprehensive assessment against all possible attacks.

## 11. FUTURE WORK

Future research could broaden this study by assessing the proposed password manager on various hardware platforms and with different cryptographic implementations to improve the generalizability of performance findings. The system may also be expanded to test additional post-quantum KEMs and hybrid classicalpost-quantum key-establishment methods. Further work should involve larger-scale benchmarking, formal security analysis, side-channel evaluation, and interoperability testing. The prototype could be further developed to include secure synchronization, multi-device support, and cryptographic agility to accommodate future updates in post-quantum cryptographic standards and algorithms.

## 12. CONCLUSION

This study designed and evaluated a post-quantum password manager using Argon2id, AES-256-GCM, and ML-KEM. The results showed increasing computational and storage overhead from ML-KEM-512 to ML-KEM-1024, while ML-KEM-768 provided an intermediate profile. ML-KEM-768 was because of this selected for the integrated prototype, which successfully passed the functional and cryptographic tests. The study demonstrates the practical trade-offs involved in selecting an ML-KEM parameter set for post-quantum password management.

## REFERENCES

[1]. National Institute of Standards and Technology (NIST), “Module-Lattice-Based Key-Encapsulation Mechanism Standard,” FIPS 203, Aug. 2024.https://doi.org/10.6028/NIST.FIPS.203

[2]. A. Biryukov, D. Dinu, D. Khovratovich, and S. Josefsson, “Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications,” RFC 9106, Internet Engineering Task Force, Sept. 2021.https://doi.org/10.17487/RFC9106

[3]. M. J. Dworkin, “Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC,” NIST Special Publication 800-38D, Nov. 2007.https://doi.org/10.6028/NIST.SP.800-38D

[4]. P. Ren, X. Gu, and Z. Wang, “Efficient module learning with errors-based post-quantum password-authenticated key exchange,” IET Information Security, vol. 17, no. 1, pp. 3–17, 2023.https://doi.org/10.1049/ise2.12094

[5]. O. R. Aremu, S. Tonyalı, and A. Köse, “A Password Manager for Post-Quantum Era,” in Proceedings of the International Conference on Information Security and Cryptology (ICONSEC), 2022.

[6]. Z. Li, D. Wang, and E. Morais, "Quantum-Safe Round-Optimal Password Authentication for Mobile Devices," IEEE Transactions on Dependable and Secure Computing, 2021.