Information Security

Course Project

Department of Artificial Intelligence

National University of Computer and Emerging Sciences
Islamabad, Pakistan

Submitted by:

21I-2707 Shanzay Alam

21I-0345 Amna Khan

Post-Quantum Cryptography Web Application

I. Introduction

The rise of quantum computing poses a significant threat to traditional cryptographic systems. Post-Quantum Cryptography (PQC) aims to create cryptographic algorithms secure against quantum attacks.

In this project, we developed a Flask-based Web Application that demonstrates the use of Post-Quantum Cryptography using the Kyber512 algorithm (a NIST finalist) to securely perform:

- Key Generation
- Message Encryption
- Message Decryption

II. Tools and Technologies Used

Category	Tools
Programming	Python 3.x
Backend Framework	Flask
Cryptography	pqcrypto (Kyber512)
Frontend	HTML, CSS, Bootstrap
Security	Flask-Talisman, Flask-WTF
Version Control	Git + GitHub

III. Description of PQC Algorithm Used

Kyber512 is a key encapsulation mechanism (KEM) that is part of the NIST Post-Quantum Cryptography Standardization process.
Key Features:

i. Small key sizes.
ii. High efficiency in encryption and decryption operations.
iii. Strong theoretical security guarantees.

In this application, Kyber512 is used for:

i. Key Generation: Generating a pair of public and private keys.
ii. Encapsulation: Encrypting data to generate a shared secret.
iii. Decapsulation: Decrypting ciphertext to retrieve the shared secret.

IV. Application Architecture

The project follows a modular structure:

- init.py: Initializes the Flask app and applies security headers.
- crypto_utils.py: Contains functions for key generation, encryption, and decryption using Kyber512.
- routes.py: Handles user requests and orchestrates the cryptographic operations.
- Frontend: Simple HTML templates styled with Bootstrap for better UI.

V. Working Explanation

I. Key Generation

User clicks Generate Keys, and the application uses Kyber512.keygen() to create a public and private key. Keys are encoded in base64 format for safe display and transmission.

II. **Encryption**

User enters a message and provides a public key, and application performs key encapsulation using Kyber512.encaps(public key). The ciphertext and shared secret are encoded and displayed.

III. **Decryption**

User provides ciphertext and private key and application performs decapsulation using Kyber512.decaps(private key, ciphertext). The decrypted shared secret is displayed securely.

VI. Application Screenshots

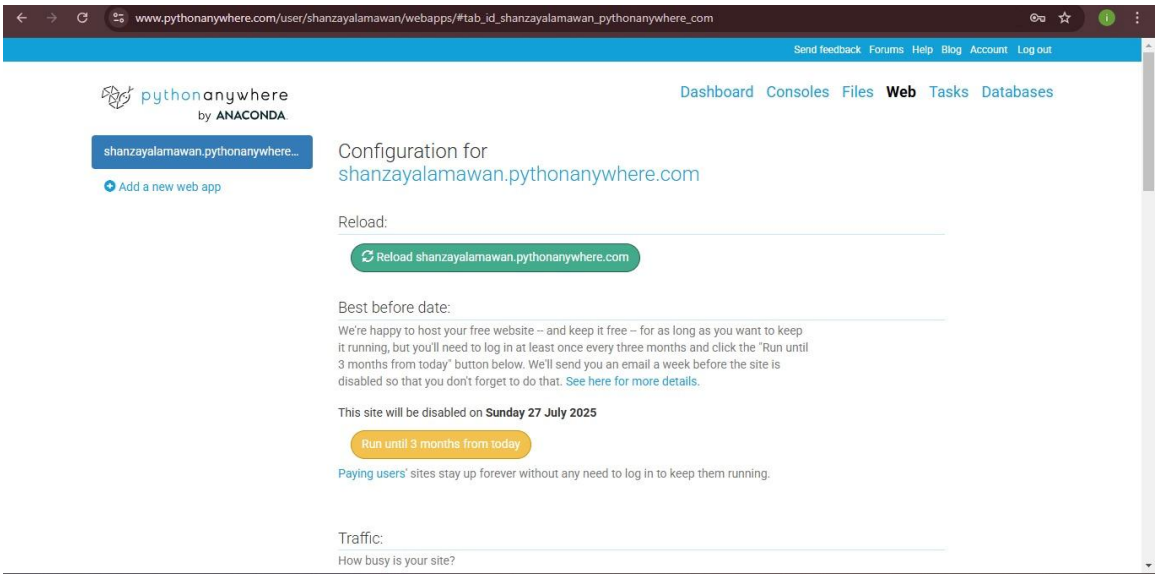

Figure 1: App Webpage Screenshot

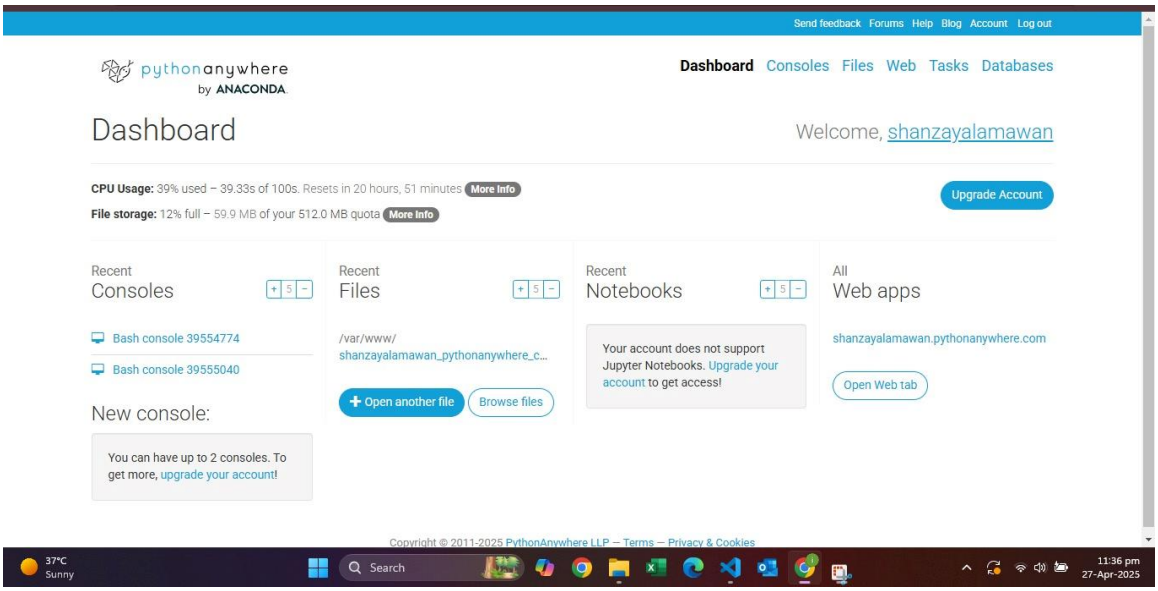

Figure 2: App Dashboard Screenshot

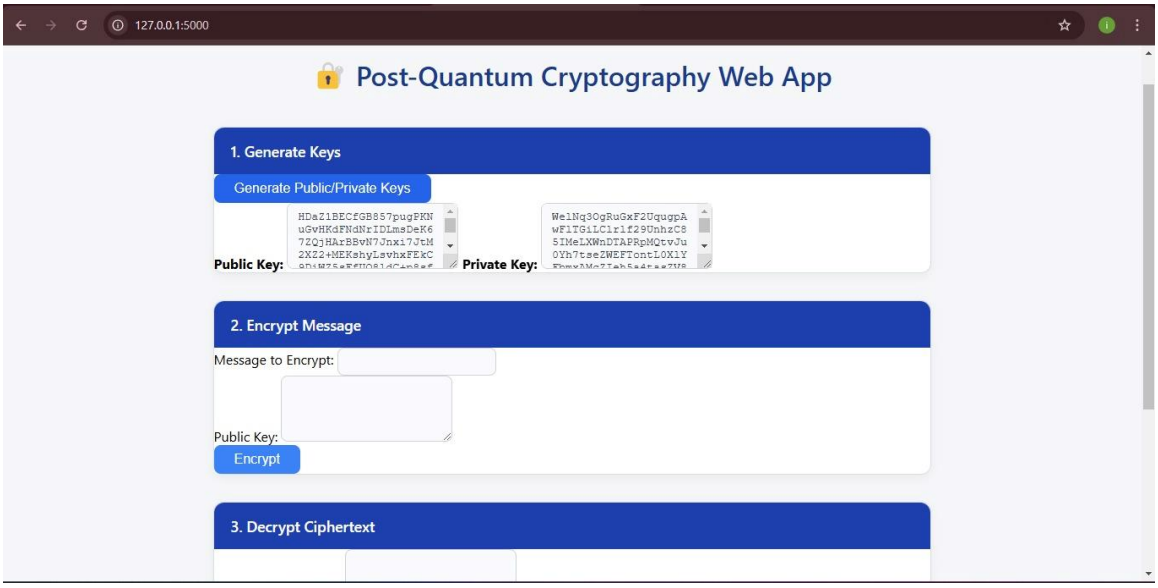

Figure 3: App Homepage Screenshot

VII. Conclusion

The project demonstrates the use of Post-Quantum Cryptography (Kyber512) inside a secure Flask web application. Through key generation, encryption, and decryption, we can understand the basic working of PQC algorithms that are resilient even in the presence of quantum computers.

VIII. References

i. NIST PQC Project: https://csrc.nist.gov/projects/post-quantum-cryptography
ii. GitHub Repository: https://github.com/shanzayalam/PQC_webApp.git