Title: Post-Quantum Cryptography Flask Web Application

Course: Information Security
Student Name: Nibtahil Nafees, Muhammad Faizan Karamat
RollNum: 21I-0330, 21I-0269

GitHub Repository link:

https://github.com/NibtahilNafees/pqc-encryption-flask-app

1. Introduction

The rise of quantum computing threatens the foundation of classical cryptographic systems, prompting research and experimentation into post-quantum cryptography (PQC). This project aims to demonstrate a web-based simulation of a PQC-style encryption-decryption system using Python and Flask. Due to the installation and compatibility limitations of Kyber and liboqs on local machines, the final application uses the `cryptography` library's `Fernet` module to simulate a secure symmetric encryption workflow.

2. Objectives

- Develop a simple encryption/decryption tool accessible via a browser.
- Simulate public/private key generation.
- Provide a hands-on understanding of encryption APIs.
- Explore limitations of integrating true PQC libraries like Kyber/liboqs.

3. Technologies Used

Component	Technology

Backend	Python 3, Flask
Cryptography	`cryptography.Fernet`
Frontend	HTML5, CSS, JavaScript
Version Control	Git & GitHub

4. Key Functionalities

1. Key Generation

A symmetric key is generated using `Fernet.generate_key()`. For the sake of demonstration, the same key is used as both the "public" and "private" key.

2. Encryption

User input is encrypted via `Fernet.encrypt()`, taking the message and the key. The result is a secure ciphertext in base64 format.

3. Decryption

The ciphertext is decrypted using `Fernet.decrypt()` with the same key, returning the original plaintext message.

5. System Architecture

Frontend (HTML form)
↓
Flask Backend (app.py)
↓
Crypto Functions (pqc_crypto.py)
↓

JSON Response (encrypted/decrypted output)

Project Structure:

```
pqc-flask-app/
├── app.py                  # Main Flask app
├── crypto/
│   └── pqc_crypto.py       # Fernet encryption functions
├── templates/
│   └── index.html          # Frontend interface
├── static/                 # CSS/JS (if any)
├── requirements.txt        # Dependencies
└── .gitignore              # Excludes venv and other unnecessary files
```

6. Challenges & Solutions

- **Challenge:** Installation errors with `pqcrypto` and `liboqs`.
    - **Solution:** Switched to `cryptography.Fernet` for ease of use while still retaining encryption principles.
- **Challenge:** Virtual environment files were being committed to GitHub.
    - **Solution:** Added a `.gitignore` to exclude `venv` and other system-specific files.

7. Future Improvements

- Integrate true PQC libraries like Kyber with Docker or Linux-based systems.
- Add support for asymmetric encryption and secure key exchange.

- Improve the frontend with better styling and form validations.
- Extend to encrypt/decrypt file uploads.

8. Conclusion

This project serves as a foundational implementation of cryptographic principles in a web app. Although it uses a symmetric algorithm rather than a true post-quantum system, it effectively simulates key management, encryption, and decryption processes in a user-friendly interface. It provides the groundwork for integrating advanced post-quantum cryptography tools in future work.