CS-3002 Information Security

BSAI – Spring 2025

Semester Project AI - K

Deadline: Wednesday, April 30, 2025, 11:59 pm

Assignment Instructions

- This is a **Group Project** with a maximum of **two members**.
- Only **one member** must submit the **Public GitHub repository link** in the **PDF report** on **Google Classroom** before the specified deadline.
- Each member **must** have visible commits in the GitHub repository to demonstrate their contribution.
- The repository structure must follow **best coding practices**, adhering to a **conventional hierarchical system**.
- Any **commits made after the deadline** will be considered **cheating**, and both members will receive a **zero** for the assignment.
- Submissions **will only be accepted via Google Classroom**. Any other mode of submission will not be entertained.
- It is the **responsibility of each student** to ensure a **timely and correct submission**.
- **No extensions will be granted**, so students are strongly advised to **start early** to avoid last-minute issues.

• Plagiarism is strictly prohibited. If any part of your work is found to be plagiarized, you will receive zero marks for the entire assignment category.

Project Specification: Post-Quantum Cryptography Web Application

Overview

The goal of this project is to develop a secure web application using **Flask** that demonstrates **Post-Quantum Cryptography (PQC)** techniques for encrypting and decrypting data. Students will integrate a PQC algorithm (e.g., Kyber, Dilithium, or any NIST finalist) and implement a user interface to showcase key generation, encryption, and decryption functionality.

Objectives

- Understand the need for PQC in a post-quantum world.
- Implement basic PQC operations in a Flask web app.
- Design a front-end interface for user interaction.
- Provide proper feedback and error handling in the application.

Tools and Technologies

- **Programming Language**: Python 3.x
- **Framework**: Flask
- **Cryptography**: Use `pqcrypto` or similar libraries
- **Frontend**: HTML, CSS, JavaScript (Bootstrap optional)
- **Security**: Flask-Talisman, Flask-WTF (optional)
- **Version Control**: Git + GitHub

Functional Requirements

1. **Homepage** with a simple UI (Flask + HTML):
    - Key generation button
    - Input fields for message encryption/decryption
    - Display generated keys and encrypted/decrypted data
2. **PQC Integration**:
    - Use one post-quantum encryption scheme (e.g., Kyber512 or similar)
    - Functionality for:
        - Public/private key generation
        - Message encryption (using public key)
        - Message decryption (using private key)

3. **Frontend Features**:
    - Clear, interactive layout (HTML/CSS/Bootstrap)
    - Use AJAX or JavaScript for smooth form submissions (optional)
    - Display success/error messages
4. **Security Best Practices**:
    - Use Flask-Talisman to set secure HTTP headers
    - Avoid displaying stack traces or raw error messages
    - Sanitize user input if needed
5. **Project Structure**:
    - Organized Flask app (`/static`, `/templates`, `/routes`, etc.)
    - `.env` file for configs
    - `requirements.txt` for dependencies
    - README.md with setup instructions

Implementation Steps

1. **Setup environment**
    - Create virtualenv
    - Install dependencies (Flask, pqcrypto, etc.)
2. **Basic Flask setup**
    - Set up app routes, templates, and static files
3. **Implement cryptography logic**
    - Add Python functions for keygen, encryption, decryption
4. **Frontend integration**
    - Create HTML forms + result pages
5. **Test functionality + deploy demo (optional)**
    - Use PythonAnywhere or Render for hosting

Deliverables

1. **GitHub Repository**:
    - Codebase with organized folder structure
    - Includes `README.md`, `.env.example`, and `requirements.txt`
2. **PDF Report**:
    - Group members + GitHub repo link
    - Description of PQC algorithm used
    - App screenshots
    - Working explanation (key generation, encryption, decryption)
3. **Live Demo**
    - Deploy the final app using Heroku, Render, or PythonAnywhere for demo purposes.

Recommended Libraries

- Flask
- pqcrypto / pycrystals / third-party PQC lib
- Flask-Talisman
- Flask-WTF (for CSRF)
- Python-dotenv