Final Project (Group Project)

Applied Cryptography - CSAC 329 Cryptographic Application

Project Description:
The Applied Cryptography Application project aims to develop a simple application that implements various cryptographic techniques to **secure communication, data, and information exchange.** Cryptography is the science of encoding and decoding messages to protect their **confidentiality, integrity, and authenticity.** The application will provide a user-friendly interface that allows users to **encrypt, decrypt, and hash** messages/files using different cryptographic algorithms.

Core Requirements:

- **Algorithm Implementation (Minimum):**
    - **3 Symmetric Algorithms:** Must support both Text and File encryption/decryption.
    - **2 Asymmetric Algorithms:** Must support Text encryption/decryption.
    - **4 Hashing Functions:** Must support hashing both Text and Files.
- **User Interface:**
    - Must be a UI-based application (e.g., using Flask, Streamlit, PyQT5/6, tkinter, etc.).
- **Algorithm Information:**
    - Each cryptographic algorithm implemented should include accessible information/descriptions within the application and documentation (e.g., brief history, pseudocode overview, process description, use cases).
- **Cryptographic Libraries:**
    - Utilize standard Python cryptographic modules/libraries such as:
        - pyca/cryptography
        - PyCryptodome
        - Cryptocode
        - RSA
        - hashlib
        - pyaes
        - *(Other relevant libraries are also acceptable)*
- Useful Links (Reference):
    - Crypto with python
    - Cryptographic Services — Python documentation
    - Python Cryptography Toolkit (pycryptodome)
- Project App Example: Click here or Flask App Click here
- Deployment: The application can be developed as a local application or published (e.g., hosted web app).

Project Documentation (README.md):

Your project repository must contain a README.md file in the root directory. This file serves as your project documentation and must include the following sections (use appropriate Markdown formatting):

- **Title Page Information:** (Project Title, Course Name, Date)
- **Group Members:** (List all members)
- **Introduction:** (Brief overview of the project, its purpose, and the importance of cryptography)
- **Project Objectives:** (List at least 3 specific objectives your project aims to achieve)
- **Discussions:**
    - Describe the overall application architecture and UI choice.
    - For each implemented cryptographic algorithm:
        - Name and type (Symmetric/Asymmetric/Hash)
        - Brief history/background

        - Description of the process/how it works (can be high-level or include simplified pseudocode)
        - Libraries used for its implementation
        - How it's integrated into your application
- **Sample Runs/Outputs:**
    - Include screen snippets (screenshots) or text-based output examples for *each* algorithm's functionality (encryption, decryption, hashing for both text and files where applicable). Embed images directly in the README.md or link to them within the repository. Use Markdown code blocks for text output.
    -

Submissions:

- **LEONS (or specified submission platform):**
    - Submit the URL link to your group's project repository (e.g., GitHub, GitLab, Bitbucket).
    - Ensure the repository is accessible to the instructor/TAs.
    - **Repository Contents:**
        - README.md (formatted as described above)
        - All source code files (*.py)
        - Any necessary supporting files (e.g., requirements.txt, UI files *.ui, sample data, images used in README) *.*
        - The repository should be well-organized.
- **Group Oral Presentation:** (Details to be provided separately)

Repository Contribution Guidelines:

- Each group member must use their own GitHub account for contributions
- All work must be committed through individual member accounts (no shared accounts)
- Commit messages should be clear and descriptive of the changes made
- **Important Notes on Repository Contributions:**
    - Individual contributions will be evaluated through:
        - Commit history
        - Number of meaningful commits
        - Code authorship
        - Pull requests (if using branch workflow)
    - Each member should have a reasonable distribution of commits throughout the project timeline
    - Avoid bulk commits at the end of the project
    - Use meaningful commit messages that clearly describe the changes made
    - Consider using branches for feature development
- **Best Practices:**
    - Regular commits (don't wait to commit all changes at once)
    - Use descriptive branch names if implementing feature branches
    - Review and comment on team members' pull requests
    - Document major decisions and changes in commit messages