1. Project Overview

Small groups (3-4 students), to design, implement, and analyze a secure client-server communication system. The project is divided into three major phases, each aligning with the course's progression. The goal is not just to build a "working" system, but to understand the "why" and "how" behind each security decision and to be able to analyze its strengths and weaknesses.

2. Learning Objectives

Upon successful completion of this project, students will be able to:

1- Apply core cryptographic primitives (symmetric encryption, hashing, public-key cryptography) to a practical problem.

2- Design a secure protocol that ensures confidentiality, integrity, and authentication.

3- Implement a security protocol in Python.

4- Analyze the security properties of their own implementation and those of others.

5- Critique and justify design choices based on established security principles.

3. Project Phases & Deliverables

The project is structured to follow the course timeline.

Phase 1: Design and Protocol Specification

1. System Architecture: Diagram of the client-server model.

2. Protocol Specification: A step-by-step description of how a client and server establish a secure session and exchange messages. This must include:

    * Handshake Protocol: How will the client and server mutually authenticate? (e.g., using digital certificates or pre-shared keys). How is a session key established?

    * Record Protocol: How is application data encrypted and protected? You must specify the cryptographic algorithms you plan to use (e.g., AES-256-GCM for authenticated encryption, SHA-384 for hashing, RSA-2048 or ECDSA for key exchange/signatures).

3. Justification of Choices: Why did you choose these specific algorithms and protocols? Justify your choices based on security strength and performance trade-offs discussed in class.

4. Threat Model: List the potential threats your system is designed to mitigate (e.g., eavesdropping, message tampering, man-in-the-middle attacks, replay attacks).

Phase 2: Implementation

1. Source Code: Well-commented code for both a client and a server.

2. Core Requirements:

   * The server must authenticate the client.

   * All communication after the handshake must be confidential and integrity-protected.

   * The system must be resilient to replay attacks.

3. Implementation Report (1-2 pages): Describe any challenges faced, deviations from the original design document, and the libraries used (e.g., `cryptography` in Python).

Phase 3: Security Analysis and Peer Review

1. Final Report (PDF):**

   * Introduction: Brief summary of the project.

   * Updated Design: Final protocol specification (highlighting any changes from Phase 1).

   * Security Analysis: A critical analysis of your *own* system. What are its limitations? Under what conditions could it be broken? (e.g., weak random number generation, side-channel attacks, compromise of the server's private key). Propose one potential improvement.

   * Conclusion: Lessons learned.

2. Peer Review: Students will be assigned one other group's Design Document (from Phase 1) and Final Report. They must write a 1-page review assessing the strengths and weaknesses of the other group's design and analysis.

4. Suggested Technical Pathways

To provide variety, students can choose one of the following focus areas:

* Pathway A: The Traditionalist: Implement a simplified TLS-like protocol using RSA for key exchange and AES-GCM for bulk encryption.

* Pathway B: The Modernist: Implement a protocol using Elliptic-Curve Cryptography (ECDH for key exchange and ECDSA for signatures) and ChaCha20-Poly1305 for authenticated encryption.

* Pathway C: The Post-Quantum Explorer: Research and implement a key exchange mechanism using a post-quantum cryptographic algorithm (e.g., using the OpenQuantumSafe library) while

using classical cryptography for the rest of the protocol. This pathway is more research-oriented.

5. Grading Rubric (100 Points Total)

* Phase 1: Design Document (30 Points)
    * Clarity and Completeness (10 pts)
    * Cryptographic Soundness and Justification (15 pts)
    * Threat Model (5 pts)
* Phase 2: Implementation (40 Points)
    * Functional Correctness - Does it work as designed? (20 pts)
    * Code Quality and Documentation (10 pts)
    * Adherence to Security Best Practices in Code (e.g., no hard-coded keys) (10 pts)
* Phase 3: Final Report & Peer Review (30 Points)
    * Depth and Insight of Self-Analysis (20 pts)
    * Quality and Constructiveness of Peer Review (10 pts)