**Project Proposal: Secure VPN Server in C**

**Course:** Operating Systems  
**Instructor:** Sir Abdul Rehman  
**Team Members:**

* Mehwish Zehra (23K-0773)
* Areeza Bashir (23K-0890)
* Tasbiha Saeed (23K-0889)

**Project Title:** Secure VPN Server in C  
**Submission Date:** 23-03-2025

**1. Introduction**

With the increasing importance of online privacy and security, Virtual Private Networks (VPNs) have become essential for protecting data during transmission. This project aims to design and implement a **Secure VPN Server in C** that provides robust encryption, secure communication, and reliable network tunneling.

**2. Objectives**

* **Develop a Secure VPN Server:**  
  Create a client-server architecture that establishes secure, encrypted communication channels.
* **Implement Encryption:**  
  Integrate strong encryption algorithms (e.g., AES, RSA) using libraries like OpenSSL.
* **Support Multiple Clients:**  
  Ensure the server can handle multiple connections simultaneously using multi-threading and proper synchronization.
* **Enhance Network Security:**  
  Provide secure tunneling protocols to protect data from eavesdropping and attacks.
* **Optimize Performance:**  
  Focus on performance and resource management by leveraging low-level C programming for efficiency.

**3. Motivation and Rationale**

* **Real-World Relevance:**  
  VPNs are widely used in both corporate and personal settings, making this project highly relevant.
* **Learning Opportunity:**  
  This project provides hands-on experience with socket programming, encryption techniques, multi-threading, and advanced operating system concepts.
* **Career Advancement:**  
  Skills gained from this project (network security, secure communications, low-level programming) are in high demand in the IT and cybersecurity sectors.

**4. Project Overview and Architecture**

**4.1 System Architecture**

* **Client-Server Model:**
  + **Server Side:** Listens for incoming connections, authenticates clients, and establishes secure tunnels.
  + **Client Side:** Connects to the server, negotiates encryption parameters, and sends/receives data securely.
* **Encryption Module:**
  + Integrate encryption libraries (e.g., OpenSSL) to secure data transfers.
  + Implement key exchange and encryption/decryption processes.
* **Connection Handling:**
  + Use multi-threading to manage concurrent client connections.
  + Ensure synchronization to prevent race conditions and manage resources efficiently.
* **User Authentication:**
  + Implement a simple authentication protocol to verify user identities before granting access to the VPN.

**4.2 Proposed Flow**

1. **Initialization:**  
   The server starts and listens on a designated port.
2. **Connection Request:**  
   A client initiates a connection.
3. **Authentication:**  
   The server authenticates the client.
4. **Key Exchange:**  
   A secure key exchange is performed to establish encryption.
5. **Secure Tunnel Establishment:**  
   A secure tunnel is created for data transmission.
6. **Data Transmission:**  
   Encrypted data is exchanged between the client and server.
7. **Termination:**  
   Connections are securely terminated after use.

**5. Methodology**

1. **Research Phase:**
   * Study existing VPN protocols and encryption methods.
   * Review open-source projects like OpenVPN for design insights.
2. **Design Phase:**
   * Outline system architecture and define modules (encryption, socket handling, multi-threading, authentication).
   * Prepare design documents and flowcharts.
3. **Implementation Phase:**
   * Set up a development environment (Linux, GCC, OpenSSL).
   * Develop individual modules in C and integrate them.
   * Test each module for functionality and security.
4. **Testing and Debugging:**
   * Conduct extensive testing with simulated client connections.
   * Use debugging tools to ensure robustness and performance under load.
5. **Documentation:**
   * Prepare comprehensive documentation covering design, implementation, and user instructions.

**7. Expected Outcomes**

* A fully functional Secure VPN Server implemented in C.
* A working prototype that demonstrates secure communication, encryption, and multi-threaded connection handling.
* Comprehensive documentation including design decisions, code explanations, and testing results.

**8. Challenges and Risks**

* **Encryption Implementation:**  
  Ensuring robust and secure encryption while managing keys safely.
* **Concurrency Issues:**  
  Managing multiple connections with proper synchronization to avoid race conditions.
* **Resource Management:**  
  Efficiently handling memory and processor usage in a multi-threaded environment.
* **Testing:**  
  Simulating real-world network conditions to thoroughly test the VPN server's security and performance.

**9. Conclusion**

This project will allow our team to delve deeply into advanced networking, encryption, and operating system concepts. By developing a Secure VPN Server in C, we aim to build a practical tool with significant real-world applications while enhancing our technical skill set.