Technical Report: Secure Chat Application

Project Overview and Architecture Analysis

1. Executive Summary

This report presents a detailed analysis of a secure chat application implemented in C, utilizing GTK for the graphical user interface and OpenSSL for secure communications. The application demonstrates a robust client-server architecture with emphasis on security, usability, and reliable file transfer capabilities.

2. Technical Stack

Core Technologies

- **Programming Language:** C
- **GUI Framework:** GTK 3
- **Security Layer:** OpenSSL
- **Network Protocol:** TCP/IP with SSL/TLS
- **Build System:** Make (Linux/WSL)

Development Environment

- **Primary Platform:** Linux/WSL
- **Cross-Platform Support:** Windows (via WSL)
- **Required Libraries:**
    - libgtk-3-dev
    - libssl-dev
    - pkg-config

3. Architecture Design

3.1 Server Architecture

- Multi-client support using threaded connections
- SSL/TLS encryption for all communications
- Certificate-based authentication
- Central message broadcasting system
- File transfer management and verification

3.2 Client Architecture

- GTK-based graphical user interface
- Asynchronous message handling
- File transfer progress monitoring
- User authentication system
- Secure connection management

4. Security Implementation

4.1 Communication Security

- SSL/TLS encryption for all data transmission
- Certificate verification system
- Secure key exchange protocols
- Protection against man-in-the-middle attacks

4.2 File Transfer Security

- SHA-256 integrity checking
- User authentication for file reception
- Size limit enforcement (100MB)
- Secure file handling and storage

5. Network Architecture

5.1 Local Network Support

- Direct TCP/IP connections
- Dynamic IP address handling
- Port configuration
- Network discovery

5.2 Internet Connectivity

- Radmin VPN support
- NAT traversal capabilities
- Remote connection handling

6. Features Analysis

6.1 Messaging System

- Real-time text messaging
- Multi-client broadcast support

- Message formatting with Pango markup
- Error handling and recovery

6.2 File Transfer System

- Large file support (up to 100MB)
- Progress monitoring
- Integrity verification
- User confirmation dialog
- Automatic received files directory management

7. User Interface Design

7.1 Server Interface

- User name configuration
- Connection status monitoring
- IP address and port display
- Client management interface

7.2 Client Interface

- Message display area
- Input field for messages
- File transfer progress indication
- Connection status display
- Server connection configuration

8. Build and Deployment

8.1 Build System

- Makefile-based compilation
- Dependency management
- Cross-platform considerations
- Optimization flags

8.2 Deployment Process

- Certificate generation and management
- Library dependency handling
- Directory structure setup
- Permission management

9. Testing and Validation

9.1 Security Testing

- SSL/TLS connection verification
- Certificate validation
- File integrity checking
- Authentication testing

9.2 Performance Testing

- Multi-client handling
- Large file transfers
- Network stability
- Resource utilization

10. Future Improvements

10.1 Potential Enhancements

- Group chat functionality
- End-to-end encryption
- User authentication system
- Message history persistence
- File compression support
- Custom emoticon support

10.2 Scalability Considerations

- Database integration
- Load balancing
- Connection pooling
- Caching mechanisms

11. Conclusion

The secure chat application successfully implements a robust, secure, and user-friendly communication platform. The combination of C, GTK, and OpenSSL provides a solid foundation for secure messaging and file transfer capabilities. The architecture demonstrates good separation of concerns, security consciousness, and consideration for user experience.

12. Appendix

A. Build Instructions

```
# Install dependencies

sudo apt update

sudo apt install build-essential libgtk-3-dev libssl-dev pkg-config

# Build application

make
```

B. Certificate Generation

```
# Using configuration file

openssl req -new -nodes -newkey rsa:4096 -keyout key.pem -out cert.csr -config san.cnf

openssl x509 -req -in cert.csr -signkey key.pem -out cert.pem -days 365 -extensions v3_req
-extfile san.cnf

# Quick self-signed certificate

openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365 -nodes -subj
"/CN=localhost"
```
