Secure Chat and VoIP Application

Network Communications II Project
Electrical and Computer Engineering
Aristotle University of Thessaloniki

Academic Year 2024-2025

Contents

1 Project Overview

1.1 Background

This project is developed as the main assignment for the Computer Networks II course in the Electrical and Computer Engineering department at the Aristotle University of Thessaloniki (AUTH) during the 2024-2025 academic year.

1.2 Objectives

The primary objectives of this Peer-to-Peer (P2P) Chat and VoIP application are:

- Implement a secure peer-to-peer communication platform
- Provide real-time text-based chat functionality
- Enable Voice over IP (VoIP) communication
- Ensure end-to-end encryption for both text and voice data
- Demonstrate practical network programming using Java

2 Technologies and Tools

2.1 Technical Stack

- **Programming Language:** Java
- **Core Libraries:**
  - `java.net` for networking
  - `javax.sound` for audio capture and playback
- **Communication Protocol:** UDP (User Datagram Protocol)
- **Development Tools:**
  - Git for version control
  - Visual Studio Code for development
  - Wireshark for packet analysis

3 System Architecture

3.1 Components

The application consists of two primary Java classes:

1. `App.java`: Main application class handling GUI, network communication, and call management
2. `SecurityModule.java`: Handles encryption and secure key exchange

3.2 App.java

The `App.java` file serves as the main entry point for the application. It initializes the graphical user interface (GUI), handles user interactions, and manages the core functionalities, including:

- **Chat Interface:** Provides a text area for displaying messages and an input field for sending chat messages. It supports both plaintext and encrypted communications.
- **VoIP Communication:** Manages real-time voice data transmission, including initialization of audio devices (microphone and speakers) and threading for concurrent send/receive operations.
- **Networking:** Utilizes UDP sockets for communication, specifying separate ports for chat and voice data.
- **Event Handling:** Implements actions for sending messages and initiating or terminating calls using Java Swing components.
- **Secure Communication:** Integrates with `SecurityModule.java` to handle key exchange, encryption, and decryption processes.

3.3 SecurityModule.java

The `SecurityModule.java` file encapsulates the security features of the application, implementing robust mechanisms for encryption and key exchange using a hybrid encryption approach:

- **Public Key Infrastructure (PKI):** Generates RSA key pairs (2048-bit) for asymmetric encryption and facilitates secure initial key exchange.
- **Hybrid Encryption:** Combines the *security of RSA* for symmetric key exchange with the *efficiency of AES (256-bit)* for ongoing message encryption during communication.
- **Key Management:** Includes mechanisms to generate AES keys and securely transmit them to the remote party using RSA encryption.
- **Message Security:** Provides methods for encrypting and decrypting messages using AES, ensuring data confidentiality during chat transmissions.

Key Exchange Protocol The key exchange process ensures a secure connection between the communicating parties:

1. **Initial Public Key Exchange:** The application exchanges RSA public keys between the local and remote party.
2. **Symmetric Key Generation:** A 256-bit AES key is generated for secure communication.
3. **Encrypted Symmetric Key Transmission:** The AES key is encrypted using the remote party's RSA public key and transmitted securely.
4. **Secure Connection Establishment:** Once the symmetric key is received and decrypted, the connection is secured for encrypted message exchange.

4 Technical Implementation Details

4.1 Audio Configuration

- **Audio Format:**
  - Sampling Rate: 8000 Hz
  - Bit Depth: 8-bit
  - Channels: Mono
  - Encoding: PCM (Pulse Code Modulation)
- **Audio Devices:**
  - Microphone input using `TargetDataLine`
  - Speaker output using `SourceDataLine`

5 Graphical User Interface

5.1 Interface Components

The application features a simple GUI with the following elements:

- Text Field for composing messages
- Text Area for displaying sent and received messages
- "Send" button for transmitting text messages
- "Call" button for initiating VoIP communication

6 Wireshark

6.1 Chat data example

The images below illustrate text messages exchanged through the app:

udp.port == 8000

No.	Time	Source	Destination	Protocol	Lengtl	Info
3	11.331959	192.168.1.87	192.168.1.87	UDP	435	8000 → 8000 Len=403
4	11.397161	192.168.1.87	192.168.1.87	UDP	390	8000 → 8000 Len=358

Figure 1: Packets sent while establishing secure connection.

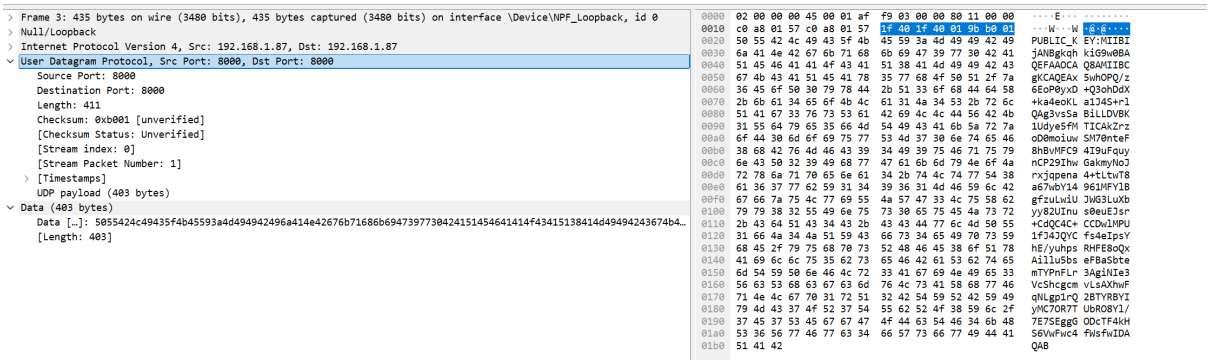

Figure 2: Network/Internet header and payload of public key exchange message.

6.2 Voice data example

The images below illustrate examples of voice data exchanged through the app:

udp.port == 8001

No.	Time	Source	Destination	Protocol	Length	Info
9	7.372983	192.168.1.87	192.168.1.87	UDP	1056	8001 → 8001 Len=1024
10	7.500117	192.168.1.87	192.168.1.87	UDP	1056	8001 → 8001 Len=1024
11	7.626926	192.168.1.87	192.168.1.87	UDP	1056	8001 → 8001 Len=1024
12	7.753099	192.168.1.87	192.168.1.87	UDP	1056	8001 → 8001 Len=1024
13	7.879399	192.168.1.87	192.168.1.87	UDP	1056	8001 → 8001 Len=1024
14	8.005867	192.168.1.87	192.168.1.87	UDP	1056	8001 → 8001 Len=1024
15	8.131954	192.168.1.87	192.168.1.87	UDP	1056	8001 → 8001 Len=1024
16	8.257655	192.168.1.87	192.168.1.87	UDP	1056	8001 → 8001 Len=1024
17	8.383672	192.168.1.87	192.168.1.87	UDP	1056	8001 → 8001 Len=1024
18	8.510398	192.168.1.87	192.168.1.87	UDP	1056	8001 → 8001 Len=1024
19	8.636934	192.168.1.87	192.168.1.87	UDP	1056	8001 → 8001 Len=1024
20	8.763822	192.168.1.87	192.168.1.87	UDP	1056	8001 → 8001 Len=1024
21	8.890127	192.168.1.87	192.168.1.87	UDP	1056	8001 → 8001 Len=1024
22	9.015892	192.168.1.87	192.168.1.87	UDP	1056	8001 → 8001 Len=1024
23	9.142134	192.168.1.87	192.168.1.87	UDP	1056	8001 → 8001 Len=1024
24	9.166464	192.168.1.87	192.168.1.87	UDP	672	8001 → 8001 Len=640

Figure 3: All packets exchanged during a voice call session as captured by Wireshark.

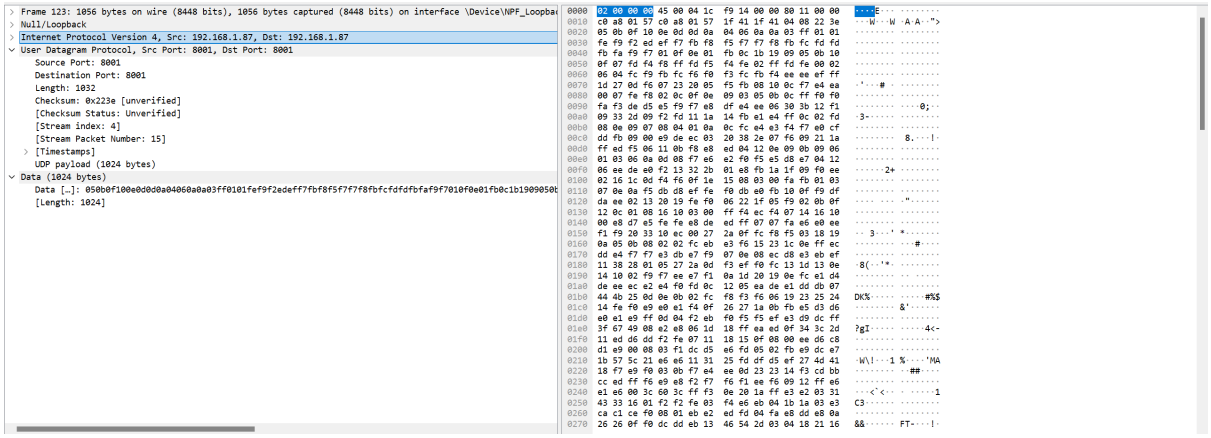

Figure 4: Network/Internet header and payload of a specific voice data packet.

7 Conclusion

The P2P Chat and VoIP application demonstrates a practical implementation of secure, real-time communication using Java networking and encryption technologies. By leveraging UDP protocols, native Java libraries, and a hybrid encryption approach, the project provides a foundational understanding of network programming, concurrency, and multimedia data exchange between peers.