

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	2
1.1	Background	2
1.2	Objectives	2
2	Technologies and Tools	2
2.1	Technical Stack	2
3	System Architecture	2
3.1	Components	2
3.2	App.java	3
3.3	SecurityModule.java	3
4	Technical Implementation Details	4
4.1	Audio Configuration	4
5	Graphical User Interface	4
5.1	Interface Components	4
6	Wireshark	4
6.1	Chat data example	4
6.2	Voice data example	5
7	Conclusion	6

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

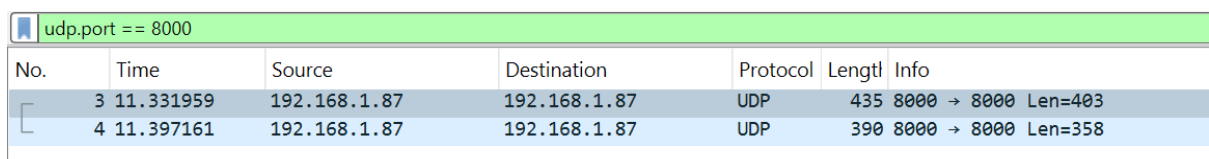
The application features a simple GUI with the following elements:

- 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:



The image shows a Wireshark packet capture window with a green filter bar at the top containing the text 'udp.port == 8000'. Below the filter bar is a table of captured packets. The table has columns for No., Time, Source, Destination, Protocol, Length, and Info. Two packets are listed: packet 3 at time 11.331959 and packet 4 at time 11.397161. Both packets are UDP and have source and destination IP addresses of 192.168.1.87. The info column for packet 3 shows '435 8000 → 8000 Len=403' and for packet 4 shows '390 8000 → 8000 Len=358'.

No.	Time	Source	Destination	Protocol	Length	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.

> Frame 3: 435 bytes on wire (3480 bits), 435 bytes captured (3480 bits) on interface \Device\NPF_{Loopback, id 0 > Null/Loopback > Internet Protocol Version 4, Src: 192.168.1.87, Dst: 192.168.1.87 > User Datagram Protocol, Src Port: 8000, Dst Port: 8000		0000 02 00 00 00 45 00 01 af f9 03 00 00 00 11 00 00 0010 c0 a8 01 57 c0 a8 01 57 14 4b 14 4b 01 5b 00 01 0020 50 55 42 4c 49 43 5f 4b 45 59 3a 4d 49 49 42 49 0030 6a 41 4e 42 67 6b 71 68 6b 69 47 39 77 30 42 41 0040 51 45 46 41 41 4f 43 41 51 38 41 4d 49 49 42 43 0050 67 4b 43 41 51 45 41 78 35 77 68 4f 50 51 2f 7a 0060 36 45 6f 50 30 79 78 44 2b 51 33 6f 68 44 64 58 0070 2b 6b 61 34 65 6f 4b 4c 61 31 4a 34 53 2b 72 6c 0080 51 41 67 33 76 73 53 61 42 69 4c 4c 44 56 42 4b 0090 31 55 64 79 65 35 66 4d 54 49 43 41 6b 5a 72 7a 00a0 6f 44 30 6d 6f 69 75 77 53 4d 37 30 6e 74 65 46 00b0 38 68 42 76 4d 46 43 39 34 49 39 75 46 71 75 79 00c0 6e 43 50 32 39 49 68 77 47 61 6b 6d 79 4e 6f 4a 00d0 72 78 6a 71 70 65 6e 61 34 2b 74 4c 74 77 54 38 00e0 61 36 37 77 62 59 51 34 39 36 31 4d 46 59 6c 42 00f0 67 66 7a 75 4c 77 69 55 4a 57 47 33 4c 75 58 62 0100 79 79 38 32 55 49 6e 75 73 39 65 75 45 4a 73 72 0110 2b 43 64 51 43 34 43 2b 43 43 44 77 6c 4d 50 55 0120 31 66 4a 34 4a 51 59 43 66 73 34 65 49 70 73 59 0130 68 45 2f 79 75 68 70 73 52 48 46 45 38 6f 51 78 0140 41 69 6c 6c 75 35 62 73 65 46 42 61 53 62 74 65 0150 6d 54 59 50 6e 46 4c 72 33 41 67 69 4e 49 65 33 0160 56 63 53 68 63 67 63 6d 76 4c 73 41 58 68 77 46 0170 71 4e 4c 67 70 31 72 51 32 42 54 59 52 42 59 49 0180 79 4d 43 37 4f 52 37 54 55 62 52 4f 38 59 6c 2f 0190 37 45 37 53 45 67 67 47 4f 44 63 54 46 34 6b 48 01a0 53 36 56 77 46 77 63 34 66 57 73 66 77 49 44 41 01b0 51 41 42
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

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.

```

> Frame 123: 1056 bytes on wire (8448 bits), 1056 bytes captured (8448 bits) on interface \Device\NPF_{...}
> Null/Loopback
> Internet Protocol Version 4, Src: 192.168.1.87, Dst: 192.168.1.87
  User Datagram Protocol, Src Port: 8001, Dst Port: 8001
    Source Port: 8001
    Destination Port: 8001
    Length: 1032
    Checksum: 0x223e [unverified]
    [Checksum Status: Unverified]
    [Stream Index: 4]
    [Stream Packet Number: 15]
    [Timestamps]
  UDP payload (1024 bytes)
    Data (1024 bytes)
      Data [..]: 050b0f100e0d0a04060a0e03ff8101fef9f2edeff7fb8f8f5f7f7f8fbfcfdcfbaf9f7810f0e01f0dc1b1909050e
      [Length: 1024]
0000 02 00 00 00 45 00 04 1c f9 14 00 00 80 11 00 00  E.....
0010 c0 a8 01 37 c0 a8 01 57 1f 41 1f 41 04 08 22 3e  W...W..A..>
0020 95 0b 0f 10 0e 0d 0a 04 06 0a 0e 03 ff 01 01  ..
0030 fe f9 f2 ed ef f7 fb f8 f5 f7 f7 f8 fb fc fd fd  ..
0040 fb fa f9 f7 01 0f 0e 01 fb 0c 1b 19 09 05 0b 10  ..
0050 0f 07 fd f4 f8 ff fd f5 f4 fe 02 ff fd fe 00 02  ..
0060 06 04 fc f9 fb fc f6 f0 f3 fc fb f4 ee ee ef ff  ..
0070 1d 27 0d f6 07 23 20 05 f5 fb 08 10 0c f7 e4 ea  ..
0080 00 07 fe f8 02 8c 0f 0a 09 03 05 0b 8c ff f0 f0  ..
0090 fa f3 de d5 e5 f9 f7 a8 df e4 ee 06 30 3b 12 f1  ..
00a0 09 33 2d 09 f2 fd 11 1a 14 fb e1 e4 ff 0c 02 fd  -3-
00b0 08 0e 09 07 08 04 01 0a 8c fc e4 a3 f4 f7 e0 cf  ..
00c0 dd fb 09 00 e9 de ec 03 20 38 2e 07 f6 09 21 1a  ..
00d0 ff ad f5 06 11 00 f0 e8 e0 04 12 0e 09 00 09 06  ..
00e0 01 03 0e 0a 0d 08 f7 e6 a2 f0 f5 a5 d8 e7 04 12  ..
00f0 06 ee de e0 f2 13 32 2b 01 e8 fb 1a 1f 09 f0 ee  ..
0100 92 16 1c 0d f4 f6 0f 1a 15 00 03 00 fa fb 01 03  ..
0110 07 0e 0a f5 db d8 ef fe f0 db ea fb 10 0f f9 df  ..
0120 da ee 02 13 20 19 fe f0 06 22 1f 05 f9 02 0b 0f  ..
0130 12 0c 01 0e 16 10 03 00 ff f4 ec f4 07 14 16 10  ..
0140 00 e8 d7 a5 fe fe e8 de ad ff 07 07 fa e5 e0 ee  ..
0150 f1 f9 20 33 10 ec 00 27 2a 0f fc f8 f5 03 18 19  ..
0160 0a 05 00 00 02 02 fc eb a3 f6 15 23 1c 0e ff ec  ..
0170 dd e4 f7 f7 e3 db e7 f9 07 0e 08 ec d8 e3 eb ef  ..
0180 11 38 28 01 05 27 2a 0d f3 ef f0 fc 13 d1 d3 0e  -8(-
0190 14 10 02 f9 f7 ee e7 f1 0e 1d 20 19 0e fc e1 04  ..
01a0 de ee ec e2 e4 f0 fd 0c 12 05 ea de e1 dd db 07  ..
01b0 44 4b 25 0d 0e 0b 02 fc f8 f3 f6 06 19 23 25 24  DNS.....#5$
01c0 14 fe f0 e9 e0 e1 f4 0f 26 27 1a 0b fb e5 03 06  ..
01d0 e0 e1 e9 ff 0d 04 f2 eb f0 f5 f5 ef e3 d9 dc ff  ..
01e0 3f 07 49 08 e2 e8 06 1d 18 ff ea ed 0f 34 3c 2d  ?gi.....4c-
01f0 11 ed d6 dd f2 fe 07 11 18 15 0f 08 00 0e 06 c8  ..
0200 d1 e9 00 00 03 f1 dc 05 a6 fd 05 02 fb e9 dc e7  ..
0210 1b 57 5c 21 e6 e6 11 31 25 fd df d5 ef 27 4d 41  -W|...1 %...MA
0220 18 f7 e9 f0 03 00 f7 e4 ee 0d 23 14 f3 cd bb  ..
0230 cc ed ff f6 e9 e8 f2 f7 f6 f1 ee f6 09 12 ff e6  ..
0240 e1 e6 00 3c 60 3c ff f3 0e 20 1a ff e3 e2 03 31  ..
0250 43 33 16 01 f2 f2 fe 03 f4 e6 eb 04 1b 1a 03 e3  C3
0260 ca c1 ce f0 08 01 eb e2 ed fd 04 fa e8 dd e8 8a  ..
0270 26 26 0f f0 dc dd eb 13 46 54 2d 03 04 18 21 16  &&.....FT...1-

```

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.