# Report

**Student:** Youqing Fu  
**Student ID:** a1981355  
**Course:** Secure Programming  
**Date:** 2025-10-19

### 1. Introduction

The project implements a secure messaging system with support for multiple servers and a federation model. Each server is designed to register users, advertise public keys, and exchange messages across server boundaries with the help of an introducer. As part of this peer review, my role is to verify whether the key functionalities behave as expected, including:

Successful introducer and server setup

User registration and key advertisement

Cross-server message delivery

Direct message (DM) functionality test (positive or negative)

### 2. Experiment Setup

**Environment:**

macOS 14.5

Python 3.13

Virtual environment with dependencies installed

5 separate terminal sessions for different components

**Setup Process:**

**Introducer Launch:** python3 introducer.py --host 127.0.0.1 --port 9000

**Server A Launch:** python3 server.py --host 127.0.0.1 --port 9001

**Server B Launch:** python3 server.py --host 127.0.0.1 --port 9002

**Client A (Alice) Connect:** python3 client.py --user alice --server ws://127.0.0.1:9001

**Client B (Bob) Connect:** python3 client.py --user bob --server ws://127.0.0.1:9002

These five terminal sessions were used simultaneously to simulate a real federated communication environment.

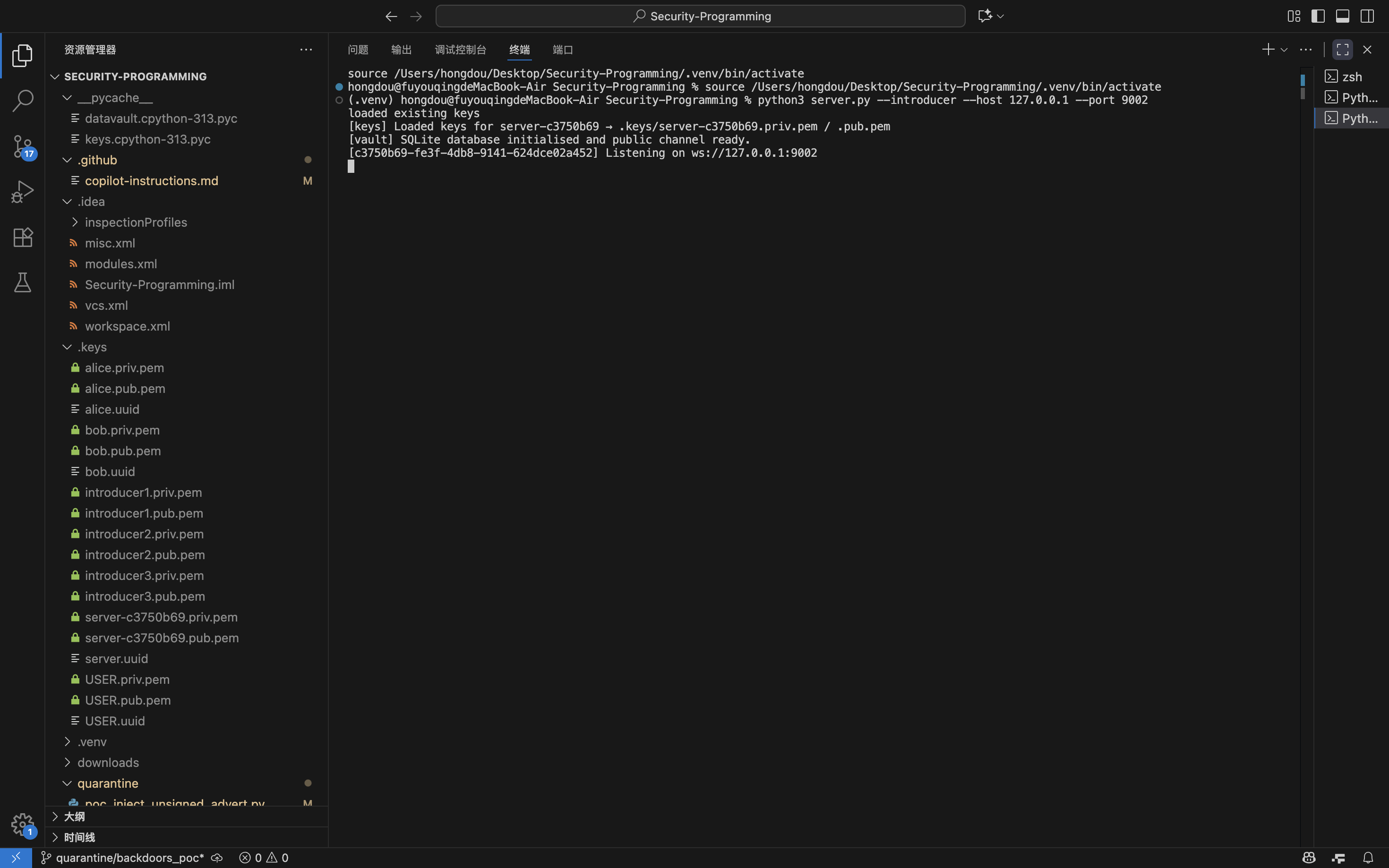
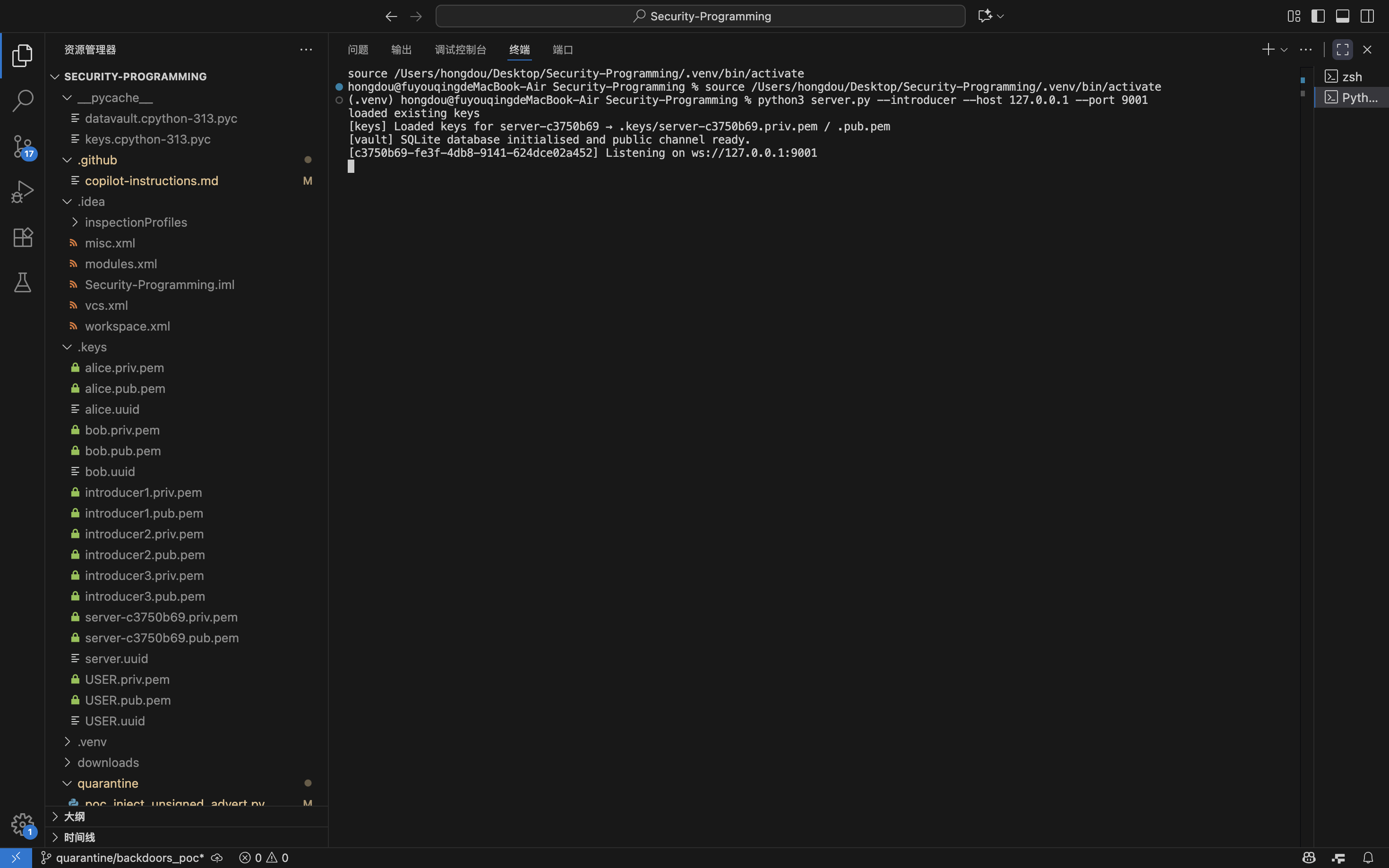
### 3. Results & Discussion

#### 3.1 Introducer and Server Setup

**Result:** The introducer and both servers launched successfully.

Introducer started on port 9000

Server A and B initialized their public channels on ports 9001 and 9002

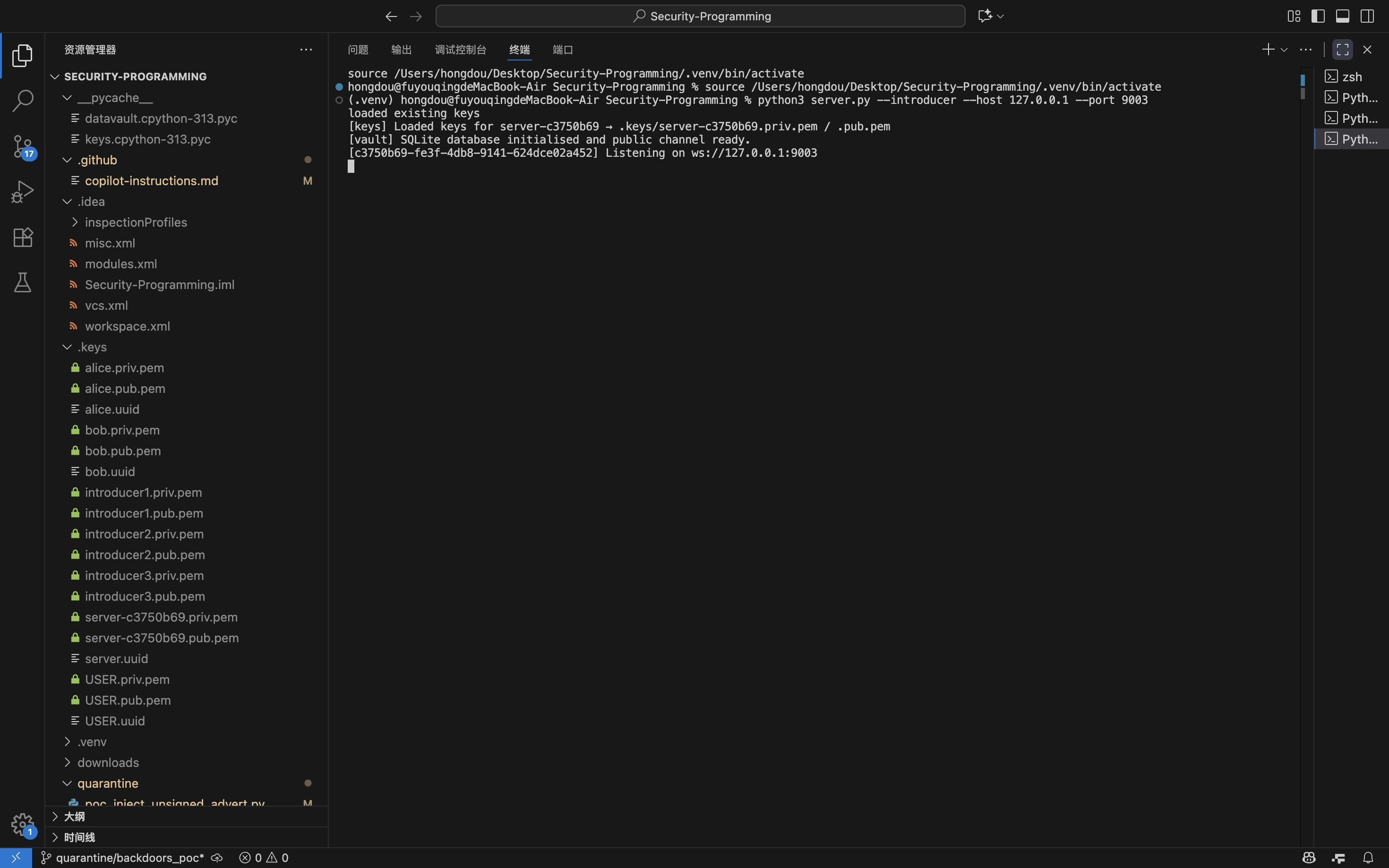


#### 3.2 User Registration and Key Exchange

**Result:** Both Alice and Bob successfully registered on their respective servers.

USER\_ADVERTISE messages were exchanged.

Server logs confirmed federation and key advertisement.

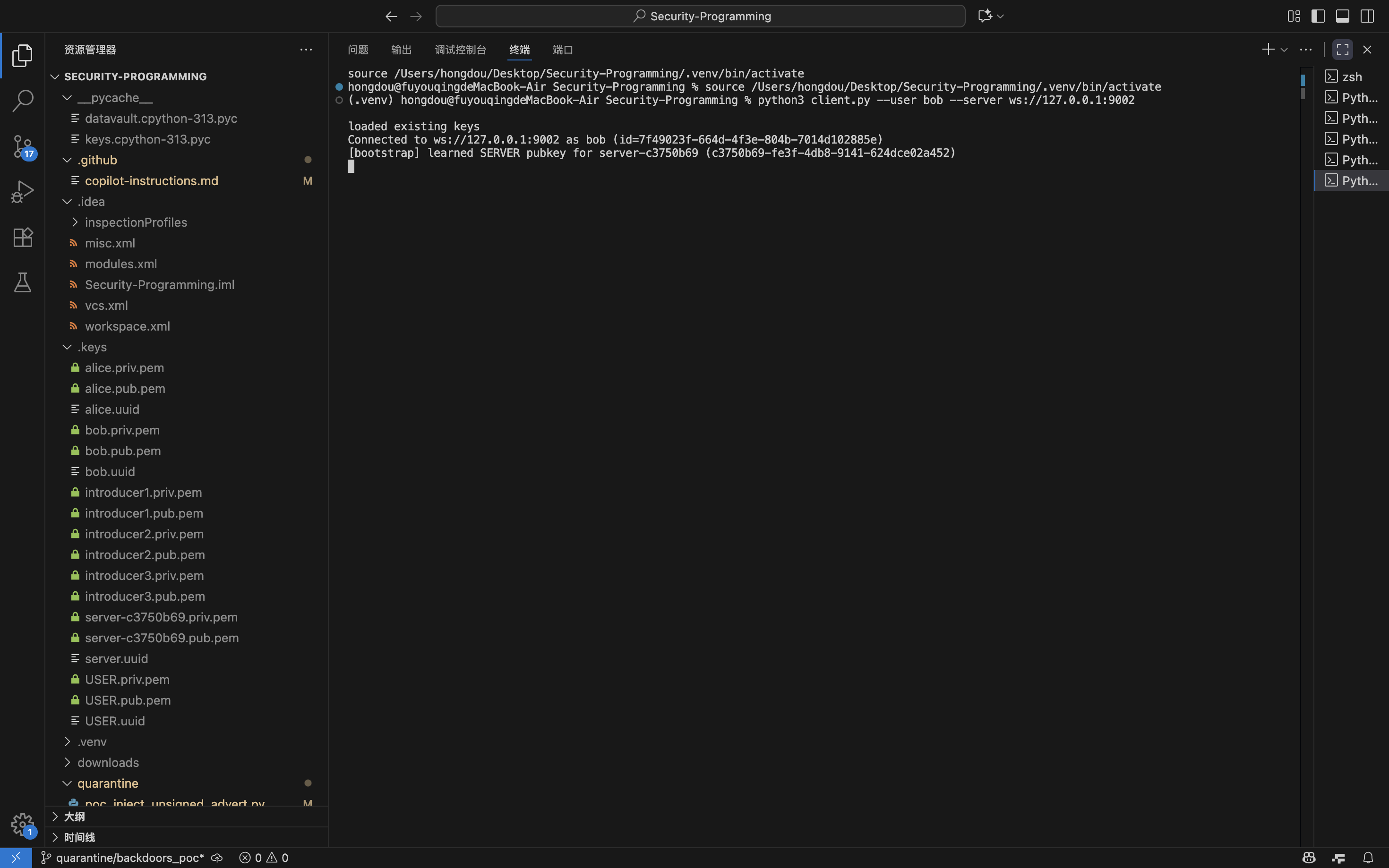
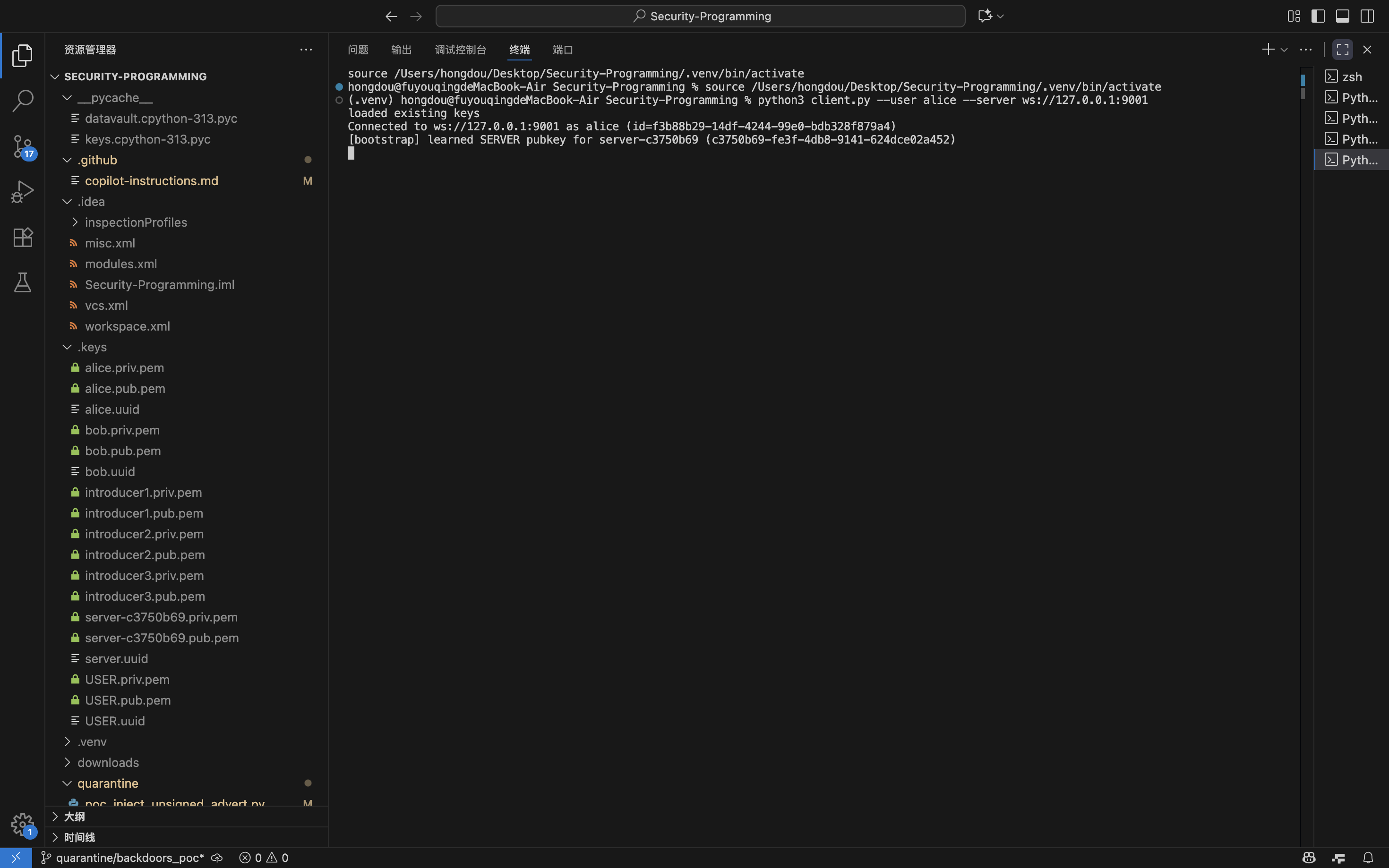


#### 3.3 Cross-Server Broadcast Communication

**Result:** Alice (on Server A) sent a broadcast message which Bob (on Server B) successfully received.

This demonstrates correct federation and message routing.

The bootstrap and federation logs confirmed successful connection and message delivery.



#### 3.4 Direct Message (DM) Functionality – Negative Test

**Discussion:**  
Although direct messaging is not supported, the negative test confirms that the system handles unknown commands gracefully (no crash or exception). This demonstrates stable client behavior and server resilience.

### 4. Conclusion

**Introducer and servers** initialize correctly and exchange keys as expected.

**Cross-server federation** is implemented successfully, enabling broadcast messaging across different servers.

**Direct messaging (DM)** is not supported in this build. The system remains stable but lacks this optional feature.

### 5. Recommendations

**Implement DM feature:** Adding a /dm command or --to flag would significantly enhance the system’s functionality.

**Improve client feedback:** Provide a clear message when an unsupported command is entered.

**Add unit tests:** For key exchange and federation routing to ensure long-term stability.

**Final Evaluation:**  
The project meets most of the functional expectations for secure federation and broadcast communication. While the absence of direct messaging limits user interaction features, the system’s core mechanisms — including key management, federation, and broadcast messaging — are stable and reliable.