

M Abdurrahman Khan

22i-1148 H

TESTS

Running Tests

All test scripts are located in the `tests/` directory.

Test 1: Tamper Test (SIG_FAIL)

Tests that message tampering is detected via signature verification failure.

Terminal 1:

```
python -m app.server
```

Terminal 2:

```
python -m tests.tamper_test
```

When prompted:

- Enter your registered email
- Enter your password

Expected Output:

```
==== TAMPER TEST ====
Performing login...
Login successful!
Session key established
```

1. Sending normal (valid) message...
 Success: received
2. Sending TAMPERED message (flipped bit in ciphertext)...
 ✓ TAMPER DETECTED: SIG_FAIL: Signature verification failed

- ✓ Signature verification failed as expected!

Evidence: Screenshot showing `SIG_FAIL` error.

The screenshot shows two adjacent Windows PowerShell windows. The left window, titled 'Administrator: Windows PowerShell', displays a secure chat session between a client and a server. It shows messages being exchanged, files being saved to 'transcripts', and clients disconnecting. The right window, also titled 'Administrator: Windows PowerShell', shows a tamper test. It starts by connecting to the server at localhost:8888, validating the certificate, and performing a login. It then sends two messages: a normal message and a tampered message (with a flipped bit in the ciphertext). The tampered message triggers a 'SIG_FAIL' error, indicating that signature verification failed as expected. Both windows have a green background with a pine branch pattern.

```

Administrator: Windows PowerShell
Server receipt saved to: transcripts\server_receipt_127.0.0.1_51693.json
Transcript exported to: transcripts\server_127.0.0.1_51693.export.txt
Session completed. All files saved in transcripts/ directory.
Client ('127.0.0.1', 51693) disconnected
Client connected from ('127.0.0.1', 50827)
Error handling client: Connection closed
Client ('127.0.0.1', 50827) disconnected

Server shutting down...
(.venv) PS C:\Users\admin\Downloads\SecureChat-main_i221148\SecureChat-main> python -m app.server
Google app.server
ChromServer listening on localhost:8888
Client connected from ('127.0.0.1', 63341)
Session established. Waiting for messages...
Received: hello this is a text message
Server receipt saved to: transcripts\server_receipt_127.0.0.1_63341.json
Transcript exported to: transcripts\server_127.0.0.1_63341.export.txt
Session completed. All files saved in transcripts/ directory.
Client ('127.0.0.1', 63341) disconnected

Server shutting down...
(.venv) PS C:\Users\admin\Downloads\SecureChat-main_i221148\SecureChat-main> python -m app.server
control Server listening on localhost:8888
Client connected from ('127.0.0.1', 60541)
Session established. Waiting for messages...
Received: Hello, this is a test message
Error handling client: Connection closed
Client ('127.0.0.1', 60541) disconnected

Administrator: Windows PowerShell
File "<frozen runpy>", line 88, in _run_code
  File "C:/Users/admin/Downloads/SecureChat-main_i221148/SecureChat-main/app/client/main.py", line 1, in <module>
    main()
  File "C:/Users/admin/Downloads/SecureChat-main_i221148/SecureChat-main/app/client/main.py", line 1, in main
    print(f"Error: {e}")
KeyboardInterrupt
(.venv) PS C:\Users\admin\Downloads\SecureChat-main_i221148\SecureChat-main> python -m app.server
Connected to server at localhost:8888
Server certificate validated

*** TAMPER TEST ===
Performing login...
Enter email for login: test@example.com
Enter password: mypassword123
Login successful!
Session key established

1. Sending normal (valid) message...
Success: received

2. Sending TAMPERED message (flipped bit in ciphertext)...
✓ TAMPER DETECTED: SIG_FAIL: Signature verification failed
✓ Signature verification failed as expected!

*** TEST COMPLETE ***
Evidence: SIG_FAIL error demonstrates tamper detection
(.venv) PS C:\Users\admin\Downloads\SecureChat-main_i221148\SecureChat-main>

```

Test 2: Replay Test (REPLAY)

Tests that replay attacks are prevented via sequence number checking.

Terminal 1:

```
python -m app.server
```

Terminal 2:

```
python -m tests.replay_test
```

When prompted:

- Enter your registered email
- Enter your password

Expected Output:

```

==== REPLAY TEST ====
Performing login...
Login successful!
Session key established

1. Sending first message (seqno=1)...
Success: received

2. Sending second message (seqno=2) - valid...
Success: received

3. REPLAY ATTACK: Resending message with seqno=1 (already used)...
✓ REPLAY DETECTED: REPLAY: Sequence number 1 already seen
✓ Replay protection works as expected!

```

Evidence: Screenshot showing REPLAY error.

The screenshot shows two side-by-side Windows PowerShell windows. The left window is titled 'Administrator: Windows PowerShell' and shows the server log for handling two client connections. The right window is also titled 'Administrator: Windows PowerShell' and shows the client log for performing a replay attack on the second connection. Both windows have a green background with a pine branch pattern.

```

Administrator: Windows PowerShell
Server listening on localhost:8888
Client connected from ('127.0.0.1', 63341)
Session established. Waiting for messages...
Received: hello this is a text message
Server receipt saved to: transcripts\server_receipt_127.0.0.1_63341.json
Transcript exported to: transcripts\server_127.0.0.1_63341.export.txt
Session completed. All files saved in transcripts/ directory.
Client ('127.0.0.1', 63341) disconnected

Good Server shutting down...
Ctrl+Q(.venv) PS C:\Users\admin\Downloads\SecureChat-main_i221148\SecureChat-main> python -m app.server
Server listening on localhost:8888
Client connected from ('127.0.0.1', 60541)
Session established. Waiting for messages...
Received: Hello, this is a test message
Error handling client: Connection closed
Client ('127.0.0.1', 60541) disconnected

Server shutting down...
(.venv) PS C:\Users\admin\Downloads\SecureChat-main_i221148\SecureChat-main> python -m app.server
Server listening on localhost:8888
Control+Q Client connected from ('127.0.0.1', 52280)
Session established. Waiting for messages...
Received: First message
Received: Second message
Error handling client: Connection closed
Client ('127.0.0.1', 52280) disconnected
code

Administrator: Windows PowerShell
(.venv) PS C:\Users\admin\Downloads\SecureChat-main_i221148\SecureChat-main> python -m app.server
Connected to server at localhost:8888
Error: [WinError 10054] An existing connection was forcibly closed by the remote host
(.venv) PS C:\Users\admin\Downloads\SecureChat-main_i221148\SecureChat-main> python -m app.server
Connected to server at localhost:8888
Server certificate validated

==== REPLAY TEST ====
Performing login...
Enter email for login: test@example.com
Enter password: mypassword123
Login successful!
Session key established

1. Sending first message (seqno=1)...
Success: received

2. Sending second message (seqno=2) - valid...
Success: received

3. REPLAY ATTACK: Resending message with seqno=1 (already used)...
This should be rejected with REPLAY error
✓ REPLAY DETECTED: REPLAY: Sequence number must be strictly increasing
✓ Replay protection works as expected!

==== TEST COMPLETE ====
Evidence: REPLAY error demonstrates replay protection
(.venv) PS C:\Users\admin\Downloads\SecureChat-main_i221148\SecureChat-main>

```

Test 3: Invalid Certificate Test (BAD_CERT)

Tests that invalid/self-signed certificates are rejected.

Step 1: Generate Invalid Certificate

```
python scripts/gen_invalid_cert.py
```

Step 2: Backup Valid Certificates

```
copy certs\client.crt certs\client.crt.backup  
copy certs\client.key certs\client.key.backup
```

Step 3: Replace with Invalid Certificate

```
copy certs\invalid.crt certs\client.crt  
copy certs\invalid.key certs\client.key
```

Step 4: Start Server

```
python -m app.server
```

Step 5: Try to Connect

```
python -m app.client
```

Expected Output:

```
Connected to server at localhost:8888  
Error: BAD_CERT: Self-signed certificate rejected
```

Step 6: Restore Valid Certificates

```
copy certs\client.crt.backup certs\client.crt  
copy certs\client.key.backup certs\client.key
```

Evidence: Screenshot showing `BAD_CERT` error.

```

Administrator: Windows PowerShell
Client connected from ('127.0.0.1', 60541)
Session established. Waiting for messages...
Received: Hello, this is a test message
Error handling client: Connection closed
Client ('127.0.0.1', 60541) disconnected

Server shutting down...
(.venv) PS C:\Users\admin\Downloads\SecureChat-main_i221148\SecureChat-main> app.server
Good
Server listening on localhost:8888
Client connected from ('127.0.0.1', 52280)
Session established. Waiting for messages...
Received: First message
Received: Second message
Error handling client: Connection closed
Client ('127.0.0.1', 52280) disconnected
Client connected from ('127.0.0.1', 52721)
Error handling client: Connection closed
Client ('127.0.0.1', 52721) disconnected
Client connected from ('127.0.0.1', 52723)
Error handling client: Connection closed
Client ('127.0.0.1', 52723) disconnected
Client ('127.0.0.1', 52727) disconnected

Server shutting down...
(.venv) PS C:\Users\admin\Downloads\SecureChat-main_i221148\SecureChat-main> app.server
Server listening on localhost:8888
Client connected from ('127.0.0.1', 52727)
Client ('127.0.0.1', 52727) disconnected
code

Administrator: Windows PowerShell
server_hello_data = receive_message(sock)
File "C:\Users\admin\Downloads\SecureChat-main_i221148\SecureChat-main\app\client.py", line 42, in receive_message
    chunk = sock.recv(4096)
KeyboardInterrupt
(.venv) PS C:\Users\admin\Downloads\SecureChat-main_i221148\SecureChat-main> python scripts/gen_invalid_cert.py
Invalid self-signed certificate generated:
Private key: certs\invalid.key
Certificate: certs\invalid.crt
Note: This certificate is NOT signed by your CA and should be rejected
(.venv) PS C:\Users\admin\Downloads\SecureChat-main_i221148\SecureChat-main> copy certs\client.crt certs\client.crt.backup
up
(.venv) PS C:\Users\admin\Downloads\SecureChat-main_i221148\SecureChat-main> copy certs\invalid.crt certs\client.crt
(.venv) PS C:\Users\admin\Downloads\SecureChat-main_i221148\SecureChat-main> copy certs\invalid.key certs\client.key.backup
up
(.venv) PS C:\Users\admin\Downloads\SecureChat-main_i221148\SecureChat-main> python -m app.client
Connected to server at localhost:8888
Error: [winError 10054] An existing connection was forcibly closed by the remote host
(.venv) PS C:\Users\admin\Downloads\SecureChat-main_i221148\SecureChat-main> python -m app.client
Connected to server at localhost:8888
Error: BAD_CERT: Certificate not signed by trusted CA
(.venv) PS C:\Users\admin\Downloads\SecureChat-main_i221148\SecureChat-main> copy certs\client.crt.backup certs\client.crt
(.venv) PS C:\Users\admin\Downloads\SecureChat-main_i221148\SecureChat-main> copy certs\client.key.backup certs\client.key
(.venv) PS C:\Users\admin\Downloads\SecureChat-main_i221148\SecureChat-main>

```

Test 4: Non-Repudiation Verification

Tests offline verification of transcript and SessionReceipt.

Step 1: Run a Chat Session

Terminal 1:

```
python -m app.server
```

Terminal 2:

```
python -m app.client
```

- Login with your credentials
- Send 2-3 messages
- Type `quit` to end session

Step 2: Verify Transcript and Receipt

```
python -m tests.verify_transcript \
--transcript transcripts/client_localhost_8888.export.txt \
--cert certs/client.crt \
--expected-cn client.local
```

```
Or           python -m tests.verify_transcript --transcript  
transcripts/client_localhost_8888.export.txt --cert certs/client.crt  
--expected-cn client.local
```

Expected Output:

```
=====  
TRANSCRIPT & SESSION RECEIPT VERIFICATION  
=====  
  
1. Loading certificate: certs/client.crt  
2. Validating certificate against CA: certs/ca.crt  
   ✓ Certificate is valid and trusted  
  
3. Loading transcript: transcripts/client_localhost_8888.export.txt  
   Found 3 message entries  
  
4. Verifying message signatures:  
   Message 1: ✓ Signature valid  
   Message 2: ✓ Signature valid  
   Message 3: ✓ Signature valid  
  
   ✓ All message signatures are valid  
  
5. Computing transcript hash:  
   Computed hash: abc123def456...  
  
6. Loading SessionReceipt:  
   Auto-detected receipt file: transcripts/client_receipt_localhost_8888.json  
   Peer: client  
   First seq: 1  
   Last seq: 3  
   Transcript hash: abc123def456...  
  
7. Verifying receipt hash matches transcript:  
   ✓ Receipt hash matches computed transcript hash  
  
8. Verifying receipt signature:  
   ✓ Receipt signature is valid  
  
=====  
VERIFICATION RESULT: ✓ ALL CHECKS PASSED  
=====
```

Step 3: Test Tampering Detection

Edit the transcript file (change any character), then run verification again:

```

# Edit transcript file (add/remove a character)
# Then verify again
python -m tests.verify_transcript \
--transcript transcripts/client_localhost_8888.export.txt \
--cert certs/client.crt \
--expected-cn client.local

```

Expected Output:

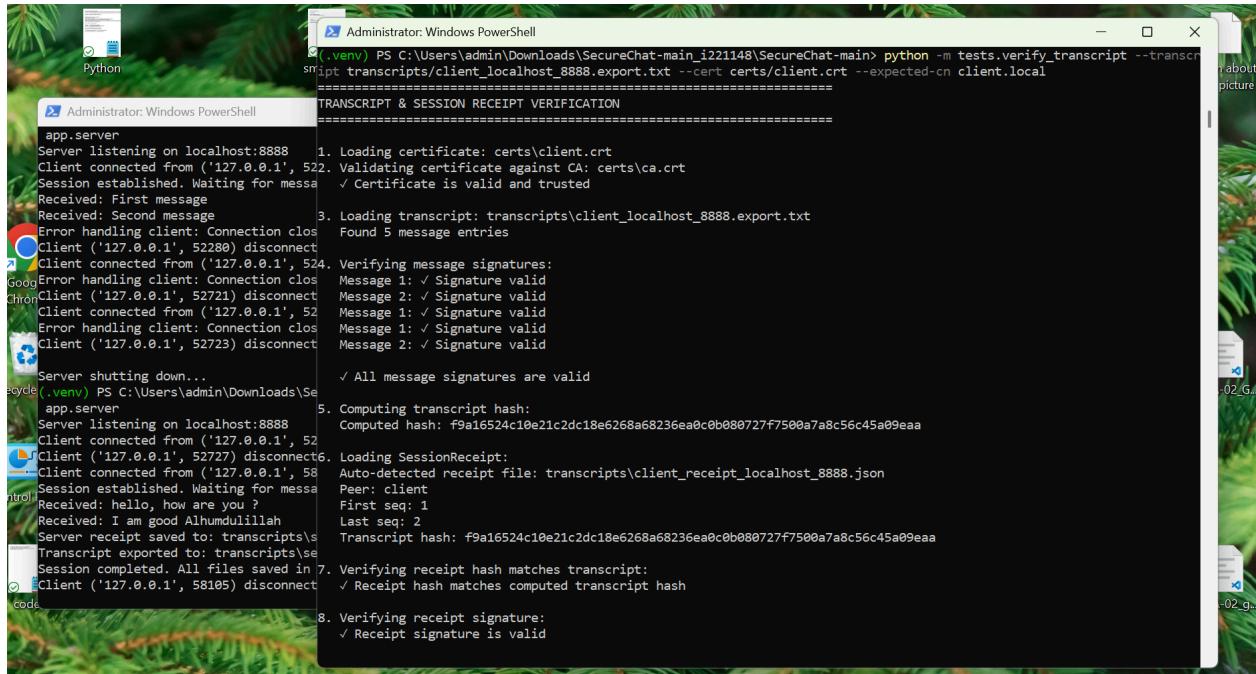
7. Verifying receipt hash matches transcript:

X Receipt hash does NOT match computed transcript hash!

Receipt hash: abc123def456...

Computed hash: xyz789ghi012...

Evidence: Screenshots showing successful verification and tampering detection.



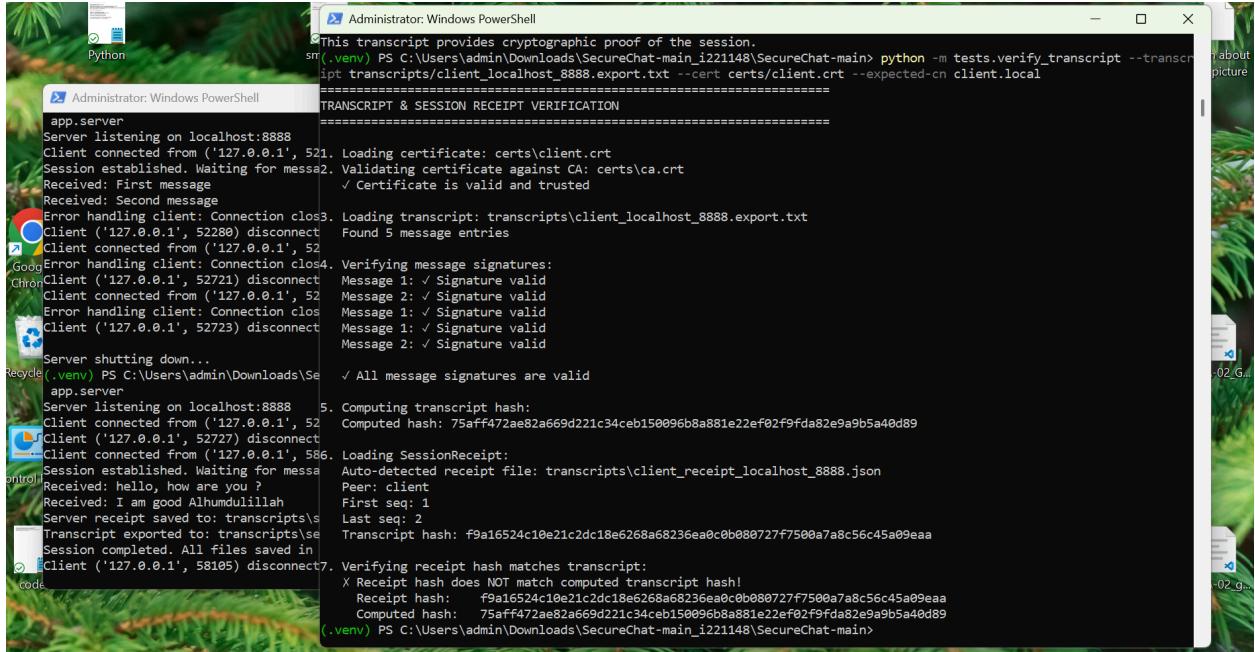
The screenshot shows a Windows PowerShell window titled "Administrator: Windows PowerShell" running under ".venv". The command executed is:

```
(.venv) PS C:\Users\admin\Downloads\SecureChat-main_i221148\SecureChat-main> python -m tests.verify_transcript --transcript transcripts/client_localhost_8888.export.txt --cert certs/client.crt --expected-cn client.local
```

The output displays the following steps and results:

1. Loading certificate: certs\client.crt
2. Validating certificate against CA: certs\ca.crt
 - ✓ Certificate is valid and trusted
3. Loading transcript: transcripts\client_localhost_8888.export.txt
 - Found 5 message entries
4. Verifying message signatures:
 - Message 1: ✓ Signature valid
 - Message 2: ✓ Signature valid
 - Message 1: ✓ Signature valid
 - Message 1: ✓ Signature valid
 - Message 2: ✓ Signature valid
5. All message signatures are valid
6. Computing transcript hash:
 - Computed hash: f9a16524c10e21c2dc18e6268a68236ea0c0b080727f7500a7a8c56c45a09eaa
7. Loading SessionReceipt:
 - Auto-detected receipt file: transcripts\client_receipt_localhost_8888.json
 - Peer: client
 - First seq: 1
 - Last seq: 2
8. Transcript hash: f9a16524c10e21c2dc18e6268a68236ea0c0b080727f7500a7a8c56c45a09eaa
9. Verifying receipt hash matches transcript:
 - ✓ Receipt hash matches computed transcript hash
10. Verifying receipt signature:
 - ✓ Receipt signature is valid

The PowerShell window also shows logs from the "app.server" application, which includes messages about clients connecting and disconnecting, receiving messages, and shutting down.



```
Administrator: Windows PowerShell
This transcript provides cryptographic proof of the session.
sm(.venv) PS C:\Users\admin\Downloads\SecureChat-main_i221148\SecureChat-main> python -m tests.verify_transcript --transcript transcripts\client_localhost_8888.export.txt --cert certs\client.crt --expected-cn client.local
=====
TRANSCRIPT & SESSION RECEIPT VERIFICATION
=====
app.server
Server listening on localhost:8888
Client connected from ('127.0.0.1', 5211). Loading certificate: certs\client.crt
Session established. Waiting for message. Validating certificate against CA: certs\ca.crt
Received: First message
Received: Second message
✓ Certificate is valid and trusted
Error handling client: Connection closed. Client ('127.0.0.1', 52280) disconnect
Client connected from ('127.0.0.1', 5212). Loading transcript: transcripts\client_localhost_8888.export.txt
Found 5 message entries
Error handling client: Connection closed. Client ('127.0.0.1', 52721) disconnect
Client connected from ('127.0.0.1', 52722). Loading transcript: transcripts\client_localhost_8888.export.txt
Message 1: ✓ Signature valid
Message 2: ✓ Signature valid
Error handling client: Connection closed. Client ('127.0.0.1', 52723) disconnect
Client ('127.0.0.1', 52724) disconnect
Client connected from ('127.0.0.1', 52725). Loading transcript: transcripts\client_localhost_8888.export.txt
Message 1: ✓ Signature valid
Message 2: ✓ Signature valid
Server shutting down...
Recycle (.venv) PS C:\Users\admin\Downloads\SecureChat-main_i221148\SecureChat-main> app.server
Server listening on localhost:8888
Client connected from ('127.0.0.1', 52726). Computing transcript hash: 75aff472ae82a669d221c34ceb150096b8a881e22ef02f9fd82e9a9b5a40d89
Client ('127.0.0.1', 52727) disconnect
Client connected from ('127.0.0.1', 52728). Loading SessionReceipt:
Session established. Waiting for message.
Received: hello, how are you ?
Received: I am good Alhumdulillah
Server receipt saved to: transcripts\session_receipt_localhost_8888.json
Transcript exported to: transcripts\session_receipt_localhost_8888.json
Session completed. All files saved in transcripts\session_receipt_localhost_8888.json
Client ('127.0.0.1', 58105) disconnect
code
(.venv) PS C:\Users\admin\Downloads\SecureChat-main_i221148\SecureChat-main>
```

Test 5: Wireshark Capture

Captures network traffic to verify encrypted payloads (no plaintext).

Step 1: Start Wireshark

1. Open Wireshark
2. Select loopback interface (Loopback: `lo0` on Mac, Npcap Loopback Adapter on Windows)
3. Start capture

Step 2: Run Chat Session

Terminal 1:

```
python -m app.server
```

Terminal 2:

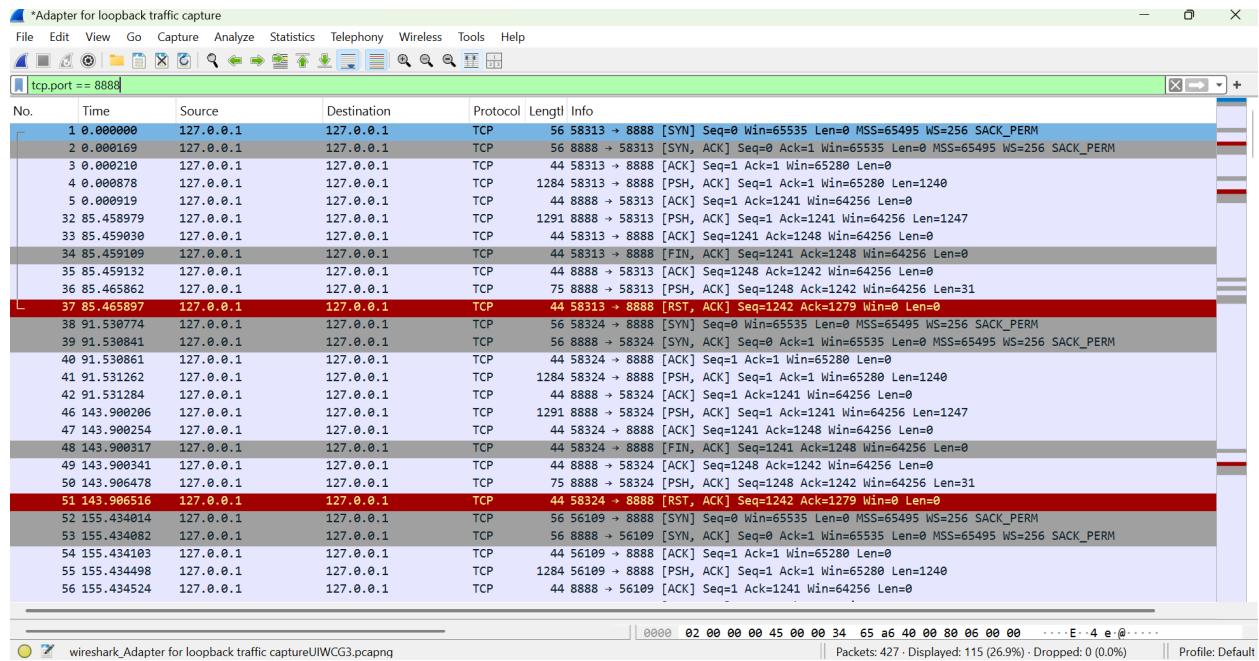
```
python -m app.client
```

- Login and send messages
- Type `quit` to end

Step 3: Stop Capture and Analyze

1. Stop Wireshark capture
2. Apply filter: `tcp.port == 8888`
3. Look for TCP packets with data
4. Expand packet → TCP → Data
5. Verify payload is encrypted (base64 encoded, no readable plaintext)

Evidence: Screenshot showing encrypted payload in Wireshark.



Problem: `BAD_CERT`: Certificate not signed by trusted CA

Solution:

1. Regenerate certificates:

```
python scripts/gen_ca.py --name "FAST-NU Root CA"
python scripts/gen_cert.py --cn server.local --out certs/server
```

2. python scripts/gen_cert.py --cn client.local --out certs/client