SecureChat - Assignment #2 Information Security (Fall 2025)

Name: Ali Ahmed

Roll No: 22i-1055

Section: G

Repository: https://github.com/AvidAli1/infoSec-A2

Table of Contents

1. Assignment Overview

This assignment involved designing and implementing a secure chat system using cryptographic primitives such as AES-128, RSA with X.509 certificates, Diffie-Hellman key exchange, and SHA-256 hashing. The system achieves Confidentiality, Integrity, Authenticity, and Non-Repudiation (CIANR) in a client-server architecture. I built a mini root CA, issued certificates, implemented registration and authentication, key agreement, and encrypted messaging. The system is a console-based secure chat application that demonstrates real-world cryptographic integration.

1.1 System Architecture

The system consists of a client and server communicating over TCP. The architecture includes: - **Control Plane:** Mutual authentication using X.509 certificates validated against a root CA. - **Authentication:** User registration and login with salted SHA-256 password hashing stored in MySQL. - **Key Agreement:** Diffie-Hellman (DH) exchange to derive a shared AES-128 key. - **Data Plane:** Encrypted messaging using AES-128-ECB with PKCS#7 padding, per-message RSA signatures for integrity, and an append-only signed transcript for non-repudiation.

The server listens for connections, handles multiple clients via threading, and stores session transcripts in JSON files.

1.2 Features Implemented

- Root CA generation and certificate issuance using cryptography library.
- Certificate validation (chain, expiry, CN/SAN).
- User registration/login with secure password storage in MySQL.
- DH key exchange with 2048-bit safe prime.
- AES-128 encryption/decryption.
- RSA SHA-256 signatures for messages and transcripts.
- Append-only transcript logging and hashing for non-repudiation.
- Pydantic models for message validation.
- Utility functions for timestamps, base64, and hashing.

2. Repository Structure

```
securechat-A2/
├─ README.md
├─ requirements.txt
├─ .gitignore
├─ certs/
│   ├─ MyRootCA_ca_cert.pem
│   ├─ MyRootCA_ca_key.pem
│   ├─ client.example.com_cert.pem
│   ├─ client.example.com_key.pem
│   ├─ myserver.example.com_cert.pem
│   ├─ myserver.example.com_key.pem
├─ transcripts/
│   └─ demo_session.json
├─ scripts/
│   ├─ gen_ca.py
│   ├─ gen_cert.py
│   ├─ diag_pki.py
│   ├─ cli_commands.txt
├─ scripts_ss/
├─ app/
│   ├─ .env
│   ├─ client.py
│   ├─ server.py
│   ├─ app_ss/
│   ├─ common/
│   ├─ protocol.py
│   ├─ utils.py
│   ├─ common_ss/
│   ├─ crypto/
│   │   ├─ aes.py
│   │   ├─ dh.py
│   │   ├─ pki.py
│   │   ├─ sign.py
│   ├─ crypto_ss/
│   ├─ storage/
│   │   ├─ db.py
│   │   ├─ transcript.py
│   ├─ testing_files/
├─ tests/
│   └─ manual/
└─ NOTES.md
```

3. Software Implementation Tables

3.1 app/client.py

Implements client workflow: hello exchange, registration/login, DH key agreement, encrypted chat, and receipt verification. Handles console input for user interactions.

3.2 app/server.py

Implements server workflow: listens for connections, handles hello exchange, authentication, DH, chat loop, transcript logging, and signed receipts. Uses threading for multiple clients.

3.3 app/crypto

3.3.1 crypto/aes.py

AES-128-ECB Helpers with PKCS#7:
- `pkcs7_pad(data: bytes) -> bytes` — Pad input bytes to 16-byte block with PKCS#7
- `pkcs7_unpad(data: bytes) -> bytes` — Remove PKCS#7 padding; raises ValueError if invalid
- `encrypt_aes(key: bytes, plaintext: bytes) -> bytes` — AES-128-ECB encrypt with PKCS#7 padding
- `decrypt_aes(key: bytes, ciphertext: bytes) -> bytes` — AES-128-ECB decrypt and remove PKCS#7 padding

Notes: key must be exactly 16 bytes (AES-128). encrypt_aes outputs raw bytes (can be base64-encoded for JSON transport). decrypt_aes returns the original plaintext bytes.

3.3.2 crypto/dh.py

Constants: - `P_2048` — 2048-bit MODP safe prime - `G = 2` — generator

Functions
- `generate_dh_pair()`
- `get_dh_public_bytes(public_key)`
- `compute_shared_secret(own_private_key, peer_public_value)`
- `derive_aes_key(shared_secret)`
- `client_dh_initiate()`
- `server_dh_respond(client_msg, server_private_key)`
- `client_dh_finalize(client_private_key, server_msg)`

3.3.3 crypto/pki.py

Functions:
- `load_certificate(pem_path)`
- `load_ca_certificate(ca_pem_path)`
- `verify_signature(ca_cert, leaf_cert)`
- `check_validity(cert)`
- `get_common_name(cert)`
- `get_san_dns_names(cert)`
- `verify_identity(cert, expected_hostname)`
- `is_ca_certificate(cert)`
- `validate_certificate(leaf_pem_path, ca_pem_path, expected_hostname)`
- `validate_server_certificate(server_cert_pem, ca_pem_path, expected_server_name)`
- `validate_client_certificate(client_cert_pem, ca_pem_path, expected_client_name)`
- `extract_public_key_from_cert(cert_pem)`

3.3.4 crypto/sign.py

Functions:
- `sign_data(private_key_pem, data)` — Sign data using RSA PKCS#1 v1.5 SHA-256
- `verify_signature(public_key_pem, data, signature)` — Verify RSA PKCS#1 v1.5 SHA-256 signature

3.4 app/common

3.4.1 common/protocol.py

Pydantic Models:
- `BaseMessage(type: str, timestamp: int)`
- `Hello(client_cert: str)`
- `Register(username: str, password_hash: str)`
- `DHClient(p: str, g: str, A: str)`
- `DHServer(B: str)` - `Message(sender: str, ciphertext: str)`
- `Receipt(identity: str, transcript_hash: str, signature: str)`

3.4.2 common/utils.py

Helper Functions:
- `now_ms()`
- `b64e(b: bytes)`
- `b64d(s: str)`
- `sha256_hex(data: bytes)`

3.5 scripts/

3.5.1 scripts/gen_ca.py

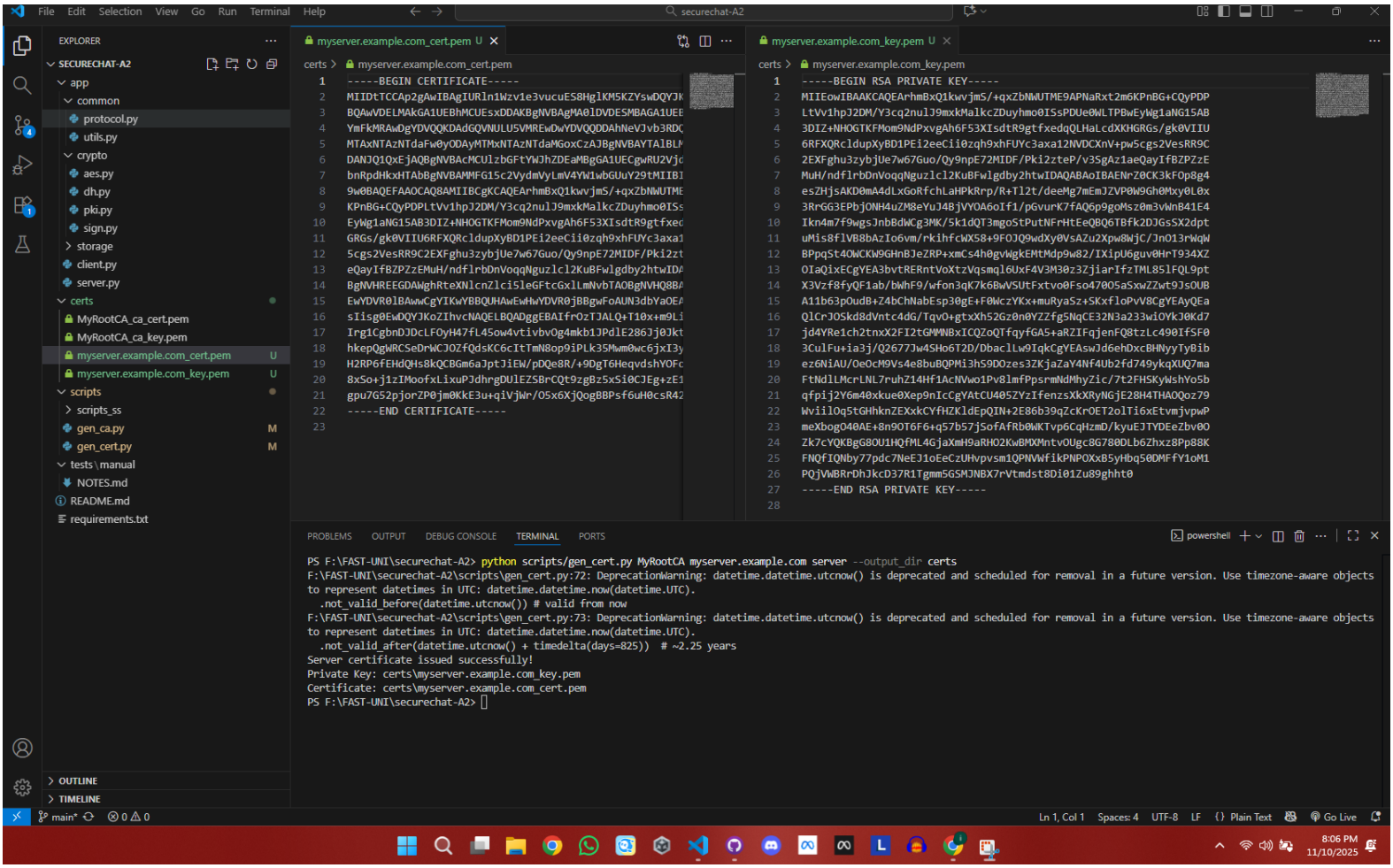
Root CA Creation:
- `create_root_ca(ca_name, output_dir='certs')`

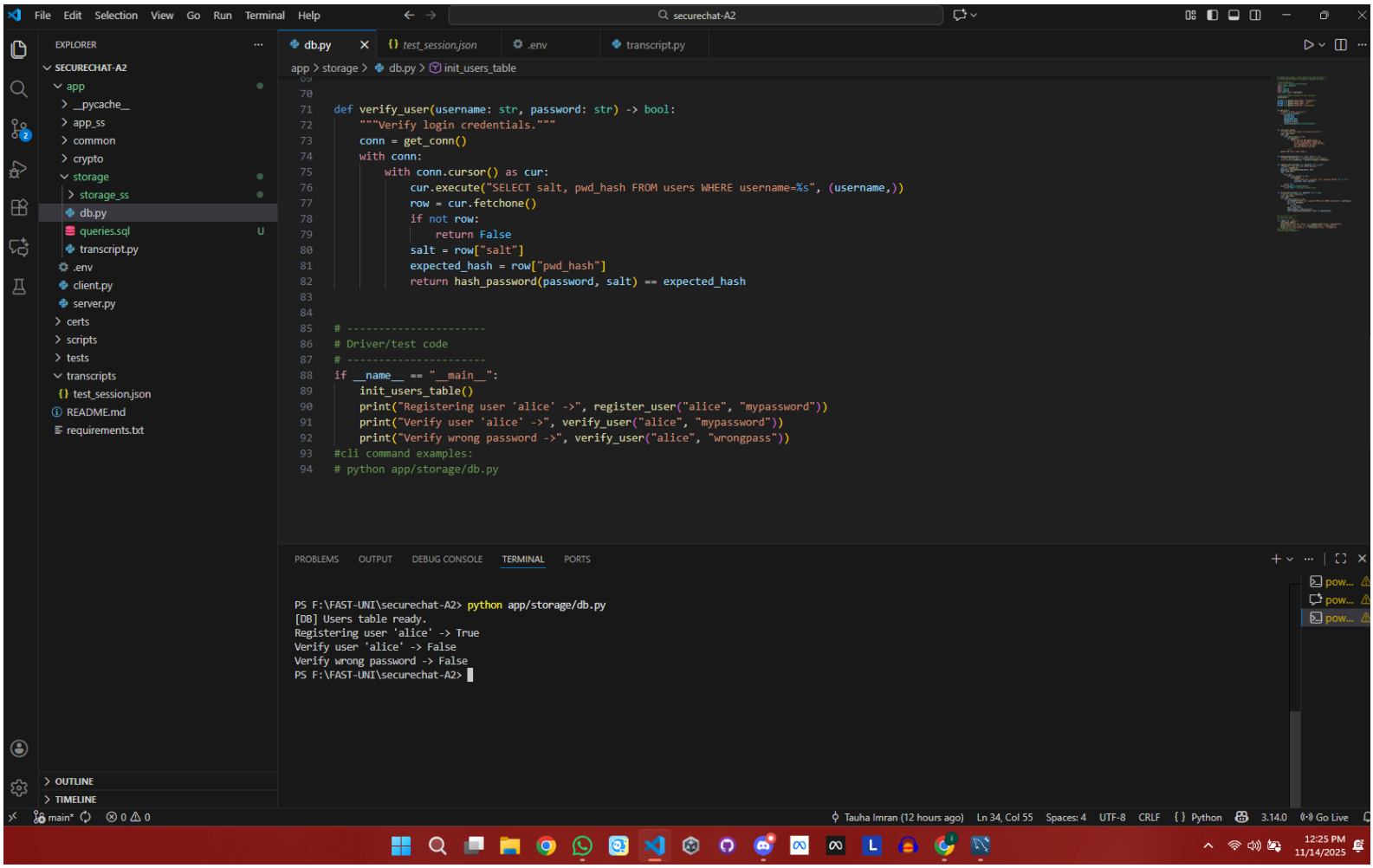
3.5.2 scripts/gen_cert.py

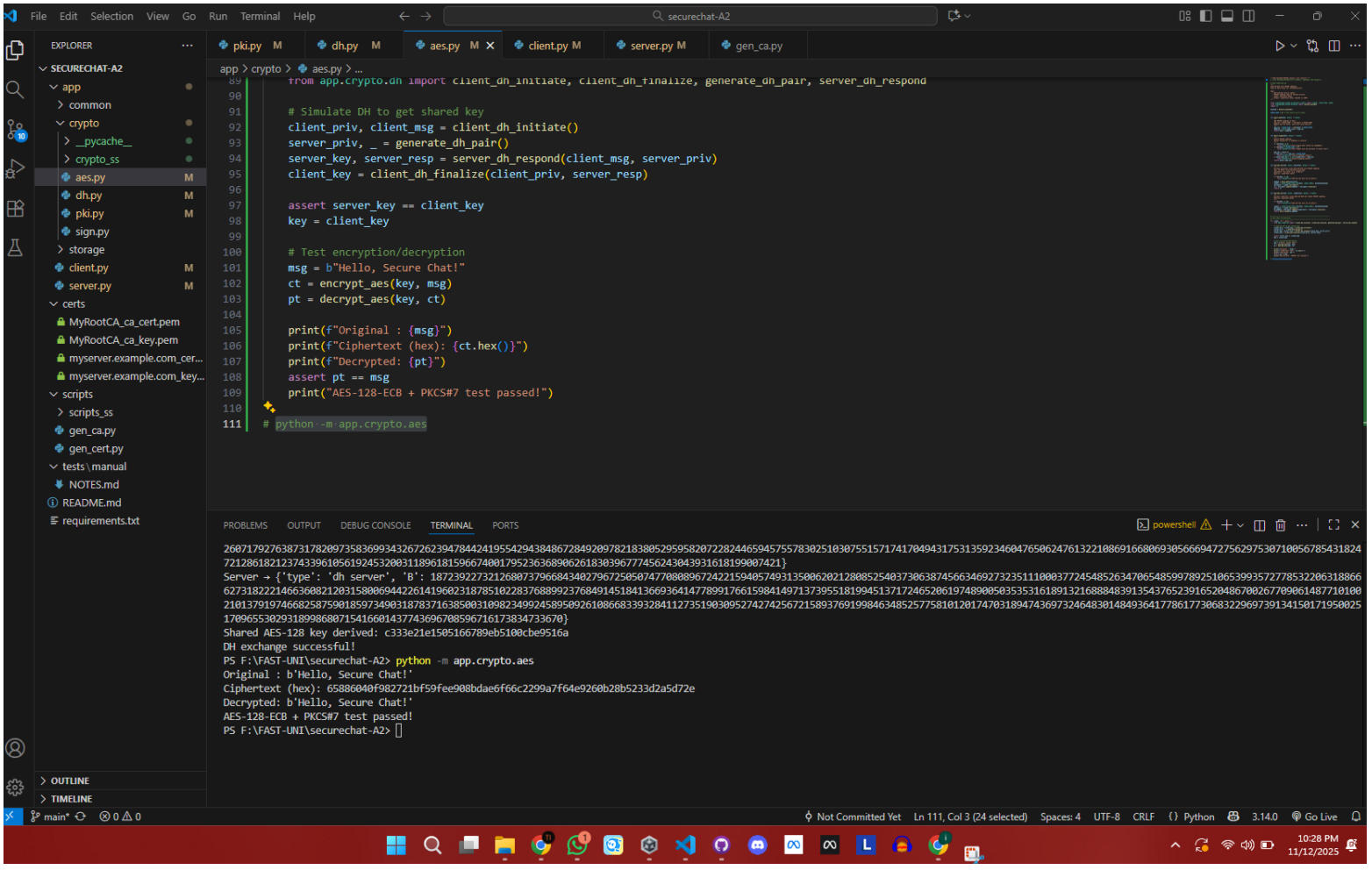
Issue Server/Client Certificate Signed by Root CA:
- `issue_certificate(ca_name, cert_name, cert_type, output_dir='certs')`

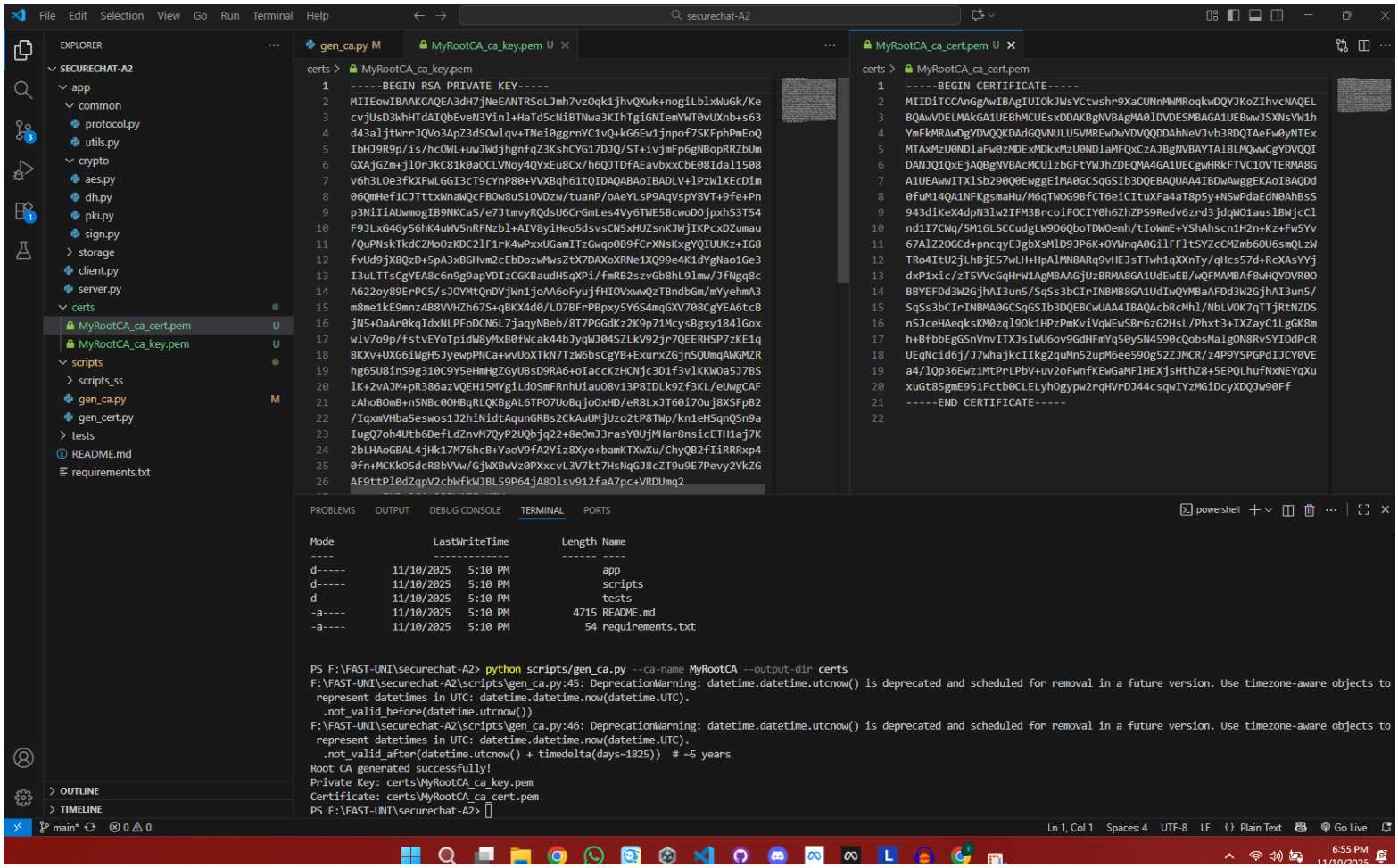
4. Evidence

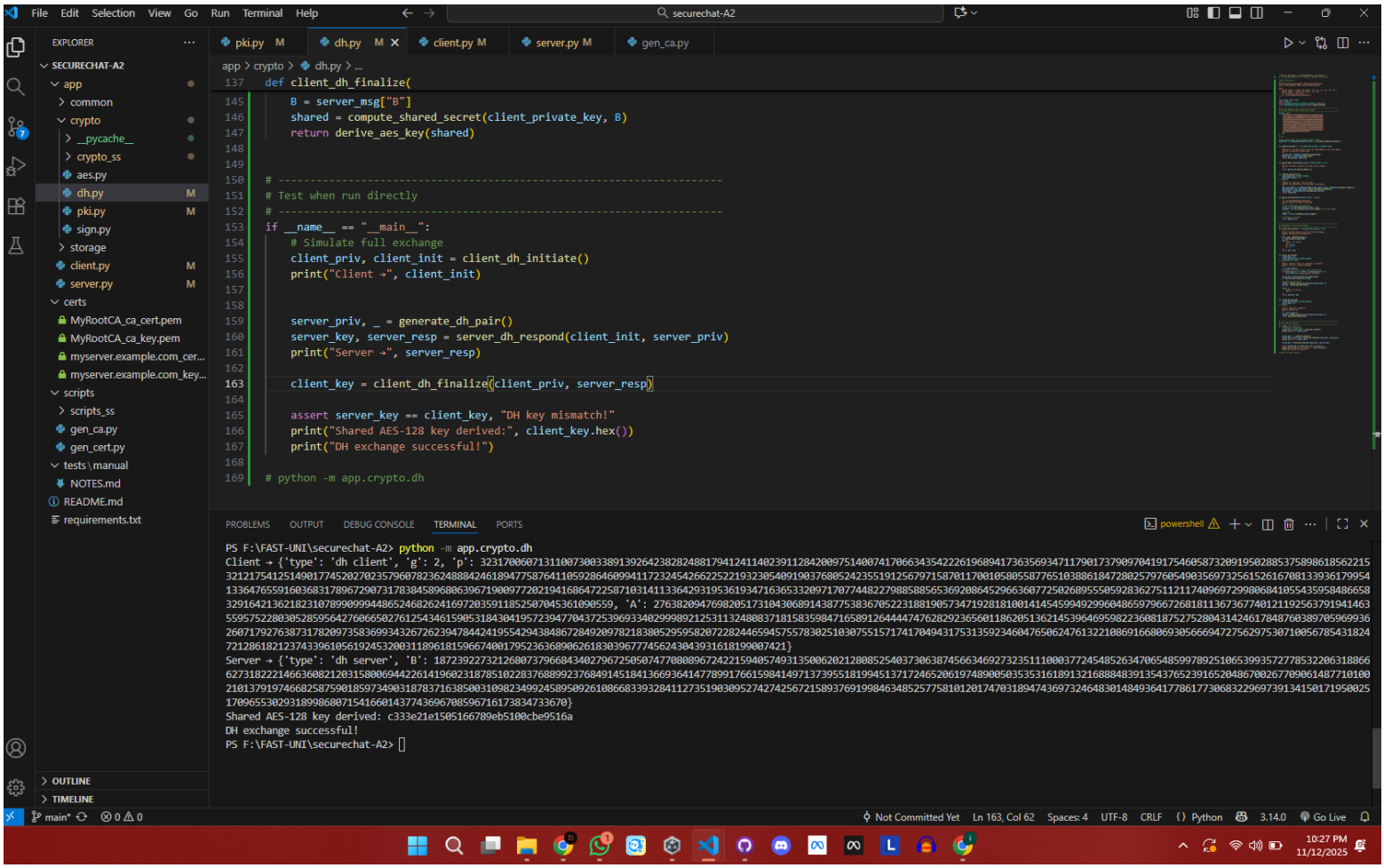
4.1 Code Execution Screenshots

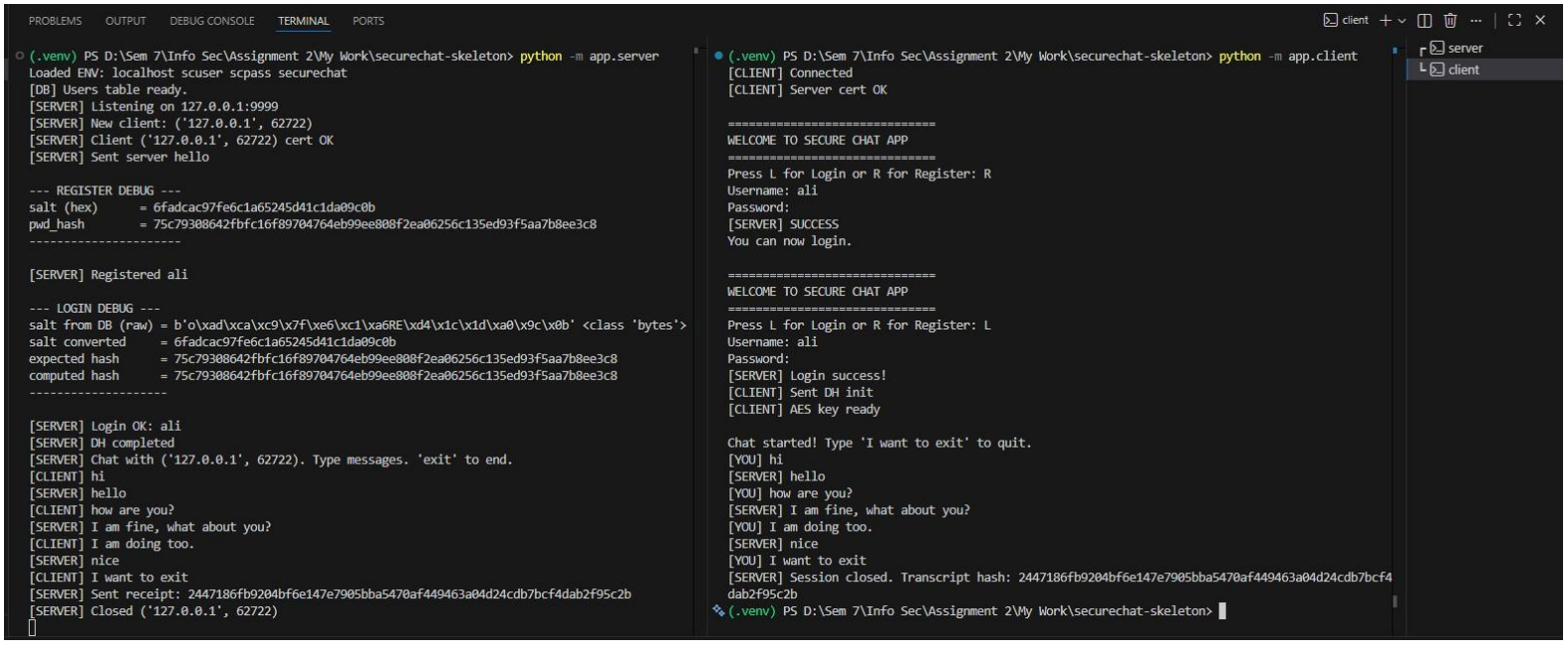



File Edit Selection View Go Run Terminal Help
securechat-A2
EXPLORER
SECURECHAT-A2
app
__pycache__
app_ss
common
crypto
storage
storage_ss
db.py
queries.sql
transcript.py
.env
client.py
server.py
certs
scripts
tests
transcripts
test_session.json
README.md
requirements.txt
OUTLINE
TIMELINE
db.py
test_session.json
.env
transcript.py
app > storage > db.py > init_users_table
def verify_user(username: str, password: str) -> bool:
"""Verify login credentials."""
conn = get_conn()
with conn:
with conn.cursor() as cur:
cur.execute("SELECT salt, pwd_hash FROM users WHERE username=%s", (username,))
row = cur.fetchone()
if not row:
return False
salt = row["salt"]
expected_hash = row["pwd_hash"]
return hash_password(password, salt) == expected_hash
# ----------------------
# Driver/test code
# ----------------------
if __name__ == "__main__":
init_users_table()
print("Registering user 'alice' ->", register_user("alice", "mypassword"))
print("Verify user 'alice' ->", verify_user("alice", "mypassword"))
print("Verify wrong password ->", verify_user("alice", "wrongpass"))
#cli command examples:
python app/storage/db.py
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS F:\FAST-UNI\securechat-A2> python app/storage/db.py
[DB] Users table ready.
Registering user 'alice' -> True
Verify user 'alice' -> False
Verify wrong password -> False
PS F:\FAST-UNI\securechat-A2>
main*
Tauha Imran (12 hours ago)
Ln 34, Col 55
Spaces: 4
UTF-8
CRLF
Python
3.14.0
Go Live
12:25 PM
11/14/2025

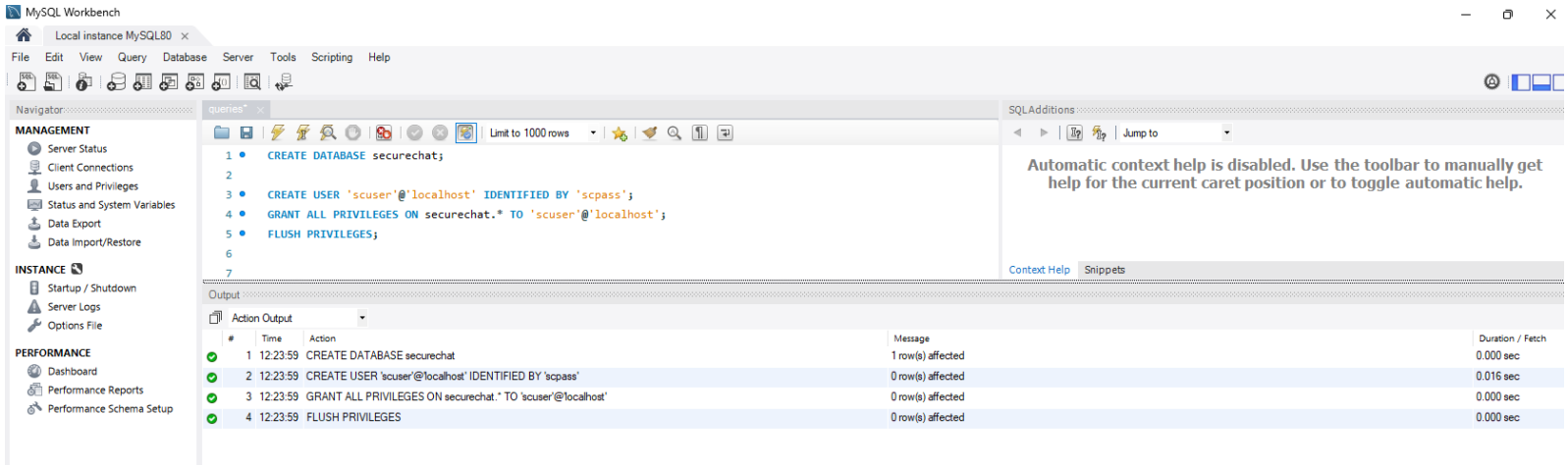

File Edit Selection View Go Run Terminal Help
securechat-A2
EXPLORER
SECURECHAT-A2
app
common
crypto
__pycache__
crypto_ss
aes.py M
dh.py M
pki.py M
sign.py
storage
client.py M
server.py M
certs
MyRootCA_ca_cert.pem
MyRootCA_ca_key.pem
myserver.example.com_cer...
myserver.example.com_key...
scripts
scripts_ss
gen_ca.py
gen_cert.py
tests\manual
NOTES.md
README.md
requirements.txt
pki.py M
dh.py M
aes.py M
client.py M
server.py M
gen_ca.py
app > crypto > aes.py > ...
from app.crypto.dh import client_dh_initiate, client_dh_finalize, generate_dh_pair, server_dh_respond
# Simulate DH to get shared key
client_priv, client_msg = client_dh_initiate()
server_priv, _ = generate_dh_pair()
server_key, server_resp = server_dh_respond(client_msg, server_priv)
client_key = client_dh_finalize(client_priv, server_resp)
assert server_key == client_key
key = client_key
# Test encryption/decryption
msg = b"Hello, Secure Chat!"
ct = encrypt_aes(key, msg)
pt = decrypt_aes(key, ct)
print(f"Original : {msg}")
print(f"Ciphertext (hex): {ct.hex()}")
print(f"Decrypted: {pt}")
assert pt == msg
print("AES-128-ECB + PKCS#7 test passed!")
# python -m app.crypto.aes
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
Server → {'type': 'dh server', 'B': 18723922732126807379668434027967250587477080896724221594057493135006202128085254037306387456634692732351110003772454852634706548599789251065399357277853220631886662731822214663608212031580086944226141960231878510228376889923768491451841366936414778991766159841497137395518199451371724652061974890050353531618913216888483913543765239165204867002677090614877101002101379197466825875901859734903187837163850031098234992458950926108668339328411273519030952742742567215893769199846348525775810120174703189474369732464830148493641778617730683229697391341501719500251709655302931899868071541660143774369670859671617383473367 0}
Shared AES-128 key derived: c333e21e1505166789eb5100cbe9516a
DH exchange successful!
PS F:\FAST-UNI\securechat-A2> python -m app.crypto.aes
Original : b'Hello, Secure Chat!'
Ciphertext (hex): 65886040f982721bf59fee908bdae6f66c2299a7f64e9260b28b5233d2a5d72e
Decrypted: b'Hello, Secure Chat!'
AES-128-ECB + PKCS#7 test passed!
PS F:\FAST-UNI\securechat-A2>
OUTLINE
TIMELINE
main*
Not Committed Yet
Ln 111, Col 3 (24 selected)
Spaces: 4
UTF-8
CRLF
Python
3.14.0
Go Live
10:28 PM
11/12/2025


File Edit Selection View Go Run Terminal Help
securechat-A2
EXPLORER
SECURECHAT-A2
app
common
protocol.py
utils.py
crypto
aes.py
dh.py
pki.py
sign.py
storage
client.py
server.py
certs
MyRootCA_ca_cert.pem U
MyRootCA_ca_key.pem U
scripts
scripts_ss
gen_ca.py M
gen_cert.py
tests
README.md
requirements.txt
OUTLINE
TIMELINE
gen_ca.py M
MyRootCA_ca_key.pem U
certs > MyRootCA_ca_key.pem
1 -----BEGIN RSA PRIVATE KEY-----
2 MIIEowIBAAKCAQEA3dH7jNeEANTRSoLJmh7vzOqk1jhvQXwk+nogiLblxWuGk/Ke
3 cvjUsD3WhHTdAIQbEveN3Yinl+HaTd5cNiBTNwa3KIhTgiGNIemYWT0vUXnb+s63
4 d43aljtWrrJQVo3ApZ3dSOwlqv+TNei0ggrnYC1vQ+kG6Ew1jnpof7SKFphPmEoQ
5 IbHJ9R9p/is/hcOWL+uwJWdjhgnfqZ3KshCYG17DJQ/ST+ivjmFp6gNBopRRZbUm
6 GXAjGZm+jlOrJkC81k0aOCLVNoy4QYxEu8Cx/h6QJTDfAEavbxxCbE08Idal1508
7 v6h3LOe3fkXFwLGGI3cT9cYnP80+VVXBqh61tQIDAQABAoIBADLV+lPzWlXEcDim
8 06QmHef1CJTttxWnaWQcFBOw8uS1OVDzw/tuanP/oAeYLsP9AqVspY8VT+9fe+Pn
9 p3NiIiAUwmogIB9NKCaS/e7JtmvyRQdsU6CrGmLes4Vy6TWE5BcwoDOjpxhS3T54
10 F9JLxG4Gy56hK4uWV5nRFNzbl+AIV8yiHeo5dsvsCN5xHUZsnKJWjIKPcxDZumau
11 /QuPNskTkdCZMoOzKDC2lF1rK4wPxxUGamITzGwqo0B9fCrXNsKxgYQIUUKz+IG8
12 fvUd9jX8QzD+5pA3xBGHvm2cEbDozwMwsZtX7DAXoXRNe1XQ99e4K1dYgNao1Ge3
13 I3uLTTsCgYEA8c6n9g9apYDIzCGKBaudH5qXPi/fmRB2szvGb8hL9lmw/JfNgq8c
14 A622oy89ErPC5/sJOYMtQnDYjWn1joAA6oFyujfHIOVxwwQzTBndbGm/mYyehmA3
15 m8me1kE9mnz4B8VVHZh67S+qBKX4d0/LD7BFrPBpxySY6S4mqGXV708CgYEA6tcB
16 jN5+OaAr0kqIdxNLPFoDCN6L7jaqyNBeb/8T7PGGdKz2K9p71McysBgxy184lGox
17 wlv7o9p/fstvEYoTpidW8yMxB0fWcak44bJyqWJ04SZLkV92jr7QEERHSP7zKE1q
18 BKXv+UXG6iWgH5JyewpPNCa+wvUoXTkN7TzW6bsCgYB+ExurxZGjn5QUmqAWGMZR
19 hg65U8inS9g310C9Y5eHmHgZGyUBsD9RA6+oIaccKzHCNjc3D1f3vlKKWOa5J7BS
20 lK+2vAJM+pR386azVQEH15MYgildOSmFRnhUiauO8v13P8IDLk9Zf3KL/eUwgCAF
21 zAhoBOmB+n5NBc0OHBqRlQKBgAL6TPO7UoBqjoOxHD/eR8LxJT60i7Ouj8XSFpB2
22 /IqxmVHba5eswos1J2hiNidtAqunGRBs2CkAuUMjUzo2tP8TWp/kn1eHSqnQSn9a
23 IugQ7oh4Utb6DefLdZnvM7QyP2UQbjq22+8eOmJ3rasY0UjMHar8nsicETH1aj7K
24 2bLHAoGBAL4jHk17M76hcB+YaoV9fA2Yiz8Xyo+bamKTXwXu/ChyQB2fIiRRRxp4
25 0fn+MCKkO5dcR8bVVw/GjWXBwVz0PXxcvL3V7kt7HsNqGJ8cZT9u9E7Pevy2YkZG
26 AF9ttPl0dZqpV2cbWfkWJBL59P64jA8Olsv912faA7pc+VRDUmq2
MyRootCA_ca_cert.pem U
certs > MyRootCA_ca_cert.pem
1 -----BEGIN CERTIFICATE-----
2 MIIDiTCCAnGgAwIBAgIUIOkJWsYCtwshr9XaCUNnMWMRoqkwDQYJKoZIhvcNAQEL
3 BQAwVDELMAkGA1UEBhMCUEsxDDAKBgNVBAgMA0lDVDESMBAGA1UEBwwJSXNsYW1h
4 YmFkMRAwDgYDVQQKDAdGQVNULU5VMREwDwYDVQQDDAhNeVJvb3RDQTAeFw0yNTEx
5 MTAxMzU0NDlaFw0zMDExMDkxMzU0NDlaMFQxCzAJBgNVBAYTAlBLMQwwCgYDVQQI
6 DANJQ1QxEjAQBgNVBAcMCUlzbGFtYWJhZDEQMA4GA1UECgwHRkFTVC1OVTERMA8G
7 A1UEAwwITXlSb290Q0EwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDd
8 0fuM14QA1NFKgsmaHu/M6qTWOG9BfCT6eiCItuXFa4aT8p5y+NSwPdaEdN0AhBsS
9 943diKeX4dpN3lw2IFM3BrcoiFOCIY0h6ZhZP59Redv6zrd3jdqWO1auslBWjcCl
10 nd1I7CWq/5M16LSCCudgLW9D6QboTDWOemh/tIoWmE+YShAhscn1H2n+Kz+Fw5Yv
11 67AlZ2OGCd+pncqyEJgbXsMlD9JP6K+OYWnqA0GilFFltSYZcCMZmb6OU6smQLzW
12 TRo4ItU2jLhBjES7wLH+HpAlMN8ARq9vHEJsTTwh1qXXnTy/qHcs57d+RcXAsYYj
13 dxP1xic/zT5VVcGqHrW1AgMBAAGjUzBRMA8GA1UdEwEB/wQFMAMBAf8wHQYDVR0O
14 BBYEFDd3W2GjhAI3un5/SqSs3bCIrINBMB8GA1UdIwQYMBaAFDd3W2GjhAI3un5/
15 SqSs3bCIrINBMA0GCSqGSIb3DQEBCwUAA4IBAQAcbRcMhl/NbLVOK7qTTjRtNZDS
16 nSJceHAeqksKM0zql9Ok1HPzPmKviVqWEwSBr6zG2HsL/Phxt3+IXZayC1LgGK8m
17 h+BfbbEgGSnVnvITXJsIwU6ov9GdHFmYq50y5N4590cQobsMalgON8RvSYIOdPcR
18 UEqNcid6j/J7whajkcIIkg2quMn52upM6ee59Og52ZJMCR/z4P9Y5PGPdIJCY0VE
19 a4/lQp36Ewz1MtPrLPbV+uv2oFwnfKEwGaMFlHEXjsHthZ8+5EPQLhufNxNEYqXu
20 xuGt85gmE951Fctb0CLELyhOgypw2rqHVrDJ44csqwIYzMGiDcyXDQJw90Ff
21 -----END CERTIFICATE-----
22
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
Mode LastWriteTime Length Name
d----- 11/10/2025 5:10 PM app
d----- 11/10/2025 5:10 PM scripts
d----- 11/10/2025 5:10 PM tests
-a---- 11/10/2025 5:10 PM 4715 README.md
-a---- 11/10/2025 5:10 PM 54 requirements.txt
PS F:\FAST-UNI\securechat-A2> python scripts/gen_ca.py --ca-name MyRootCA --output-dir certs
F:\FAST-UNI\securechat-A2\scripts\gen_ca.py:45: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timezone-aware objects to represent datetimes in UTC: datetime.datetime.now(datetime.UTC).
.not_valid_before(datetime.utcnow())
F:\FAST-UNI\securechat-A2\scripts\gen_ca.py:46: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timezone-aware objects to represent datetimes in UTC: datetime.datetime.now(datetime.UTC).
.not_valid_after(datetime.utcnow() + timedelta(days=1825)) # ~5 years
Root CA generated successfully!
Private Key: certs\MyRootCA_ca_key.pem
Certificate: certs\MyRootCA_ca_cert.pem
PS F:\FAST-UNI\securechat-A2>
main* 0 0 0
Ln 1, Col 1 Spaces: 4 UTF-8 LF Plain Text Go Live
6:55 PM 11/10/2025


File Edit Selection View Go Run Terminal Help
securechat-A2
EXPLORER
SECURECHAT-A2
app
common
crypto
__pycache__
crypto_ss
aes.py
dh.py M
pki.py M
sign.py
storage
client.py M
server.py M
certs
MyRootCA_ca_cert.pem
MyRootCA_ca_key.pem
myserver.example.com_cer...
myserver.example.com_key...
scripts
scripts_ss
gen_ca.py
gen_cert.py
tests\manual
NOTES.md
README.md
requirements.txt
OUTLINE
TIMELINE
pki.py M
dh.py M
client.py M
server.py M
gen_ca.py
app > crypto > dh.py > ...
def client_dh_finalize(
B = server_msg["B"]
shared = compute_shared_secret(client_private_key, B)
return derive_aes_key(shared)
# Test when run directly
if __name__ == "__main__":
# Simulate full exchange
client_priv, client_init = client_dh_initiate()
print("Client →", client_init)
server_priv, _ = generate_dh_pair()
server_key, server_resp = server_dh_respond(client_init, server_priv)
print("Server →", server_resp)
client_key = client_dh_finalize(client_priv, server_resp)
assert server_key == client_key, "DH key mismatch!"
print("Shared AES-128 key derived:", client_key.hex())
print("DH exchange successful!")
# python -m app.crypto.dh
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
PS F:\FAST-UNI\securechat-A2> python -m app.crypto.dh
Client → {'type': 'dh client', 'g': 2, 'p': 32317006071311007300338913926423828248817941241140239112842009751400741706634354222619689417363569347117901737909704191754605873209195028853758986185622153212175412514901774520270235796078236248884246189477587641105928646099411723245426622522193230540919037680524235519125679715870117001058055877651038861847280257976054903569732561526167081339361799541336476559160368317896729073178384589680639671900977202194168647225871031411336429319536193471636533209717077448227988588565369208645296636077250826895550592836275112174006972998068410554359548686583291642136218231078990999448652648262416972035911852507045361090559, 'A': 27638209476982051731043068914387753836705223188190573471928181001414545994929960486597966726818113673677401211925637919414635595752280305285956427606650276125434615905318430419572394770437253969334029998921253113248883718158359847165891264444747628292365601186205136214539646959822360818752752804314246178487603897059699362607179276387317820973583699343267262394784424195542943848672849209782183805295958207228244659457557830251030755157174170494317531359234604765062476132210869168806938566694725629753071005678543182472128618212374339610561924532003118961815966740017952363689062618303967774562430439316181990074217}
Server → {'type': 'dh server', 'B': 18723922732126807379668434027967250507477080896724221594057493135006202128085254037306387456634692732351110003772454852634706548599789251065399357277853220631886662731822214663608212031580069442261419602318785102283768899237684914518413669364147789917661598414971373955181994513717246526019748900503535316189132168884839135437652391652048670026770906148771010021013791974668258759018597349031878371638500310982349924589509261086683393284112735190389527427425672158937691998463485257758101201747031894743697324648301484936417786177306832296973913415017195002517096553029318998680715416601437743696708596716173834733670}
Shared AES-128 key derived: c333e21e1505166780eb5100cbe9516a
DH exchange successful!
PS F:\FAST-UNI\securechat-A2>
main*
Not Committed Yet
Ln 163, Col 62
Spaces: 4
UTF-8
CRLF
Python
3.14.0
Go Live
10:27 PM
11/12/2025


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
(.venv) PS D:\Sem 7\Info Sec\Assignment 2\My Work\securechat-skeleton> python -m app.server
Loaded ENV: localhost scuser scpass securechat
[DB] Users table ready.
[SERVER] Listening on 127.0.0.1:9999
[SERVER] New client: ('127.0.0.1', 62722)
[SERVER] Client ('127.0.0.1', 62722) cert OK
[SERVER] Sent server hello
--- REGISTER DEBUG ---
salt (hex) = 6fadcac97fe6c1a65245d41c1da09c0b
pwd_hash = 75c79308642fbfc16f89704764eb99ee808f2ea06256c135ed93f5aa7b8ee3c8
[SERVER] Registered ali
--- LOGIN DEBUG ---
salt from DB (raw) = b'o\xad\xca\xc9\x7f\xe6\xc1\xa6RE\xd4\x1c\x1d\xa0\x9c\x0b' <class 'bytes'>
salt converted = 6fadcac97fe6c1a65245d41c1da09c0b
expected hash = 75c79308642fbfc16f89704764eb99ee808f2ea06256c135ed93f5aa7b8ee3c8
computed hash = 75c79308642fbfc16f89704764eb99ee808f2ea06256c135ed93f5aa7b8ee3c8
[SERVER] Login OK: ali
[SERVER] DH completed
[SERVER] Chat with ('127.0.0.1', 62722). Type messages. 'exit' to end.
[CLIENT] hi
[SERVER] hello
[CLIENT] how are you?
[SERVER] I am fine, what about you?
[CLIENT] I am doing too.
[SERVER] nice
[CLIENT] I want to exit
[SERVER] Sent receipt: 2447186fb9204bf6e147e7905bba5470af449463a04d24cdb7bcf4dab2f95c2b
[SERVER] Closed ('127.0.0.1', 62722)
client
(.venv) PS D:\Sem 7\Info Sec\Assignment 2\My Work\securechat-skeleton> python -m app.client
[CLIENT] Connected
[CLIENT] Server cert OK
==========
WELCOME TO SECURE CHAT APP
==========
Press L for Login or R for Register: R
Username: ali
Password:
[SERVER] SUCCESS
You can now login.
==========
WELCOME TO SECURE CHAT APP
==========
Press L for Login or R for Register: L
Username: ali
Password:
[SERVER] Login success!
[CLIENT] Sent DH init
[CLIENT] AES key ready
Chat started! Type 'I want to exit' to quit.
[YOU] hi
[SERVER] hello
[YOU] how are you?
[SERVER] I am fine, what about you?
[YOU] I am doing too.
[SERVER] nice
[YOU] I want to exit
[SERVER] Session closed. Transcript hash: 2447186fb9204bf6e147e7905bba5470af449463a04d24cdb7bcf4dab2f95c2b
(.venv) PS D:\Sem 7\Info Sec\Assignment 2\My Work\securechat-skeleton>
server
client

4.2 MySQL Database Screenshots



5. Implementation Details

- Root CA and Certificates: gen_ca.py and gen_cert.py for CA and server/client certs; pki.py validates.
- DH Key Exchange: dh.py implements 2048-bit DH.
- AES Encryption: aes.py handles AES-128-ECB with PKCS#7.
- RSA Signing: sign.py ensures integrity and non-repudiation.
- Protocol Models: protocol.py defines Pydantic models.
- Utilities: utils.py provides timestamps, base64, hashing.
- Database: db.py handles MySQL users table with salted SHA-256.
- Transcript: transcript.py logs messages, computes SHA-256 hash.
- Client/Server: client.py handles UI, server.py manages connections and chat.

6. Challenges and Solutions

- Certificate Validation: chain verification, expiry, hostname matching.
- Key Derivation: Trunc16(SHA256(Ks)) for AES key.
- Signature Extraction: extract_public_key_from_cert added.
- Pydantic Validation: DH params converted to str.
- Error Handling: try/except blocks for network and validation errors.

7. Conclusion

This assignment demonstrated practical cryptography in a secure chat system. The implementation meets all CIANR requirements.