SecureChat - Test Report

Assignment #2 - Information Security

Student Name: [Your Name]
Roll Number: [Your Roll Number]
Date: [Submission Date]
GitHub Repository: [Your Fork URL]

Executive Summary

This test report documents the comprehensive security testing performed on the SecureChat system. All tests verify that the system successfully achieves Confidentiality, Integrity, Authenticity, and Non-Repudiation (CIANR) as required.

Test Environment

- **Operating System:** Windows 11
- **Python Version:** 3.11.x
- **MySQL Version:** 8.0
- **Testing Tools:** Wireshark 4.x, Python scripts

Test Cases

Test 1: Certificate Validation

Objective: Verify that invalid certificates are properly rejected

Test 1.1: Expired Certificate

Steps:

1. Run `python scripts\test_attacks.py`
2. Observe Test 1 output

Expected Result:

BAD_CERT: Certificate expired
✓ TEST PASSED: Expired certificate correctly rejected

Actual Result:

[Paste screenshot or output here]

Status: ✅ PASSED / ❌ FAILED

Test 1.2: Self-Signed Certificate

Steps:

1. Observe Test 2 output from test_attacks.py

Expected Result:

BAD_CERT: Invalid signature
✓ TEST PASSED: Self-signed certificate correctly rejected

Actual Result:

[Paste screenshot or output here]

Status: ✅ PASSED / ❌ FAILED

Evidence:

Show Image

Test 2: Encrypted Communication (Wireshark)

Objective: Verify that all communication is encrypted and no plaintext is transmitted

Steps:

1. Start Wireshark with filter `tcp.port == 5000`
2. Start server: `python -m app.server`

3. Start client and perform login
4. Send messages: "Hello World", "This is secret"
5. Capture packets

Wireshark Display Filter Used:

```
tcp.stream eq 0
```

Observations:

- All message payloads are base64-encoded
- No plaintext messages visible in packet capture
- JSON structure visible but content encrypted

Expected: No plaintext "Hello World" or "This is secret" in any packet

Actual Result:

[Describe what you see]

Status: ✅ PASSED / ❌ FAILED

Evidence:

Show Image

Test 3: Message Tampering Detection

Objective: Verify that tampered messages are detected and rejected

Steps:

1. Run `python scripts\test_attacks.py`
2. Observe Test 3 output

Expected Result:

Original Signature Valid: True
Tampered Signature Valid: False
✓ TEST PASSED: Tampered message correctly detected (SIG_FAIL)

Actual Result:

[Paste output here]

Status: ✅ PASSED / ❌ FAILED

Evidence:

Show Image

Test 4: Replay Attack Protection

Objective: Verify that replayed messages are detected and blocked

Steps:

1. Run `python scripts\test_attacks.py`
2. Observe Test 4 output

Expected Result:

Received: seqno=2, content='REPLAYED: Second message'
⚠️ REPLAY DETECTED! seqno 2 <= last 2
✗ Message REJECTED

Actual Result:

[Paste output here]

Additional Test - Live Replay:

1. Capture a message JSON from network traffic
2. Resend the same message

3. Observe server/client rejection

Server Output:

```
[SERVER] REPLAY detected! Rejecting message.
```

Status: ✅ PASSED / ❌ FAILED

Evidence:

Show Image

Test 5: Transcript Integrity

Objective: Verify that transcript modifications are detected

Steps:

1. Run `python scripts\test_attacks.py`
2. Observe Test 5 output

Expected Result:

```
Original transcript hash: [hash1]
Tampered transcript hash: [hash2]
✓ TEST PASSED: Transcript tampering detected via hash mismatch
```

Actual Result:

```
[Paste output here]
```

Status: ✅ PASSED / ❌ FAILED

Test 6: Non-Repudiation Verification

Objective: Verify session receipts can be validated offline

Steps:

1. Complete a chat session with at least 3 messages
2. Locate generated files:
   - `transcripts/client_receipt_[user]_[timestamp].json`
   - `transcripts/client_[user]_[timestamp].txt`
3. Run verification:

```
cmd

python scripts\verify_receipt.py --receipt [receipt_file] --transcript [transcript_file] --cert certs\client_cert.pem --verify-messa
```

Expected Result:

```
✓ Transcript hash MATCHES receipt
✓ Signature VALID
VERIFICATION SUCCESS: Receipt is authentic and transcript is intact

Message 1 (seq=1): ✓ VALID
Message 2 (seq=2): ✓ VALID
Message 3 (seq=3): ✓ VALID
```

Actual Result:

```
[Paste output here]
```

Status: ✅ PASSED / ❌ FAILED

Evidence:

Show Image

Test 7: Authentication Security

Objective: Verify secure credential handling

Test 7.1: Registration

Steps:

1. Register new user with credentials
2. Inspect database:

```
cmd

python scripts\export_db.py --info
```

Verification:

- Password is NOT stored in plaintext
- Salt is 16 bytes random
- pwd_hash is 64 characters (SHA-256 hex)

Database Sample:

```
email: test@example.com
username: testuser
salt: [16 random bytes shown in hex]
pwd_hash: a3f8d2e... (64 hex chars)
```

Status: ✅ PASSED / ❌ FAILED

Test 7.2: Login with Wrong Password

Steps:

1. Attempt login with incorrect password
2. Observe rejection

Expected:

```
[CLIENT] Login failed: Invalid credentials
```

Actual:

```
[Paste output here]
```

Status: ✅ PASSED / ❌ FAILED

Security Properties Verification

Property	Test	Result
Confidentiality	Wireshark capture shows only encrypted data	✅
Integrity	Message tampering detected via signature	✅
Authenticity	Certificate validation enforced	✅
Non-Repudiation	Session receipts verifiable offline	✅
Replay Protection	Sequence numbers enforced	✅
Credential Security	Salted hashing, encrypted transit	✅

Additional Tests

Database Export

Command:

```
cmd

python scripts\export_db.py --output database_export.sql
```

Result:

- Schema exported successfully
- Sample data included (with obfuscated passwords)
- File size: [X] KB

Status: ✅ PASSED / ❌ FAILED

Test Summary

Total Tests: 10
Passed: [X]
Failed: [X]
Success Rate: [X]%

Known Issues

[List any issues encountered during testing]

1. Issue 1: [Description]
    - **Severity:** Low/Medium/High
    - **Workaround:** [If applicable]

Conclusion

The SecureChat system successfully demonstrates all required security properties:

- ✅ PKI-based authentication with certificate validation
- ✅ End-to-end encryption using AES-128
- ✅ Message integrity via RSA signatures
- ✅ Replay attack prevention
- ✅ Non-repudiation through signed transcripts

All test cases passed, confirming that the system achieves CIANR as specified in the assignment requirements.

Appendix: Screenshots

A1. Wireshark Packet Capture

[Insert screenshot showing encrypted payloads]

A2. Certificate Validation Errors

[Insert screenshots of BAD_CERT rejections]

A3. Message Tampering Detection

[Insert output of tamper test]

A4. Replay Attack Prevention

[Insert server logs showing REPLAY rejection]

A5. Receipt Verification

[Insert successful verification output]

A6. Database Schema

[Insert database structure screenshot]

Report Prepared By: [Your Name]
Date: [Date]