**Eastern Mediterranean University  
Computer Engineering Department**

**Semester:**Spring 2025

**Course: CMPE455/CMSE456 Security of Computer Systems and Networks**

**Instructor: Alexander G. Chefranov**

**Project: Whatsapp Like Application**

**Lab Assistants: Ali Farrokhnejad, Morteza Farrokhnejad**

**Group 6**

**Kaan Sulkalar - 20000063**

**İlayda Yılmaz - 20000068**

**Atilla Çelik - 20331325**

**Eylül Topçu – 22000207**

# Outline

1. Problem Definition

2. WALA Detailed Description

3. Tools and Security Methods Used

4. WALA Implementation

4.1 System Architecture

4.2 Database Structure

4.3 Implementation of Actors: Sysadmin and Client

4.4 Data Structures Used

4.5 Algorithms Used

4.6 Code Developed

5. Tests Conducted and Their Results

6. Conclusion

7. References

8. Appendices

9. Screenshots

# 1. Problem Definition

# 2. WALA Detailed Description

WALA is a secure, encrypted messaging system implemented using a combination of Python and C#.   
It supports encrypted communication between clients using DES, with RSA for secure key exchange.   
Each message is digitally signed to ensure integrity. The server provides logging and an admin dashboard.

# 3. Tools and Security Methods Used

- Programming Languages: Python, C#  
- Encryption Algorithms: DES (message), RSA (key exchange)  
- Hash/Signature: RSA-based digital signatures  
- Networking: TCP sockets  
- GUI: Windows Forms (C#)  
- Server Framework: Flask (Python)  
- Logging: Plain text file (chatlog.txt)

# 4. WALA Implementation

## 4.1 System Architecture

The system consists of:  
- A Python-based server running on one machine  
- A C# Windows Forms client on a separate machine  
- Secure TCP-based communication channel  
- RSA public/private key exchange mechanism  
- Flask-based Admin Dashboard

## 4.2 Database Structure

WALA currently uses a flat file system for logging messages (`chatlog.txt`). In future versions, an SQLite database or SQLCipher can be integrated for secure, structured storage of messages and session keys.

## 4.3 Implementation of Actors: Sysadmin and Client

- Client: Encrypts messages with DES, signs them, and sends over the network.  
- Server: Verifies signatures, decrypts and relays messages, logs them.  
- Sysadmin: Monitors status via Flask dashboard and can inspect logs.

## 4.4 Data Structures Used

- Python Dictionaries for session key management  
- String buffers for message handling  
- Lists for storing active users  
- RSA and DES key objects via PyCryptodome

## 4.5 Algorithms Used

- DES: Symmetric encryption/decryption of messages  
- RSA: Asymmetric encryption for session key exchange  
- SHA256 + RSA: For generating and verifying digital signatures

## 4.6 Code Developed

Python Files:  
- server.py  
- crypto\_utils.py  
- admin\_dashboard.py  
C# Files:  
- Form1.cs  
- Program.cs  
All source files are provided in the submitted .zip archive under Appendices.

# 5. Tests Conducted and Their Results

# 6. Conclusion

6. Conclusion

# 7. References

Appendix: Source Code Overview

1. Introduction

This report presents the design and implementation of a secure messaging system, referred to as WALA (WhatsApp-Like Application), developed as part of the CMPE455 course. The project emphasizes secure client-server communication using DES for symmetric encryption, RSA for session key exchange, and digital signatures for message integrity. The system includes a Python-based server and a C# client, simulating real-world secure messaging environments on separate machines.

2. Problem Definition

In an age of rapid digital communication, the need for confidentiality, integrity, and authenticity in messaging is crucial. The objective of this project is to simulate a secure communication platform where messages are encrypted, signed, and verified. Additionally, a sysadmin must be able to monitor system activity. The system should meet the following criteria:  
  
- Ensure end-to-end message encryption  
- Use digital signatures for integrity  
- Secure session key exchange  
- Log communication history  
- Admin visibility via web dashboard

3. Methodology

The project was implemented using a combination of Python (for server-side operations and cryptography) and C# (for the client application). Communication was achieved using TCP sockets. The encryption algorithm used for messages was DES, and RSA was utilized for secure exchange of the DES session key. The Flask framework was used to provide a simple admin dashboard for sysadmin functionalities. Message integrity was preserved using digital signatures based on RSA.

4. System Design and Implementation

The system is composed of the following components:  
  
- Server (`server.py`): Handles RSA key generation, session management, message logging, and relaying between clients.  
  
- Client (C#): Connects to the server, receives the public key, encrypts the DES key, and exchanges secure messages.  
  
- `crypto\_utils.py`: Implements DES encryption/decryption, RSA key handling, and digital signature generation/verification.  
  
- Admin Dashboard (`admin\_dashboard.py`): A Flask-based static dashboard used to monitor server status.  
  
- Logging: All messages are stored in `chatlog.txt` for audit and recovery purposes.  
  
Each connected client receives a dedicated session key and must sign messages for integrity validation. The server verifies these signatures before broadcasting to other clients.

5. Testing and Results

The following tests were conducted to validate the system:  
  
- Session Key Exchange: Verified encrypted session key transfer and correct decryption.  
  
- DES Encryption Test: Confirmed correct encryption/decryption of messages between clients.  
  
- Digital Signature: Tampered messages were correctly rejected by the server.  
  
- Multiple Clients: Successfully tested messaging with 2 clients on separate machines.  
  
- Admin Dashboard: Flask interface loaded successfully and indicated active status.  
  
- Log Validation: Messages stored in `chatlog.txt` were verified for correctness.  
  
All test cases passed, demonstrating a fully functional secure messaging prototype.

6. Conclusion

The WALA project successfully demonstrates the implementation of secure communications using symmetric and asymmetric cryptographic principles. It provides insight into encryption, authentication, and network programming. While the prototype uses DES for simplicity, the structure supports upgrade to more modern algorithms like AES. The modular design enables future enhancements including group chat, real-time admin analytics, and database-backed storage.

7. References

1. PyCryptodome Documentation: https://pycryptodome.readthedocs.io/en/latest/  
2. DES Encryption Standard: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.46-3.pdf  
3. RSA Algorithm Overview: https://www.rsa.com/en-us/company/rsa-cryptography  
4. Flask Framework: https://flask.palletsprojects.com/en/2.3.x/

Appendix: Source Code Overview

- `server.py`: TCP server, RSA key exchange, message handling  
- `client.py` (C#): Secure chat interface, DES message encryption  
- `crypto\_utils.py`: DES, RSA, signature handling  
- `admin\_dashboard.py`: Flask dashboard for sysadmin monitoring  
- `chatlog.txt`: Stored encrypted message logs

# 8. Appendices

Appendix 1: server.py  
Appendix 2: crypto\_utils.py  
Appendix 3: admin\_dashboard.py  
Appendix 4: Form1.cs  
Appendix 5: Program.cs

9. ScreenShots

metin, yazılım, bilgisayar simgesi, web sayfası içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, yazılım, bilgisayar simgesi, web sayfası içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, yazılım, web sayfası, bilgisayar simgesi içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, ekran görüntüsü, yazılım, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturuldu

ekran görüntüsü, yazılım, metin, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu