



Hacettepe University
Computer Engineering Department

BBM465 Information Security Lab.- 2024 Fall

Assignment 4

January 05, 2025

2220356141- Ayşe Yaren Topgün

2210356008- Sûdenaz Yazıcı

Introduction

In this project, we built a simplified Hybrid Kerberos System (HKS) to provide secure authentication and communication. We use RSA for key exchange and AES for encrypting messages. The Key Distribution Center (KDC) handles registration, authentication, and ticket creation for both clients and servers. Our main goal is to show a ticket-based protocol that allows secure communication while protecting passwords and session keys.

Assumptions and Simplifications

1. We store all data in Dataset.csv. The file contains both client and server details on one line (Client ID, password, public key, private key, Server ID, etc.). Also if a client wants to communicate with another server, it should register with that server.
2. We send the session key in plaintext from the client to the server to keep the design simple. In a real Kerberos system, we would protect the session key more carefully.
3. We have a separate TicketGrant class acting like a “Ticket Grand Server” (TGS). The KDC calls TicketGrant to create and encrypt the session key.
4. We only store user passwords in clear text for demonstration. In a real system, we would store hashed passwords.

CSV

Each line of Dataset.csv looks like this:

Client, <clientId>, <password>, <clientPublicKey>, <clientPrivateKey>, Server,<serverId>, <serverPublicKey>, <serverPrivateKey>

We generate RSA key pairs for both the client and the server. Then we store them as Base64 strings in the CSV file.

Graphical User Interface (GUI)

We created a JavaFX GUI. It has text fields for Client ID, Password, Server ID, and a Message. It has three buttons:

1. Register (for new client–server registration)
2. LogIn (client authentication with KDC)
3. Communicate with Server (secure messaging)

A log area on the interface displays actions (e.g., “Client registered successfully.”).

Key Distribution Center (KDC)

The KDC does these tasks:

1. Registration: It generates RSA public–private keys for a client and a server. It writes them into Dataset.csv.
2. Authentication: It checks the client’s password. If correct, it creates a session key (AES) and calls **TicketGrant** to encrypt that key with the client’s and server’s RSA public keys.
3. Ticket issuance: The KDC sends the ticket (with the encrypted session key) to both the client and the server. The ticket has a 5-minute validity.

TicketGrant Class

We introduced inheritance by making the TicketGrant class a subclass of the KDC. In real Kerberos, the TGS (Ticket Grant Server) is part of the KDC, but it acts like a separate module. By using inheritance, TicketGrant can use some KDC methods (like `getPublicKey(...)`) and produce tickets with session keys. TGS works inside or under the KDC’s authority.

KDC and TicketGrant

Registration: The KDC creates RSA keys for client and server and writes them to Dataset.csv.

Login: The KDC checks the client’s password. If correct, it calls the TicketGrant subclass to generate a session key (AES). TicketGrant then encrypts that key with both the client’s and server’s RSA public keys, producing two tickets. The KDC sends these tickets to the client and server.

TicketGrant: By inheriting from KDC, TicketGrant can reuse some CSV helper methods. It focuses on creating the session key and encrypting it. This separates “ticket creation” from other KDC work.

Client and Server Interaction

The client decrypts its ticket using its RSA private key to get the session key (AES).

The server decrypts its own ticket using its RSA private key to get the same session key.

We send the session key in plaintext from the client to the server for simplicity. The server compares that key with its own key. If they match, secure communication begins.

We use AES/CBC for message encryption. Each time we encrypt a message, we create a new IV (Initialization Vector). The IV is combined with the ciphertext, so the receiver can decrypt it.

Logging System

We have a Log.txt file. It records every step, such as registration events, login attempts, ticket creation, and message sending. For example, it shows lines like:

```
[REGISTER BUTTON CLICKED]
Client client1 with password 1234567 requesting registration with server server1.

[KDC REGISTER]
Client client1 and Server server1 keys created.

[LOGIN BUTTON CLICKED]
Client client1 with password 1234567 is trying to authenticate with KDC for server server1.

[TICKETGRANT: CREATE TICKETS]
sessionKey (Base64): 34Mv/PMYuM0l2PBMdnvKtw==
clientTicket: Ticket{clientId='client1', serverId='server1', encryptedSessionKey='YauP1p9WV8Fxn2gNsU3Kwfr53bDZHPfuUc9zI2
serverTicket: Ticket{clientId='client1', serverId='server1', encryptedSessionKey='Uv0Lc4ZbdCPg9h8x+93aJbA3hmeP/RszessSYV

[Client encryptMessage]
Message: message1
IV (Base64): Ch2ZtLAGrXv44ZbSIjL+hQ==
Encrypted (Base64): Ch2ZtLAGrXv44ZbSIjL+hSF0i6XYWVZD3iBqYaMb57I=

[Server communicateWithClient]
Client's plaintext session key (Base64): 34Mv/PMYuM0l2PBMdnvKtw==
[Server] Session keys matched. Decrypted message: message1

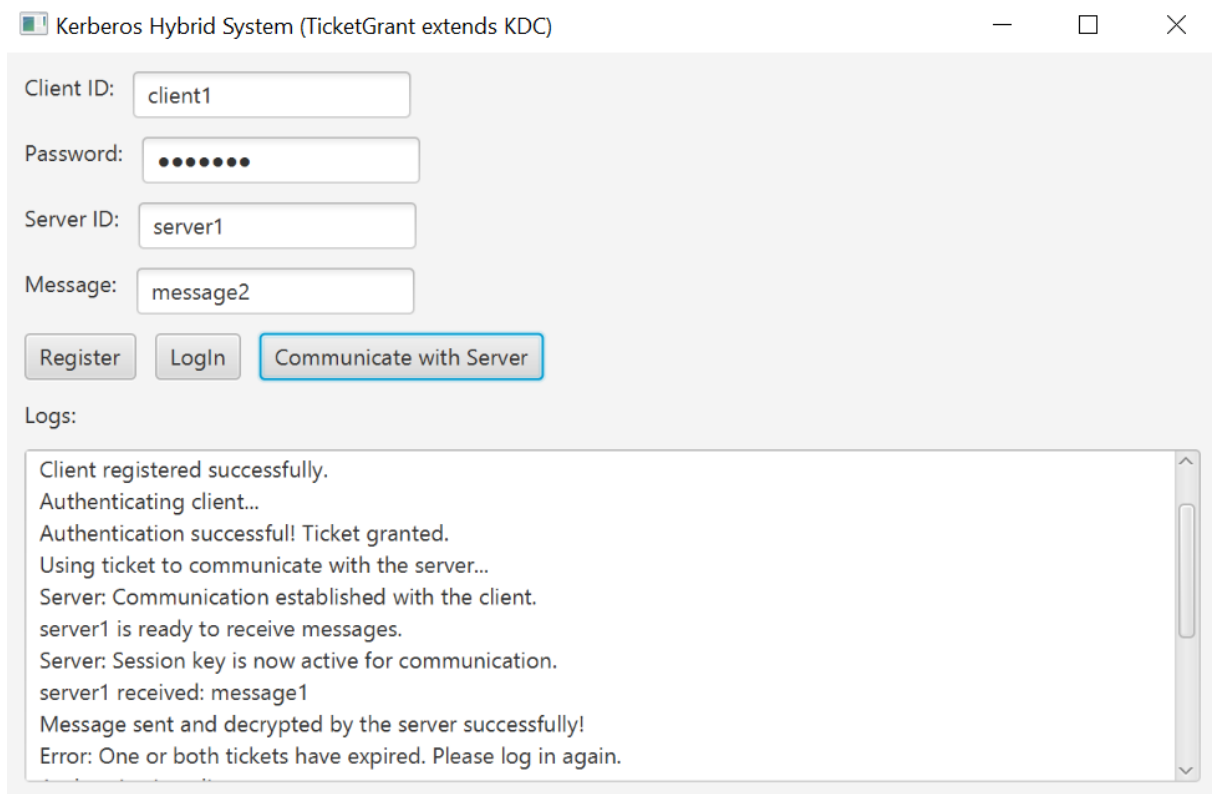
[LOGIN BUTTON CLICKED]
Client client1 with password 1234567 is trying to authenticate with KDC for server server1.

[TICKETGRANT: CREATE TICKETS]
sessionKey (Base64): jBAfDGAKJB2JxkB28xsgcQ==
clientTicket: Ticket{clientId='client1', serverId='server1', encryptedSessionKey='d+4798+0yRzJCXhH4sKAmRleY5fqK47wsjMBIY
serverTicket: Ticket{clientId='client1', serverId='server1', encryptedSessionKey='I09uvu2j4mt75TaD0vunDbhDJiYnBNHc3xIfsd

[Client encryptMessage]
Message: message2
IV (Base64): BV0C+qRE6KuKb0+0cLb8yw==
Encrypted (Base64): BV0C+qRE6KuKb0+0cLb8y+6X0a07ftZTL6j16Ukwp0=

[Server communicateWithClient]
Client's plaintext session key (Base64): jBAfDGAKJB2JxkB28xsgcQ==
[Server] Session keys matched. Decrypted message: message2
```

Also this is a example of GUI:



The screenshot shows a window titled "Kerberos Hybrid System (TicketGrant extends KDC)". It contains four input fields: "Client ID:" with the value "client1", "Password:" with masked characters, "Server ID:" with the value "server1", and "Message:" with the value "message2". Below these fields are three buttons: "Register", "Login", and "Communicate with Server". The "Communicate with Server" button is highlighted with a blue border. Below the buttons is a "Logs:" section with a scrollable text area containing the following log messages:

```
Client registered successfully.
Authenticating client...
Authentication successful! Ticket granted.
Using ticket to communicate with the server...
Server: Communication established with the client.
server1 is ready to receive messages.
Server: Session key is now active for communication.
server1 received: message1
Message sent and decrypted by the server successfully!
Error: One or both tickets have expired. Please log in again.
```

Conclusion

We have a simplified Hybrid Kerberos protocol. We store information in Dataset.csv and we let the KDC manage RSA keys for each client and server. TicketGrant class acts as our Ticket Grand Server, creating session keys. Because our project is simplified, we send the session key from client to server in plaintext. In a production system, we would protect it further. Despite these simplifications, we demonstrate the main idea of Kerberos: using tickets and a session key to secure communication between a client and a server.