Hacettepe University Computer Engineering Department

Information Security Lab.

Assignment 4

Subject: Hybrid Kerberos System
Due Date: 05.01.2025
Language: Java
T.A.s: Ali Baran TAŞDEMİR, Sibel KAPAN

In this assignment, you will implement a Hybrid Kerberos System (HKS), a network authentication protocol for secure authentication over an insecure network. HKS uses symmetric (AES) and asymmetric (RSA) encryption to protect sensitive data like passwords and session keys while ensuring efficient and secure communication.

The Key Distribution Center (KDC) manages key exchanges and issues tickets. The client registers with the KDC, authenticates, and receives a session key and ticket encrypted with the server's public key. The server decrypts the ticket using its private key, and both the client and server use the session key for secure communication. This system combines RSA's security for key exchange with AES's efficiency for data encryption, offering robust authentication and communication.

Key Features

Client and Server Registration: Clients and servers register with the Key Distribution Center (KDC), providing credentials (passwords) and public keys. The KDC stores their public-private key pairs and associates them with IDs for secure communication.

Client Authentication: Clients authenticate by sending their ID and password to the KDC. Upon verification, the KDC generates a session key, encrypts it with the server's public key, and sends it to both client and server in a secure ticket.

Ticket Granting: The KDC issues a ticket containing the session key, encrypted with the server's public key. This ticket acts as proof of authentication, enabling secure communication without exposing sensitive data.

Secure Communication: The client uses the session key from the ticket to encrypt data sent to the server. The server decrypts the session key with its private key, ensuring both parties can securely exchange messages.

Session Key Encryption: Session keys are encrypted with RSA, using client and server public keys. This ensures only the intended recipients can access the keys, enabling secure communication with AES for data encryption.

Key Management: The KDC manages all cryptographic keys, issuing and renewing session keys and tickets to maintain security. Keys are securely stored, and expired tickets are replaced to reduce risks.

Ticket Expiration and Renewal: Tickets have expiration times to limit misuse. Expired tickets require the client to request new ones, ensuring periodic key rotation for added security.

Security and Integrity: HKS combines RSA for key exchange and AES for data encryption, ensuring secure communication and protecting data from unauthorized access and tampering.

Details for HKS

Client: Clients need to register with the KDC by providing their ID and password.

1. The client sends a registration request to the KDC with its ID and password.
2. The KDC generates a public-private RSA key pair for the client.
3. The KDC stores the public/private keys along with the password in its database.
4. Store client details in a CSV file "dataset.txt" for persistence.

Server: The server also registers with the KDC by providing its ID.

1. The server sends its registration request to the KDC.
2. The KDC generates a public-private RSA key pair for the server.
3. The KDC stores the server's keys securely.

Client Authentication with KDC: The client needs to authenticate itself with the KDC to get a ticket for server communication.

1. The client sends a request to the KDC with its credentials (client ID and password).
2. The KDC checks if the password matches its records.
3. If authentication is successful, the KDC generates a session key (for AES encryption) and encrypts it.
    a. With the server's public key. (PubKeyClient(SessionKey))
    b. With the client's public key. (PubKeyServer(SessionKey))
4. The KDC issues a ticket containing the encrypted session key and sends it to the client and server..
5. Output: The client receives the ticket containing the encrypted session key, which can now be used to communicate securely with the server.

Secure Communication: Once authenticated, the client can use the ticket to securely communicate with the server.

1. The client sends the ticket to the server.
2. The server decrypts the session key from the ticket using its private key.
3. Both the client and the server now have the same session key, which they can use to encrypt and decrypt their communication securely.

Ticket Usage and Verification: The ticket serves as proof that the client has been authenticated and can now communicate with the server using the session key.

1. The server decrypts the session key from the ticket.
2. The server and client can now send encrypted messages to each other using the **AES session key**.
3. The ticket given by the KDC expires in **5 minutes**.

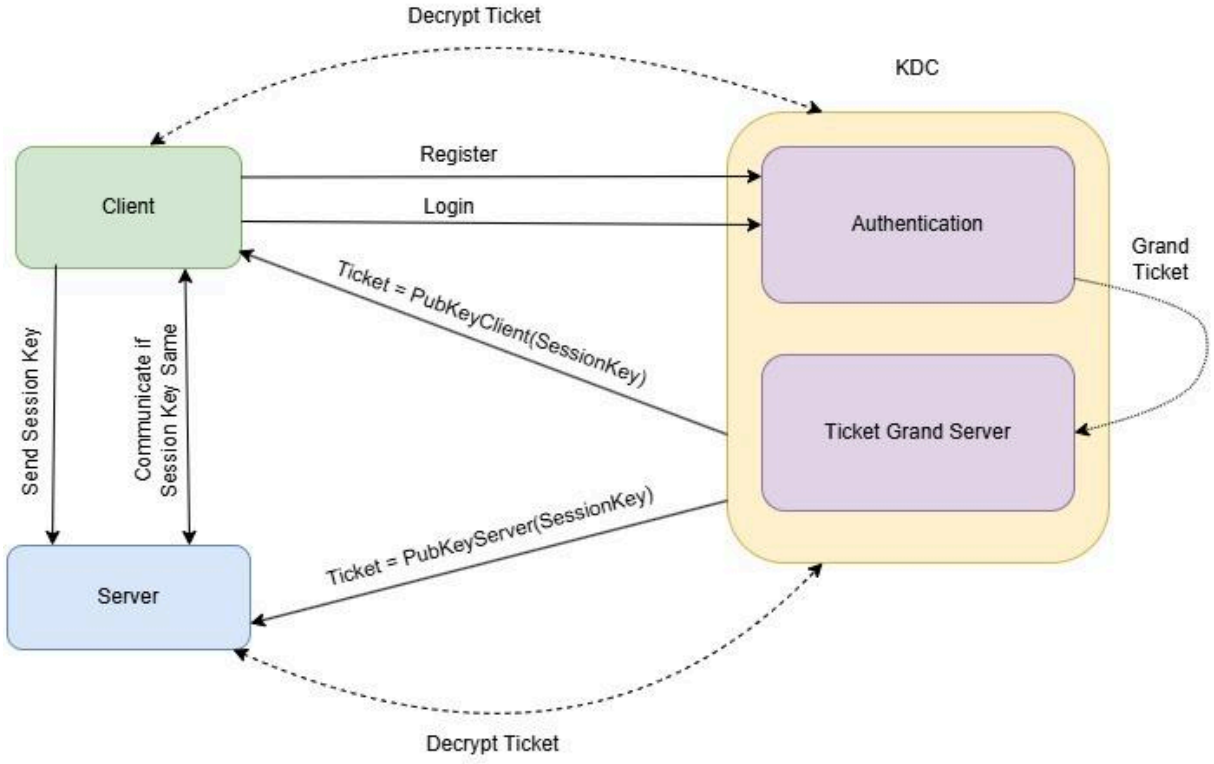

Figure 1: HKS system schema

In Figure 1, you can see steps for client, server, and KDC communications.

1. The client registers with KDC with a username, password, and server name (which the client connects to). KDC generates public-private keys for the client and server.
2. The client logs into KDC with its credentials, and the authentication center verifies the client and grants ticket permission.
3. Ticket Grand Server generates a ticket (Session Key generated with AES) and encrypts the ticket with the public keys of the client and server. And sent the encrypted ticket to the client and server.
4. The client and server decrypt the tickets with KDC to have the session key.
5. The client sends the session key to the server and the server checks that the session keys are the same and starts the communication.

Implementation Details and GUI

1. In this assignment, you will use "*javax.crypto*" for using AES and RSA algorithms.
2. When implementing your assignment, you should make sure that it is object-oriented. Client, Server, KDS, Ticket, and TicketGrant will be your classes. Also, your main file is responsible for GUI and proper function callings.
3. The **main** file uses buttons and text fields to register the client and server. And, log in to KDS (authentication) and communicate with the server.
4. The **client is** responsible for the construction of the client with a client ID and passport.
5. The **server** is used to construct a server and has two functions *communicateWithClient()*, and *decryptSessionKey()*.
6. The **ticket** is used to construct a ticket object for KDS and **ticketGrant** encrypts session keys for the client and server. The ticket expiration time is **5 minutes**.
7. The **KDS** will register the client and server, use RSA to generate the public-private key, and save that information to **"dataset.csv"** for further use. The KDS needs to authenticate the client on login and if the credentials are correct it will generate a session key (**TicketGrant**). The Generated session key will be used by the client to connect to the server.
8. Also, you need to provide a **"log.txt"** file showing every step when your application works. For example,

```
client1 with password1 sent to KDS for registration.
KDS registers client1 with password1 and generates public-private keys for the client.
KDS registers the server and generates public-private keys for the server.
private key of client1: …
public key of client1: …
…
…
…
```

GUI

You will create a user interface using JavaFX for HKS on this assignment. In the application, you will simply register, log in, and communicate with the server. You can find GUI examples in Figure 2.

GUI workflow:

1. Register a client with a client ID and password.
    - KDC generates private and public keys for the client with RSA. Save the client ID, password, and private-public keys to "dataset.csv" for further use.
2. Login with client ID and password.
    - The authentication server on KDS will check your credentials. If the credentials are correct Ticket Granting Server generates a client-to-server ticket.
3. Communicate with the server using generated client-to-server ticket.

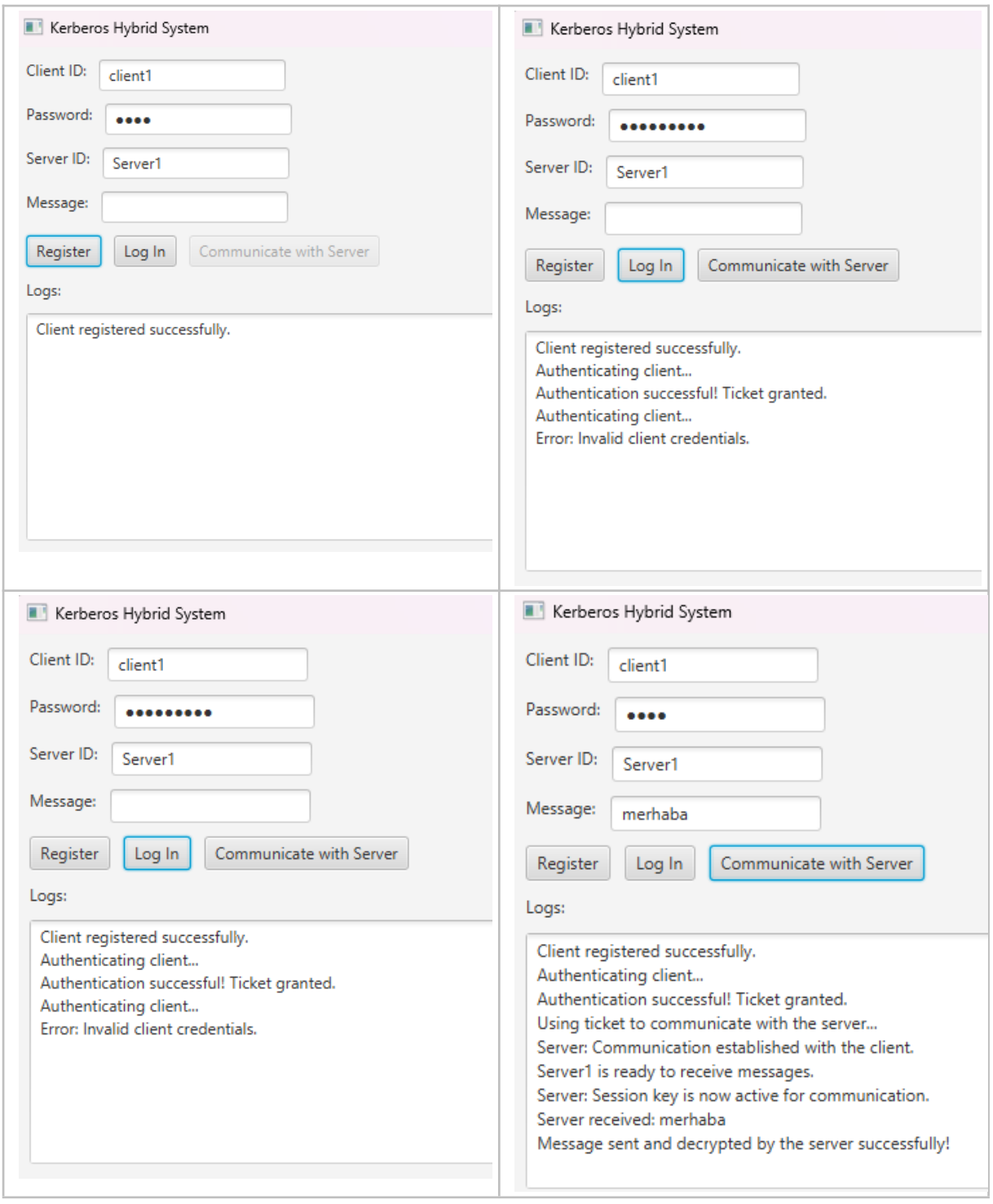

Figure 2: Screenshots of GUI.

Notes

1. You should prepare a report involving your approach and details of the implementation you have coded. You must write down the names and IDs of your teammates in the report in order to be evaluated correctly.
2. You must submit the homework in groups.
3. You should prepare a report that describes your approach to the problem with the details of your implementation. You must write down all group members' names and ids. Reports will be graded too.
4. You can ask your questions about the homework via Piazza. (www.piazza.com/hacettepe.edu.tr/fall2024/bbm465)

5. T.A. as himself has the right to partially change this document. However, the modifications will be announced in the Piazza system. In this case, it is your obligation to check the Piazza course page periodically.
6. You will submit your work via the submission system. (submit.cs.hacettepe.edu.tr) The submission format is given below:

→ src/
→Main.java
→Client.java
→Server.java
→KDC.java
→Ticket.java
→TicketGrant.java
→Dataset.csv
→Log.txt
→Report.pdf

Policy

All work on assignments must be done with your own group unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in pseudo-code) will not be tolerated. In short, turning in someone else's work (from the internet), in whole or in part, as your own will be considered a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.