CSE439 TERM PROJECT

By

Burak Eymen Çevik

Kerberos Design

Project Report

TABLE OF CONTENTS

DESIGN EXPLANATION

This project facilitates user authentication to access services from servers. Upon correct credentials verification, users receive a Ticket Granting Ticket (TGT) from the server. Using this TGT, a user can request a ticket from the server, enabling them to access the requested services.

Both client and server sides are designed to securely transmit login credentials using RSA symmetric key encryption. During user authentication, the server compares the provided credentials with data stored in an SQLite3 database. If there is a match, authentication is granted; otherwise, it is denied. Once authenticated, a user can request a TGT, and with this TGT, they can request a ticket from the server to access the requested services. The user retains access to the requested services until the ticket expires. These operations can be performed within a single server using FastAPI, SQLAlchemy, and the datetime library. Additionally, an authenticated user can request the current timestamp from the server and receive a response.

If a user has an admin role, they have the additional capability to update the server's secret key used for encrypting tickets. The server possesses three keys: two RSA keys, which are private and public RSA keys serving for symmetric encryption, and another private key used by the server for encrypting tickets. An admin user has the authority to update this key for ticket encryption.

LIBRARIES AND RESOURCES

Database content for users:

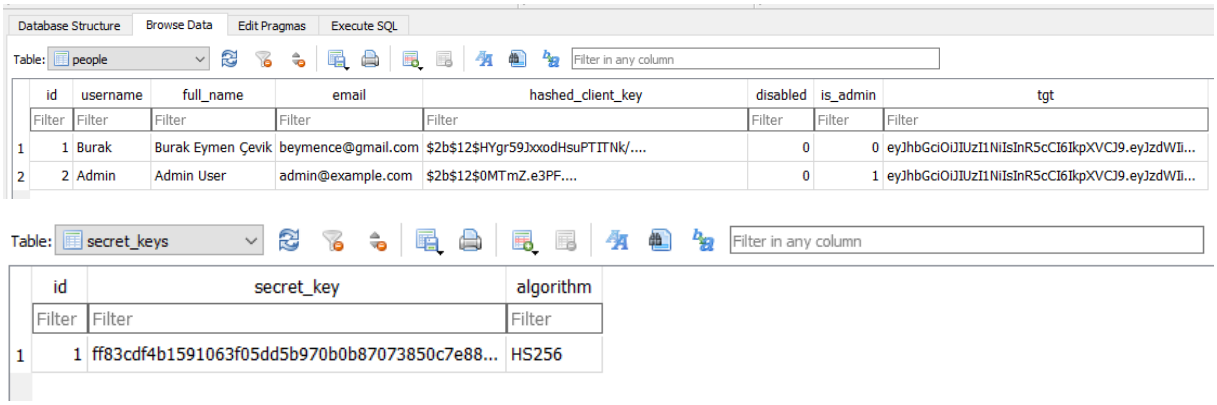


Table: people

	id	username	full_name	email	hashed_client_key	disabled	is_admin	tgt
1	1	Burak	Burak Eymen Çevik	beymence@gmail.com	$2b$12$HYgr59JxxodHsuPTITNk/....	0	0	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIi...
2	2	Admin	Admin User	admin@example.com	$2b$12$0MTmZ.e3PF....	0	1	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIi...

Table: secret_keys

	id	secret_key	algorithm
1	1	ff83cdf4b1591063f05dd5b970b0b87073850c7e88...	HS256

Libraries for Client Side:

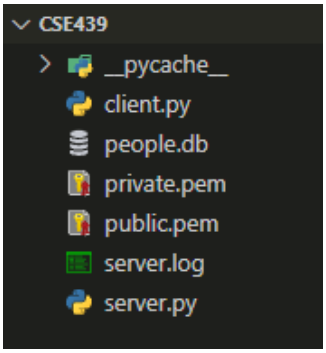
```
import requests
import base64
import rsa
```

Libraries for Server Side:

```
from fastapi import FastAPI, HTTPException, status, Depends, File, UploadFile
from fastapi.security import OAuth2PasswordBearer, OAuth2PasswordRequestForm
from pydantic import BaseModel
from datetime import datetime, timedelta
from jose import JWTError, jwt
from passlib.context import CryptContext
from sqlalchemy import create_engine, Column, Integer, String, Boolean
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker
from urllib.parse import parse_qsl
import rsa
import base64
import logging
import os
import subprocess
import pytz
```

Codes for client side and server side are included in project folder.

Project files:



EXPLANATION FOR EACH FUNCTION AT CLIENT SIDE

encrypt_and_send_credentials(username, password, public_key):

Purpose:

Encrypts the username and password using the server's public key before sending it for authentication.

Input:

username: User's username.

password: User's password.

public_key: Server's public key.

login(username, password):

Purpose:

Initiates the login process by sending encrypted credentials to the server.

Input:

username: User's username.

password: User's password.

Output:

Returns the TGT (Ticket Granting Ticket) if the login is successful.

is_admin_user(token):

Purpose:

Checks if the authenticated user has admin privileges.

Input:

token: Ticket that is obtained after Ticket request to TGS and TGT validation.

Output:

Returns True if the user is an admin; otherwise, returns False.

update_server_key(token):

Purpose:

Sends a request to update the server key (admin-only operation).

Input:

token: Ticket that is obtained after Ticket request to TGS and TGT validation.

Output:

Returns the result of the server key update.

get_current_user(token):

Purpose:

Retrieves and displays information about the currently authenticated user.

Input:

token: Ticket that is obtained after Ticket request to TGS and TGT validation.

Output:

Displays user information.

encrypt_and_send_new_password(new_client_key, public_key):

Purpose:

Encrypts the new client key before sending it for an update.

Input:

new_client_key: New client key entered by the user.

public_key: Server's public key.

update_client_key(token, new_client_key):

Purpose:

Sends a request to update the client key.

Input:

token: Ticket that is obtained after Ticket request to TGS and TGT validation.

new_client_key: New client key entered by the user.

Output:

Returns the result of the client key update.

get_current_time(token):

Purpose:

Sends a request to get the current server time.

Input:

token: Ticket that is obtained after Ticket request to TGS and TGT validation.

Output:

Displays the current server time.

validate_tgt(tgt):

Purpose:

Sends a request to validate the TGT and obtain a TGS (Ticket Granting Server) token.

Input:

tgt: TGT obtained after a successful login.

Output:

Returns the ticket if validation is successful.

logout(token):

Purpose:

Initiates the logout process by sending a request to invalidate the TGT or ticket.

Input:

token: Ticket that is obtained after Ticket request to TGS and TGT validation.

OR

token: TGT obtained after a successful login.

Output:

Displays a message indicating whether the logout was successful.

main():

Purpose:

Main function to interact with the user, initiate login, and provide a menu for various services.

Input:

Accepts user input for username and password.

Output:

Interacts with the user based on the chosen services.

EXPLANATION FOR EACH ENDPOINT AT SERVER SIDE

/authorization-and-tgt-creation (POST):

Purpose:

Handles user authentication and generates a Ticket Granting Ticket (TGT) if the user is authenticated.

Input:

encrypted_form_data: Encrypted user credentials (username and password).

Output:

Returns an access token (TGT) if the user is authenticated.

/tgt-validation-and-tgs (POST):

Purpose:

Validates the received TGT and generates a Ticket Granting Server (TGS) ticket.

Input:

update_data: TGT received from the client.

Output:

Returns a TGS ticket if the TGT is valid.

/users/me (GET):

Purpose:

Retrieves information about the currently authenticated user.

Input:

Requires a valid ticket in the request headers.

Output:

Returns user information.

/update-client-key (POST):

Purpose:

Allows a user to update their client key.

Input:

update_data: New client key encrypted with the server's public key.

Output:

Returns a message indicating successful key update and a new ticket.

/update-server-key (POST):

Purpose:

Allows an admin user to update the server's private key.

Input:

Requires admin privileges and authentication.

Output:

Returns a message indicating successful server's private key update.

/generate-new-server-key (GET):

Purpose:

Allows an admin user to generate a new server key.

Input:

Requires admin privileges and authentication.

Output:

Returns the newly generated server's private key.

/current-time (GET):

Purpose:

Returns the current time in the server's timezone (Europe/Istanbul).

Input:

Requires a valid ticket in the request headers.

Output:

Returns the current time.

/logout (POST):

Purpose:

Handles user logout by destroying the TGT if a ticket for the authenticated user yet.

If a user is authenticated and has a ticket then this endpoint handles user logout by destroying that ticket.

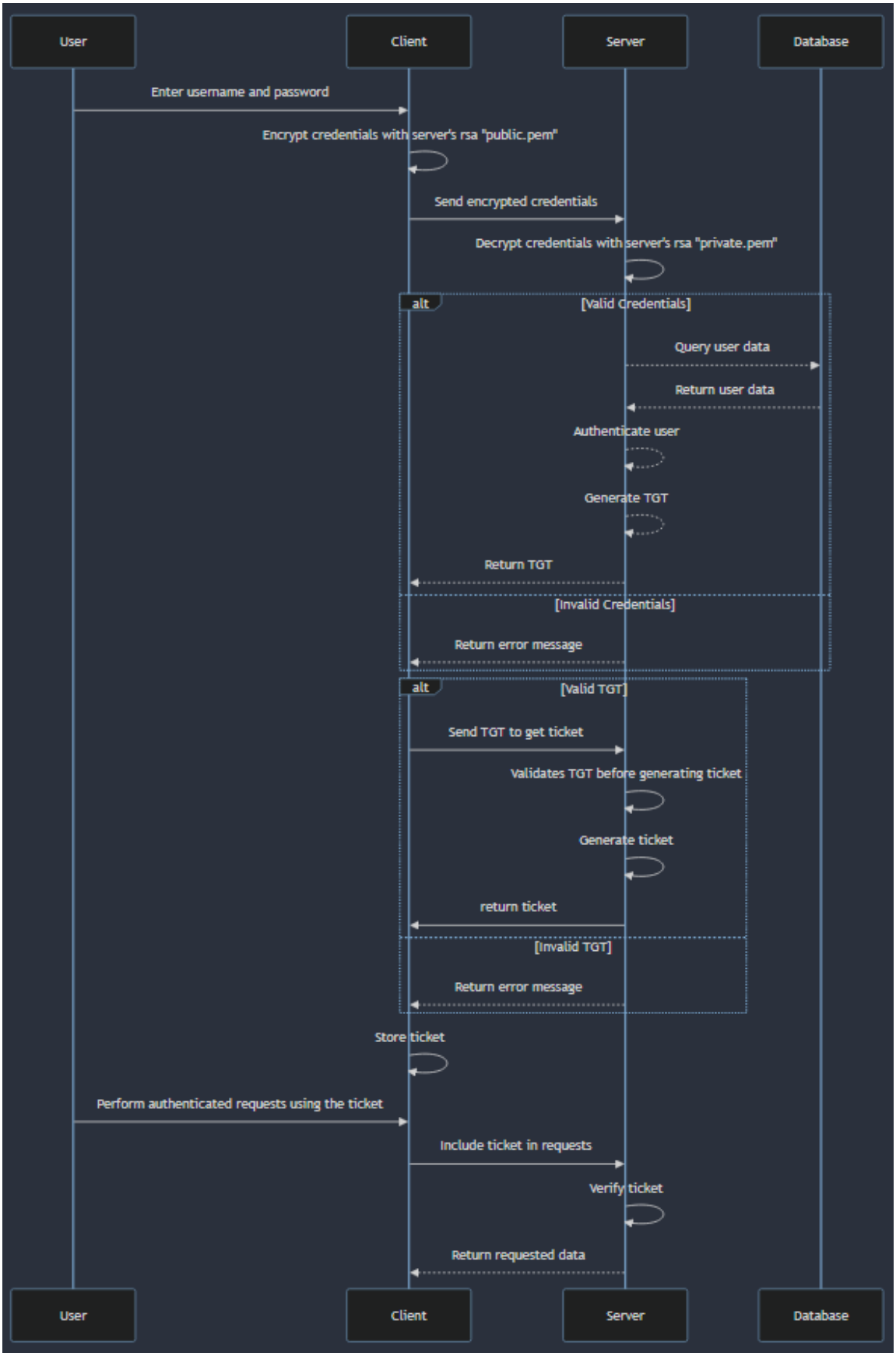
Input:

Requires a valid ticket or tgt in the request headers.

Output:

Returns a message indicating successful logout.

SEQUENCE DIAGRAM



REFERENCES

https://fastapi.tiangolo.com/tutorial/security/oauth2-jwt/

https://jwt.io/introduction

https://www.geeksforgeeks.org/rsa-algorithm-cryptography/