# Design of Processes

## Overview

The implementation involves a client-server architecture where the client interacts with an authentication server and a time server. The client can authenticate users, update server and client keys, and connect to a time server to request the current time. The authentication server manages user authentication, key management, and token generation.

## Client-Side Processes

### 1. User Authentication

- Users input their credentials (`username` and `password`).
- The client sends a request to the authentication server to validate these credentials.
- Upon successful validation, a JWT token is received and stored.

### 2. Key Management

- After authentication, the client requests the user-specific key from the server.
- This key is used for subsequent encrypted communications with the time server.

### 3. Time Server Communication

- The client establishes a connection with the time server.
- Commands (`time` or `exit`) are sent to the time server.
- For the `time` command, the current server time is requested. The time data is encrypted and decrypted using the user-specific key.

### 4. Menu Options

- An integrated menu is presented post-login, offering options to update server/client keys and connect to the time server.

### 5. Updating Server/Client Keys

- Options to update the server or client keys are provided.
- The update requests are authenticated with the stored JWT token.

## Server-Side Processes (AuthServer)

### 1. User Authentication and SignUp

- Handles `signup` and `authenticate` endpoints.

- User credentials are validated, and upon successful authentication, a JWT token is issued.
- New users are added to the database during the signup process.

### 2. Key Management

- Server keys (both client and server types) are managed.
- New keys can be generated and associated with specific users.

### 3. User Key Retrieval

- A dedicated endpoint (`getUserKey`) allows authenticated users to retrieve their specific client key.

### 4. Database Interaction

- User information, including credentials and keys, is stored and managed in a MongoDB database.

## TimeServer

### 1. Handling Client Requests

- Listens for client connections on a specified port.
- Receives and processes encrypted data from clients.
- Recognizes and responds to the `time` command by sending the current server time, encrypted with the user-specific key.

### 2. Data Encryption/Decryption:

- Utilizes the CryptoJS library for encrypting and decrypting data.
- Ensures secure communication between the server and authenticated clients.

## Security Measures

### 1. Encrypted Communication

- All sensitive data, including user credentials and time data, are encrypted during transmission.

### 2. Token-Based Authentication

- JWT tokens are used for authenticating requests to the server, enhancing security.

### 3. Key Management

- Keys are securely stored and associated with user accounts.
- Mechanisms are in place for key rotation and retrieval.

## 4. Error Handling

- Robust error handling is implemented to address potential issues during data processing and network communication.

**Aleyna Alangil**
**20190703109**