

QR코드를 활용한 차량용 안심 주차번호

BEST DRIVER

2023. 11. 20.

01

팀 이름과 팀 구성원 소개

Chapter.01

팀명: BEST DRIVER



정보보안암호수학과

20192207

김남일



정보보안암호수학과

20143369

유조은



AI빅데이터융합경영학과

20222852

김세은



AI빅데이터융합경영학과

20212570

이수인

02

프로젝트 동기

Chapter.02

프로젝트 배경

현재 차량 주차 시에 자신의 전화번호를 차량 앞부분에 부착하는 형식으로 주차 후 생기는 문제에 대해 연락받고 있다. 하지만 전화번호라는 개인정보가 너무 쉽게 노출이 되어있으므로 이를 악용하여 범죄에 이용하는 사건들이 생긴다.

(서울신문, 2021)

타인 차량 부착 전화번호 무단수집 처벌 가능한가...경찰, 분양 상담사 처리 고심

(네이버 뉴스, 2023)

"여기 살지?" 女 차량 전화번호 수집해 협박한 20대 징역

Chapter.02

프로젝트 배경

전화번호를 이용하여 카카오톡 및 sns를 통해 다른 개인정보들도 쉽게 취득할 수 있다.
그래서 전화번호를 OTP기술을 이용해 일회용 전화번호를 만드는 방법을 생각해보았다.



따라서 OTP기능을 이용하여 안심번호를 생성해 자신의 실제 핸드폰 번호 대신 안심번호를 사용하는 방법을 구상해 보았다.

Chapter.02

프로젝트 아이디어



OTP를 활용하여 일회용 전화번호 만들기 !



OTP 유형 중 **질의-응답** 방식 적용

질의-응답 선정 이유



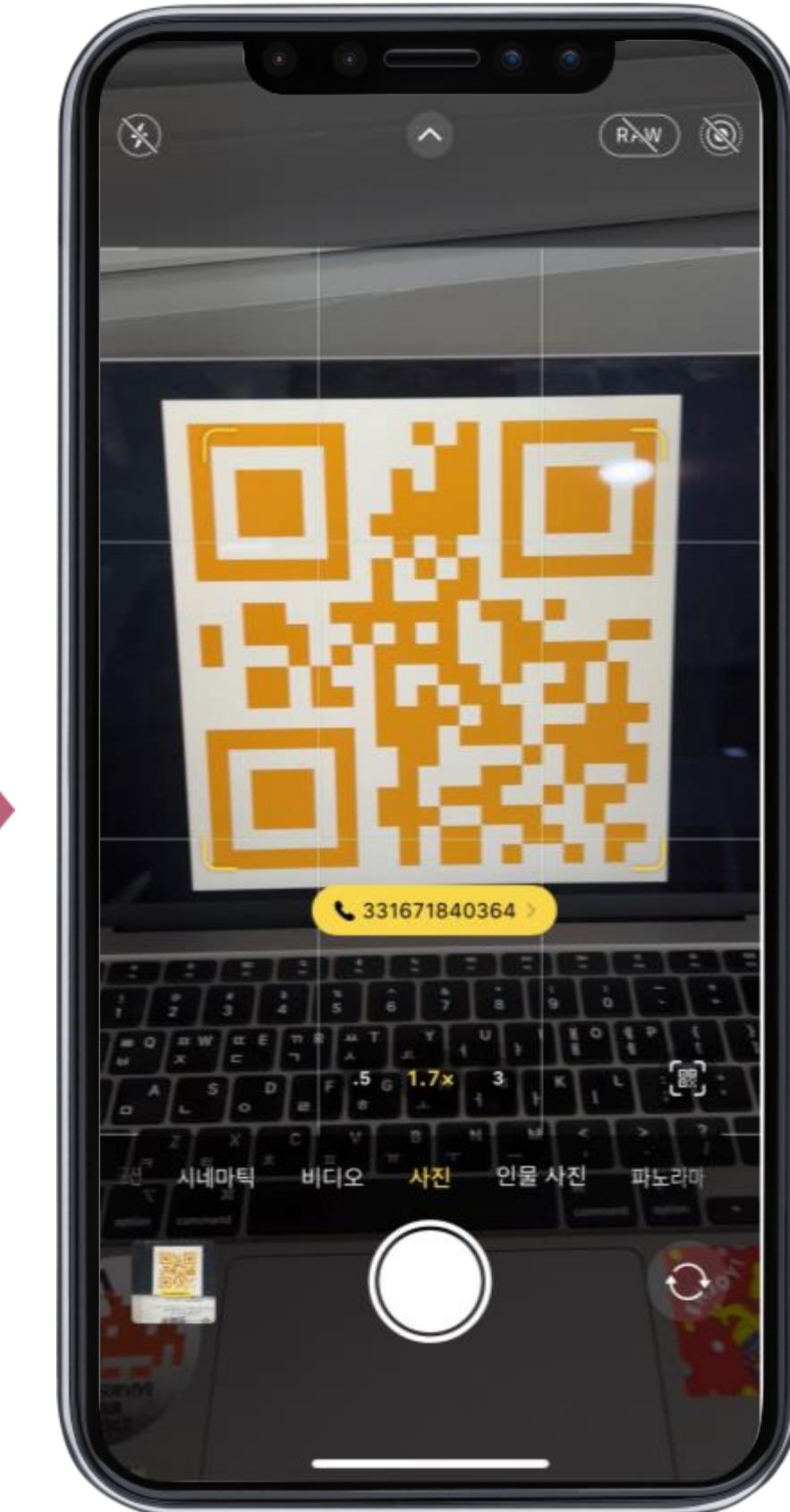
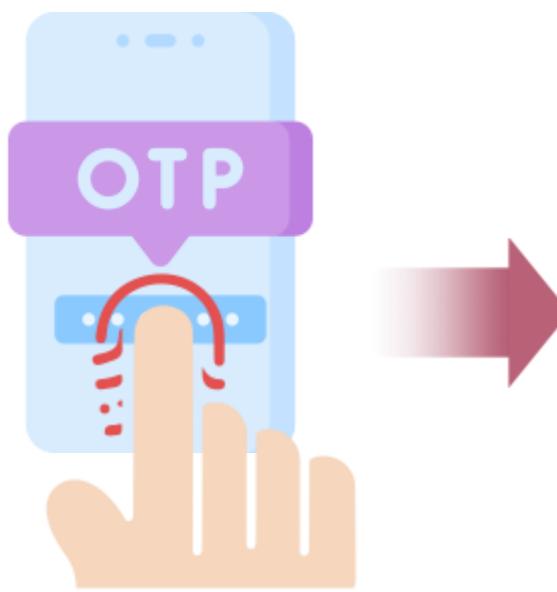
안심번호는 사용자가
갱신을 원하기 전까지는
번호가 유지되어야
상대방과 연락이 필요한 상황에
문제가 발생하지 않으므로
질의응답 방식을 선택하였다.

03

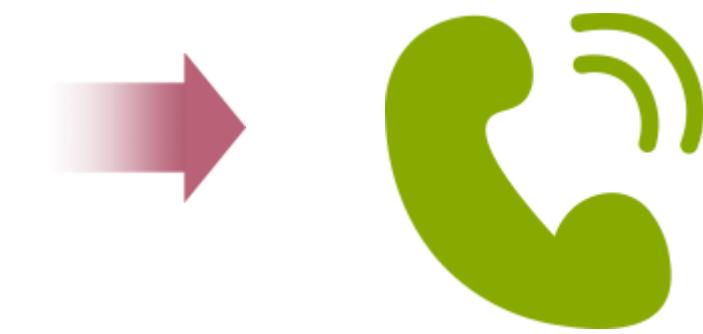
프로젝트 가상실현 화면

Chapter.03

가상 실현



331471840364



지문을 등록할 때마다 안심번호가 바뀐다!

04

계획

Chapter.04

계획

STEP 1. 문제 수립



문제 정의 현황 및 한계점 조사

질의-응답 방식 문제에 맞는 OTP 선택
후, 질의-응답 방식 알고리즘 구조화

STEP 2. OTP 선택



STEP 3. 구현



안심번호 QR 코드 + 일회용비밀번호
알고리즘으로 안심번호 생성



05

프로젝트 진행 보고



Chapter.05

인증단계

가정: C는 사전에 S(서버)에 id, K 등록
ID는 본인 핸드폰번호
 $K = H(\text{지문값})$, 지문값은 임의의 사진파일로 대체
(1) $C \rightarrow S: ID$
(2) $C \leftarrow S: RV$ (서버에서 생성한 임의의 난수)
(3) $C \rightarrow S: otp, otp = H(K \oplus RV), H = sha256$
otp를 바탕으로 서버에서 안심번호 생성 후
QR에 안심번호 정보를 담아 디바이스를 통해 보여준다.

디바이스(otp생성장치)와 지문정보로
2 Factor 기준을 성립한다.

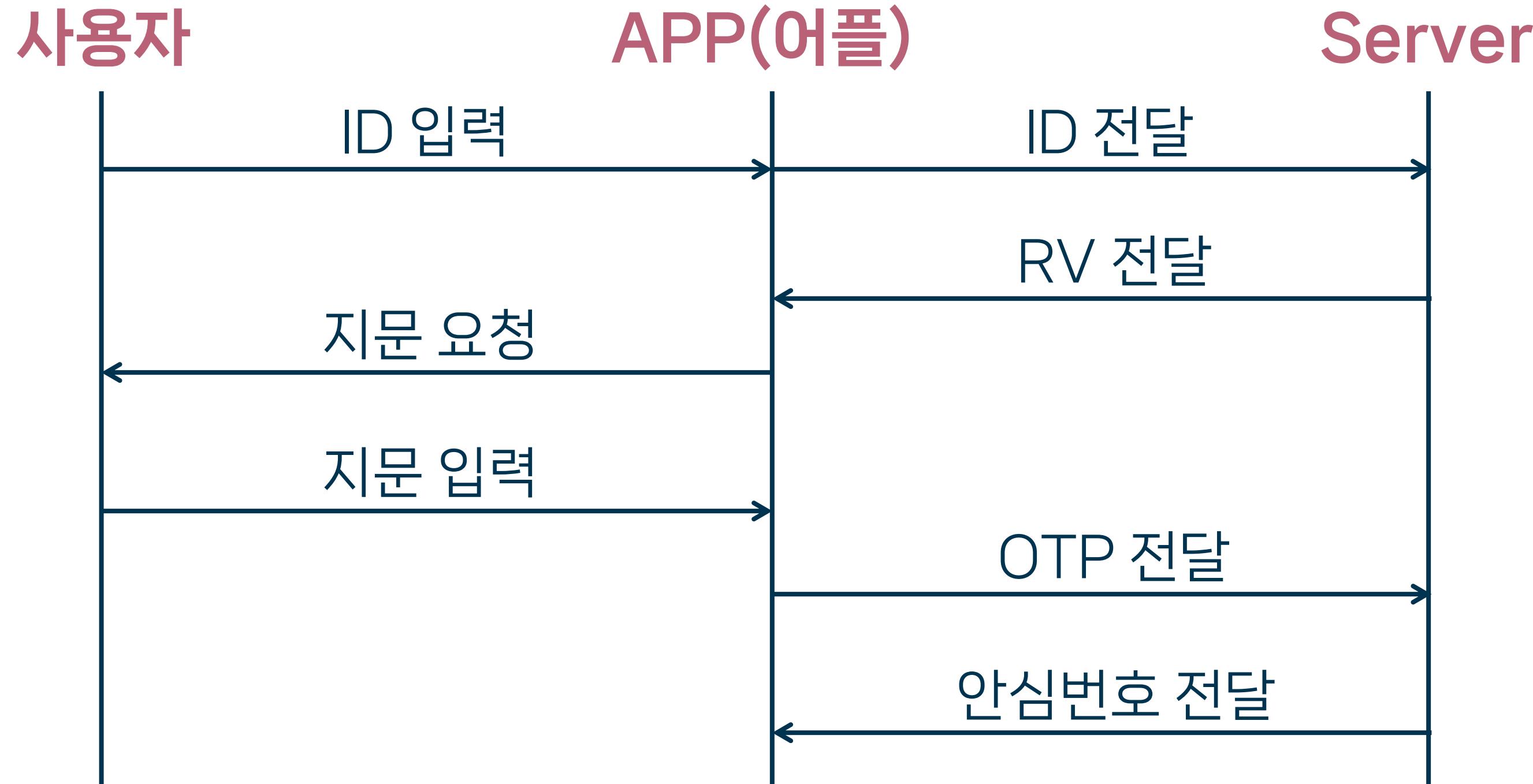


Chapter.05

실제 동작 과정

1. Server에 아이디, 지문 등록
2. 사용자가 app을 통해 아이디 입력
3. app은 서버에 ID전달
3. Server는 ID 유효성 확인 후 유효하다면 앱에게 난수 전달
4. 앱은 사용자에게 지문인증 요청
5. 사용자는 앱에 지문 입력
6. 앱은 지문과 핸드폰 번호로 K 생성 후 RV를 이용해 otp생성 후 서버에 전달
7. 서버는 미리 등록해둔 K와 RV이용해 otp계산 후 앱에서 받은 otp가 같은지 확인
8. 서버는 앱에게 안심번호 전달
9. 앱은 안심번호 정보를 담고 있는 QR코드 생성

실제 동작 과정



Chapter.05

Server.ipynb

import

```
import socket # 소켓 통신을 위한 모듈
import threading # 스레딩을 위한 모듈
import os # 운영체제 관련 기능을 사용하기 위한 모듈
import hashlib # 해시 함수를 위한 모듈

# 미리 저장된 아이디 목록
valid_ids = ["user1", "01047034826", "01012345678"]

# 서버에서 사용할 고정 비밀 값
server_PW = "f59ecdf26cb91b476bbc9dde2fe3440ce1368cce0e73320b4066be8378dafff5"
```

Chapter.05

Server.ipynb

```
# 클라이언트 연결을 처리하는 함수
def handle_client(client_socket, address):
    print(f"{address}에서의 연결이 성공적으로 이루어졌습니다.")

    # ID를 입력받음
    client_socket.send("서버에 연결되었습니다.".encode('utf-8'))
    client_id = client_socket.recv(1024).decode('utf-8').strip()
    if client_id in valid_ids:
        print(f"{address}의 ID: {client_id}")

        # 아이디 확인이 완료되었다는 메시지와 랜덤 데이터를 전송
        client_socket.send("아이디 확인이 완료되었습니다.".encode('utf-8'))
        random_data = os.urandom(16)
        client_socket.send(random_data)

    ...
    ...
```

Chapter.05

Server.ipynb

...

```
# 클라이언트로부터 OTP를 받기
client_otp = client_socket.recv(1024).decode('utf-8')

# otp유효성 확인을 위한 otp를 서버에서 생성
server_otp = generate_server_otp(random_data)

# 비교 후 결과를 출력하고 클라이언트에게 전송
if server_otp == client_otp:
    print("OTP 확인이 완료되었습니다.")
    client_socket.send("OTP 확인이 완료되었습니다".encode('utf-8'))
else:
    print("OTP가 일치하지 않습니다.")
    client_socket.send("OTP 확인에 실패하였습니다".encode('utf-8'))

...
```

Chapter.05

Server.ipynb

...

```
while True:  
    try:  
        message = client_socket.recv(1024).decode('utf-8')  
        if not message:  
            break  
        print(f"{client_id}({address})로부터의 메시지: {message}")  
  
    except ConnectionResetError:  
        print(f"{address}와의 연결이 강제로 종료되었습니다.")  
        break  
  
else:  
    print(f"{address}의 ID({client_id})가 유효하지 않습니다. 연결을 종료합니다.")  
client_socket.close()
```

Chapter.05

Server.ipynb

```
# 서버에서 OTP 생성 함수
def generate_server_otp(random_data):

    # 클라이언트와 동일한 방식으로 OTP 생성
    otp_combined_data = bytes(x ^ y for x, y in zip(random_data, bytes.fromhex(server_PW)))
    server_otp = hashlib.sha256(otp_combined_data).hexdigest()
    print("Server OTP calculated:", server_otp)
    return server_otp
```

Chapter.05

Server.ipynb

```
# 서버 설정
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(('127.0.0.1', 12345))
server_socket.listen(5)
print("서버가 포트 12345에서 대기 중입니다...")

# 스레드를 사용하여 수신된 클라이언트 연결을 처리
while True:
    client, address = server_socket.accept()
    client_handler = threading.Thread(target=handle_client, args=(client, address))
    client_handler.start()
```

Chapter.05

main.ipynb

import

```
# 소켓 및 기타 모듈 임포트
import socket # 네트워크 통신을 위한 소켓 모듈
import hashlib # 해시 함수를 사용하기 위한 모듈
import time # 시간 관련 함수를 사용하기 위한 모듈
import qrcode # QR 코드 생성을 위한 모듈
import matplotlib.pyplot as plt # 시각화를 위한 모듈
from PIL import Image # 이미지 처리를 위한 모듈
```

Chapter.05

main.ipynb

App

```
# 서버 주소 및 포트 설정
server_address = ('localhost', 12345)

# 서버에 연결
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(server_address)

# 환영 메시지 수신 및 출력
welcome_message = client_socket.recv(1024).decode('utf-8')
print(welcome_message)

# 사용자 ID 입력 및 서버로 전송
client_id = input("Enter your ID: ")
client_socket.send(client_id.encode('utf-8'))
```

Chapter.05

main.ipynb

App

```
# ID 확인 메시지 수신 및 출력  
id_confirmation_message = client_socket.recv(1024).decode('utf-8')  
print(id_confirmation_message)
```

```
# 서버로부터 랜덤 데이터 (RV) 수신  
RV = client_socket.recv(16)  
print("Server sent random data (RV):", RV)
```

Chapter.05

main.ipynb

App

```
# 파일에서 지문을 읽고 SHA-256 해시 계산
def get_fingerprint():
    with open('fingerprint.jpg', 'rb') as file:
        fingerprint_data = file.read()
    return fingerprint_data

# ID와 지문을 사용하여 K 생성
def generate_K(my_phone_number, fingerprint_data):
    concatenated_data = my_phone_number.encode('utf-8') + fingerprint_data
    K = hashlib.sha256(concatenated_data).hexdigest()
    return K

fingerprint_data = get_fingerprint()
K = generate_K(client_id, fingerprint_data)
```

Chapter.05

main.ipynb

OTP 생성기

```
# RV와 지문 해시를 XOR을 사용하여 OTP 계산
otp_combined_data = bytes(x ^ y for x, y in zip(RV, bytes.fromhex(K)))
otp = hashlib.sha256(otp_combined_data).hexdigest()
client_socket.send(client_otp.encode('utf-8'))
print("OTP calculated:", otp)

while True: # 연결을 유지하기 위한 메시지 루프
    message = input("Enter a message (type 'exit' to close the connection): ")
    client_socket.send(message.encode('utf-8'))
    if message.lower() == 'exit':
        break
    response = client_socket.recv(1024).decode('utf-8')
    print(f"Response from the server: {response}")
client_socket.close() # 연결 종료
```

Chapter.05

main.ipynb

App

```
# OTP에 의해 파생된 안심번호 만들기
def hash_to_number(hash_value):
    # 해시 값을 16진수에서 10진수로 변환
    decimal_number = int(hash_value, 16)
    return decimal_number

# 해시된 값을 10진수로 변환하여 안심번호 생성
safe_number = str(hash_to_number(client_otp))
safe_phone = safe_number[:12]
print("생성된 안심번호:", safe_phone)
```

Chapter.05

main.ipynb

차량용 주차 안심번호 QR코드

```
# OTP를 QR 코드 데이터로 변환하여 QR 코드 생성
def generate_qr_code(data):
    # QR 코드 생성을 위한 설정
    qr = qrcode.QRCode(version=1, error_correction=qrcode.constants.ERROR_CORRECT_H,
                        box_size=10, border=1)
    # 데이터를 QR 코드에 추가하고 이미지 생성
    qr.add_data(data)
    qr.make(fit=True)
    # QR 코드 이미지 저장
    img = qr.make_image(fill_color="darkorange", back_color="white")
    img.save("safe_phone_qr_code.png")

# 안심번호를 사용하여 QR 코드 생성
generate_qr_code(safe_phone)
```

06

팀 구성원의 기여도



계획 발표 역할 분담



김남일



유조은



김세은



이수인

인증단계 구조화

- OTP 인증단계 구체화
- OTP 선정 이유 조사

문제 정의

- 문제 현황 및 한계점 조사
- 안심번호 아이디어 제시

안심번호 구현 및 발표 준비

- Challenge-Response 기법 적용한 안심번호 생성
- QR 코드 생성 및 안심번호 연동, PPT 제작

최종 발표 역할 분담



김남일



유조은



김세은



이수인

인증단계 구조화

- OTP 인증단계 구체화
- OTP 선정 이유 조사

문제 정의

- 문제 현황 및 한계점 조사
- 안심번호 아이디어 제시

서버와 클라이언트의 대화 생성

- 서버 구현 및 향후 계획 및 최신 기술 조사

안심번호 구현 및 발표 준비

- Challenge-Response 기법 적용한 안심번호 생성
- QR 코드 생성 및 안심번호 연동, PPT 제작

알고리즘 구체화 및 PPT 제작

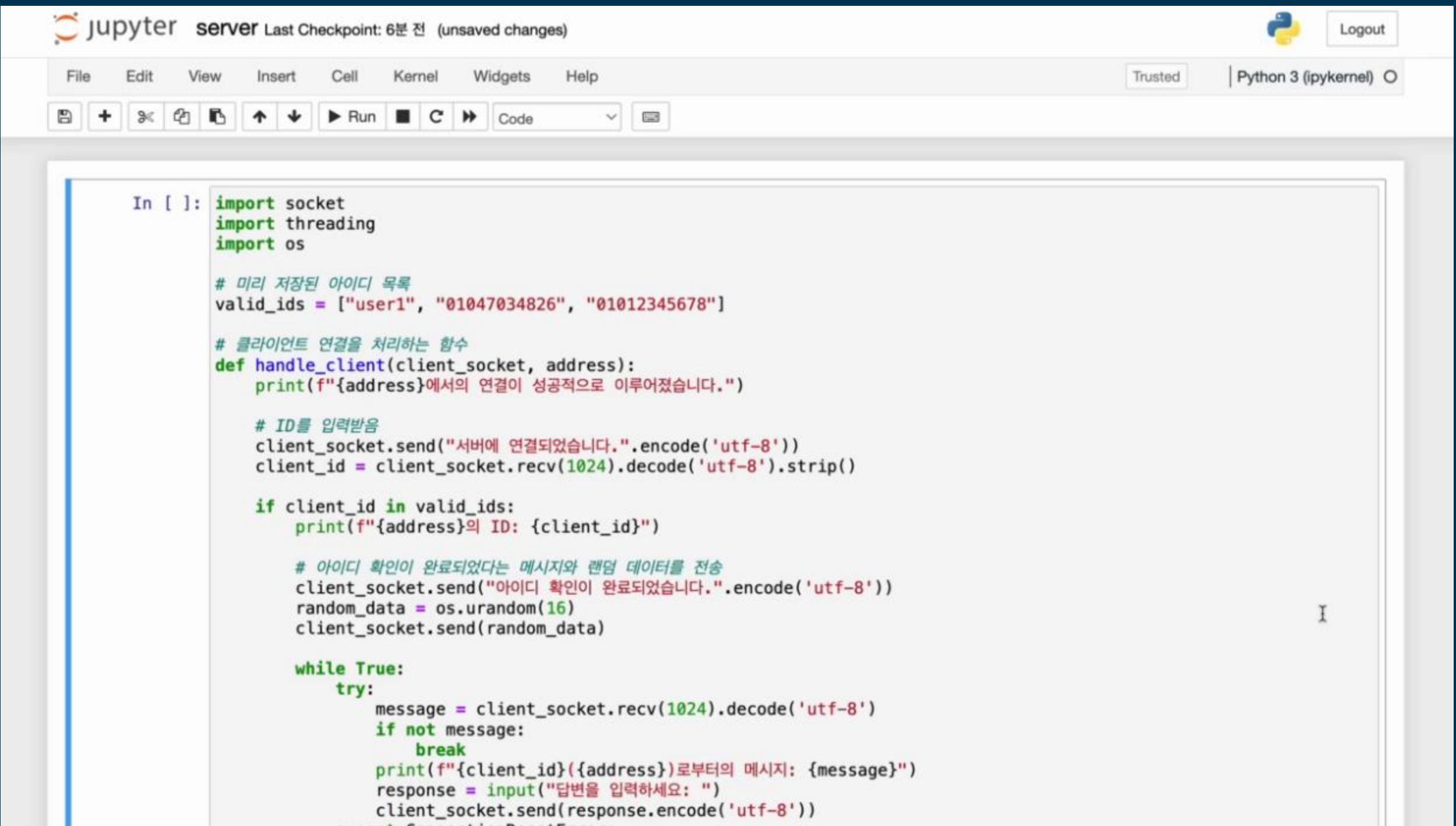
- 지문의 해쉬 값 생성 및 안심번호와 QR코드 생성

07

결과물 시연

Chapter.07

결과물 시연 영상 첨부파일 참고



The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** jupyter server Last Checkpoint: 6분 전 (unsaved changes)
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, Python 3 (ipykernel)
- Code Cell:** In []:

```
import socket
import threading
import os

# 미리 저장된 아이디 목록
valid_ids = ["user1", "01047034826", "01012345678"]

# 클라이언트 연결을 처리하는 함수
def handle_client(client_socket, address):
    print(f"{address}에서의 연결이 성공적으로 이루어졌습니다.")

    # ID를 입력받음
    client_socket.send("서버에 연결되었습니다.".encode('utf-8'))
    client_id = client_socket.recv(1024).decode('utf-8').strip()

    if client_id in valid_ids:
        print(f"{address}의 ID: {client_id}")

        # 아이디 확인이 완료되었다는 메시지와 랜덤 데이터를 전송
        client_socket.send("아이디 확인이 완료되었습니다.".encode('utf-8'))
        random_data = os.urandom(16)
        client_socket.send(random_data)

    while True:
        try:
            message = client_socket.recv(1024).decode('utf-8')
            if not message:
                break
            print(f"{client_id}({address})로부터의 메시지: {message}")
            response = input("답변을 입력하세요: ")
            client_socket.send(response.encode('utf-8'))
        except Exception as e:
            print(f"An error occurred: {e}")
            break
```

Chapter.07

프로그램 실행 결과



QR코드를 카메라에 대면 안심번호에 접근 가능



08

최신 인증 기술

Chapter.08

최신 인증 기술

" FIDO 인증 프로토콜 "

First. 안전성



처음 ID/PW를 이용할 때
생기는 보안의 문제점을
안면인식 등 생체정보 활용을
통해 로그인을 하기 때문에
더 안전해짐

Second. 편의성



생체정보 활용 인증자체가
사용자 인증이므로
OTP생성기 등
다른 외부장치가
필요하지 않음

Third. 보안성



FIDO프로토콜 중 생체정보는
인증 프로토콜 시작을 위한
정보일 뿐 외부로 전송되거나
하지 않음

THANK YOU
FOR ATTENTION

BEST DRIVER

2023. 11. 20.