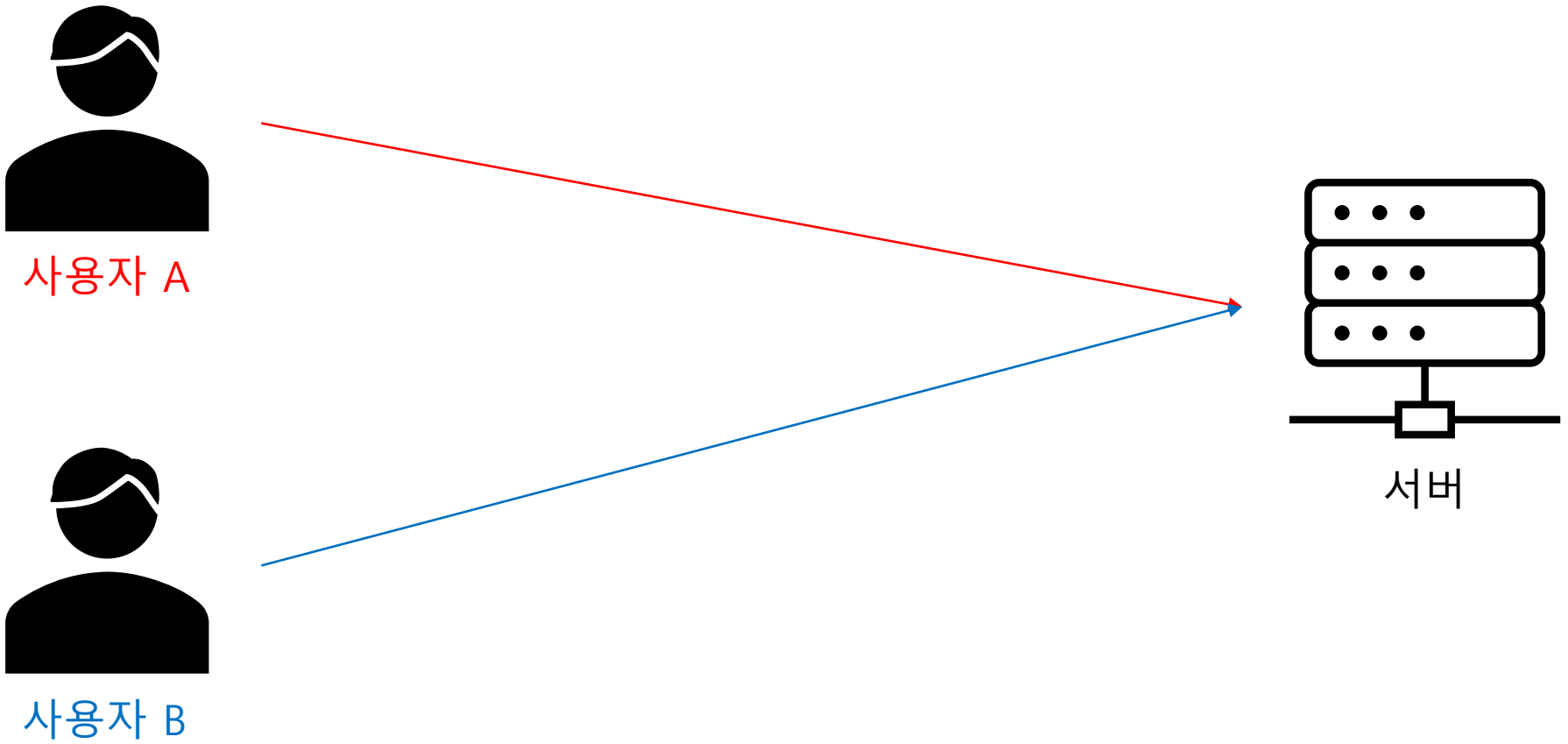
Session 기반 인증과 Token 기반 인증

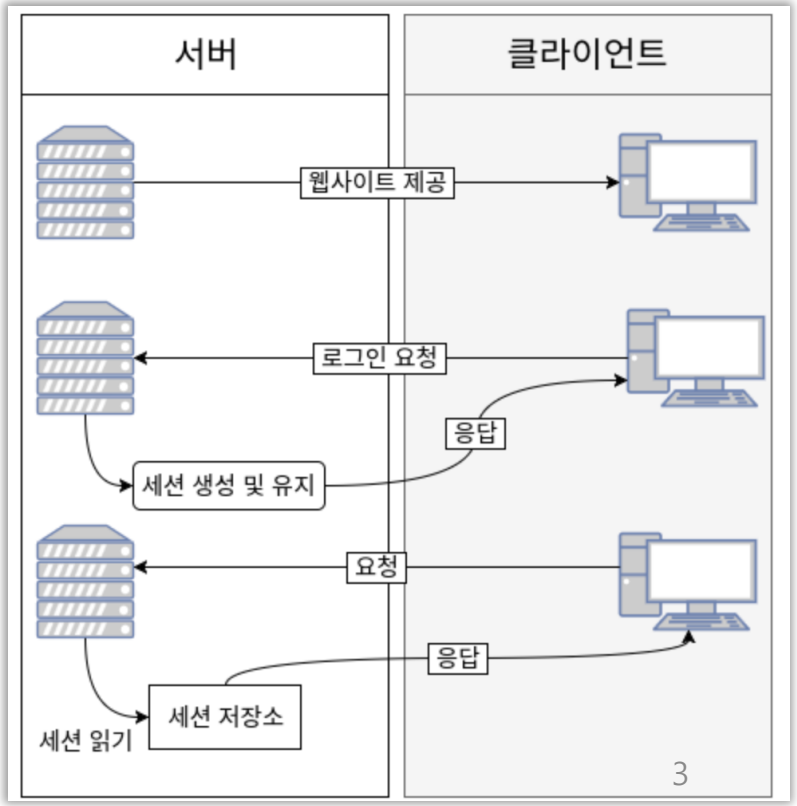
210415 이유경

인증이 필요한 이유



Session 기반 인증 방식

- Cookie/Session
- 인증 정보를 클라이언트와 서버가 모두 유지하고 있어야 함.
- 방식
  1. 사용자가 로그인을 함.
  2. 사용자 확인 후, 고유한 값을 부여해 세션을 발행
  3. 사용자는 세션을 쿠키에 저장한 후, 인증이 필요할 때마다 쿠키를 헤더에 실어서 보냄.
  4. 서버에서는 세션 아이디와 서버 저장소에 있는 세션 아이디를 비교하여 사용자에 맞는 데이터를 보냄.

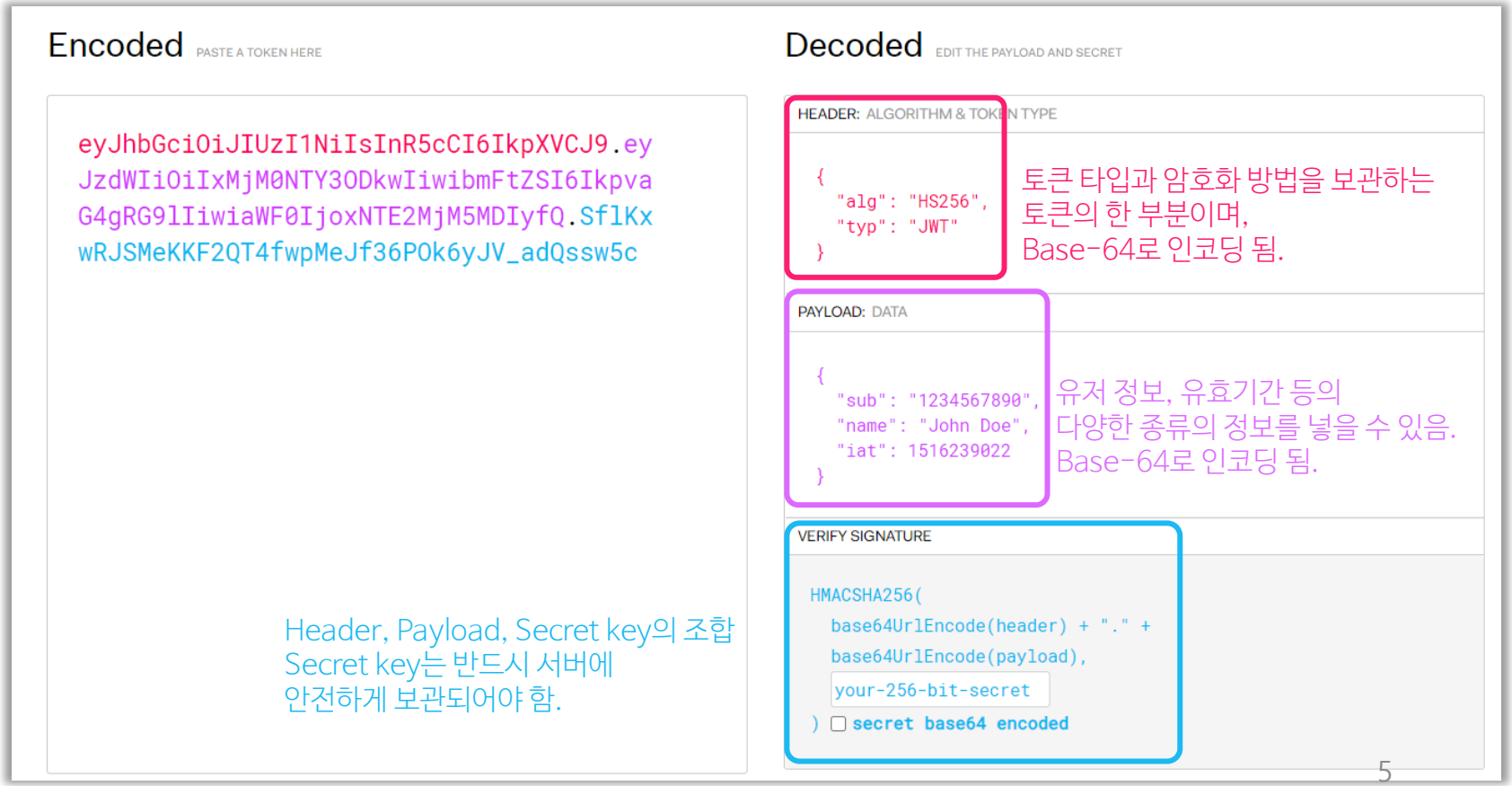


Session 기반 인증 방식

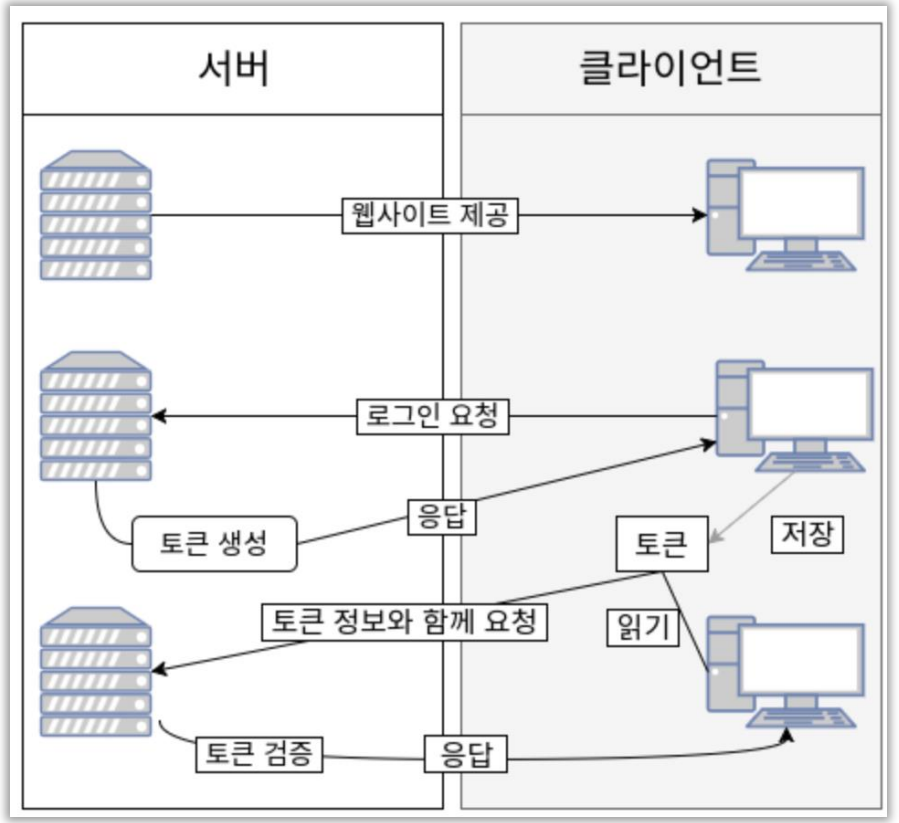
- 단점
  - 메모리 과부화
  - 확장성 문제
  - CORS(Cross-Origin Resource Sharing)
    - 쿠키는 기본적으로 도메인을 기반으로 설계가 되어있음. 따라서 서로 다른 여러 도메인에서 관리하는 것이 번거로움.

Token 기반 인증 방식

- Json web token
- 구성
  1. Header
  2. Payload
  3. Signature



Token 기반 인증 방식



- 방식
  1. 사용자가 로그인을 함.
  2. 서버에서 사용자 확인 후, 고유한 값을 부여한 후, 기타 정보와 함께 Payload에 넣음.
  3. JWT 토큰의 유효기간을 설정함.
  4. 암호화할 SECRET KEY를 이용해 토큰 발급
  5. 사용자는 토큰을 받아 저장한 후, 인증이 필요할 때마다 토큰을 헤더에 실어서 보냄.
  6. 서버에서는 해당 토큰의 Signature를 SECRET KEY로 복호화한 후, 조작 여부, 유효기간을 확인함.
  7. 검증이 완료된다면, Payload를 디코딩하여 사용자에 맞는 데이터를 가져옴.

Token 기반 인증 방식

- 특징
  - 유저의 정보를 서버나 세션에 담아두지 않음 → 서버측 메모리 과부화 해소
  - 서버를 손쉽게 확장

차이점

- Cookie/Session는 세션 저장소에 유저의 정보를 넣는 반면,
  JWT는 토큰 안에 유저의 정보들을 넣음.
- 물론 클라이언트 입장에서는 HTTP 헤더에 세션ID나 토큰을 실어서 보내준다는 점에서는 동일하나,
  서버 측에서는 인증을 위해 암호화를 하냐, 별도의 저장소를 이용하냐는 차이가 발생함.