통신 규약

1. 통신 구조

통신 모델 : 클라이언트-서버
통신 방식 : RSA/AES 암호화 + TCP 소켓 통신
데이터 전송 방식: JSON

2. 메시지 형식

메시지 형식 : 암호화된 JSON String

i) 초기 암호화 설정 과정 통신 형식

-서버 측 전송 형식

필드명	header	status	server_info	message
타입	JSON String	String	String	String
설명	status, server_info, message	현재 상태	서버 정보	전달할 메시지

필드명	publicKey	sessionKey
타입	String	RSA 공개키 암호화 String
설명	서버의 공개키 정보	클라이언트와 암호화 통신을 위해 전달하는 키

ex) { “header” : { “status” : “connecting”, “server_info” : “multiroom server”, “message” : “publicKey를 이름으로 하는 공개키를 보내세요” } , “publicKey” : “dd” }
{ “header” : { “status” : “connecting”, “server_info” : “multiroom server”, “message” : “sessionKey를 받았으면 잘 받았다는 요청을 status 이름으로 보내세요” } , “sessionKey” : “dsffsdfasd” }

-클라이언트 측 전송 형식

필드명	publicKey	status
타입	String	String
설명	클라이언트의 공개키 전송	세션키를 잘 받았는지 여부를 전송 session_key_received or fail

ex) { “publicKey” : “sadfsdfasd” }
{ “status” : “session_key_received” }

ii) 암호화 설정 이후 통신 형식

-클라이언트 측 전송 형식

필드명	command	data
타입	String	JSON String | String
설명	요청의 종류 파악용	요청에 필요한 파라미터

ex) { “command” : “echo”, “data” : data }
1. 해당 JSONString을 AES 암호화해서 전송
2. JSONString의 앞자리에는 암호화에 사용한 IV(초기화벡터)를 추가함

-서버 측 전송 형식

필드명	status	data	message
타입	String	JSON String	String
설명	응답의 종류 파악용	응답 데이터	처리에 실패한 경우 메시지 전달

ex) { “status” : “success”, “data” : data }
{ “status” : “invalid”, “message” : “메시지...” }
{ “status” : “fail”, “message” : “메시지...” }

1. status를 제외한 data부분에 대해서 AES 암호화해서 전송
2. status가 invalid이거나 fail인 경우 암호화하지 않음

3. 응답 상태

코드: connecting, success, invalid, fail,
코드 처리 방식
connecting : 암호화 연결 과정
success : 성공적으로 data를 만들어 보냄
----------> 아래 에러코드에 대해서는 추후 조정
invalid : 유효하지 않은 요청의 경우에 대한 메시지
fail : 요청을 처리 실패한 경우에 대한 메시지

4. 보안

암호화: RSA(세션키 암호화), AES256(통신 암호화), SHA256(단방향 해싱)
보안 인증: SessionKey
-데이터 통신 과정에서 SessionKey로 데이터를 암호화하고 복호화해서 통신한다.
따라서 서버와 클라이언트 간의 SessionKey가 동일한 경우에 대해서 통신이 가능하다

5. 예제

요청 예제 : IV(초기화벡터) + 암호화 된 JSON String
IV값(16byte)+{ command : “요청 명령어”, data : “파라미터” }

응답 예제 : IV(초기화벡터) + 암호화 된 JSON String
{ status : “응답 성공 or 실패”, data : “IV값(16byte)+암호화된 응답 데이터” }

6. 제한 사항 및 참고 사항

제한 사항 :

i) 초기 암호화 연결 설정 과정 : 이 과정에서 시간이 **10초 이상 지연**되는 경우 **status는 fail**을 클라이언트 측에 전송하고 서버의 소켓접속이 끊어짐.

ii) 서버에서 **세션 제공** : 로그인과 같은 인증 작업과 데이터 연결 유지가 필요한 경우 **세션ID**가 발급되고 발급된 이후의 통신부터 세션ID가 만료될 때까지 세션ID를 사용해서 **클라이언트-서버 간의 연결성** 부여. 그 외에 검색이나 조회 같은 권한이나 인증이 필요하지 않은 작업에 대해서는 세션ID를 사용하지 않아도 요청을 처리하도록 함.

참고 사항 :

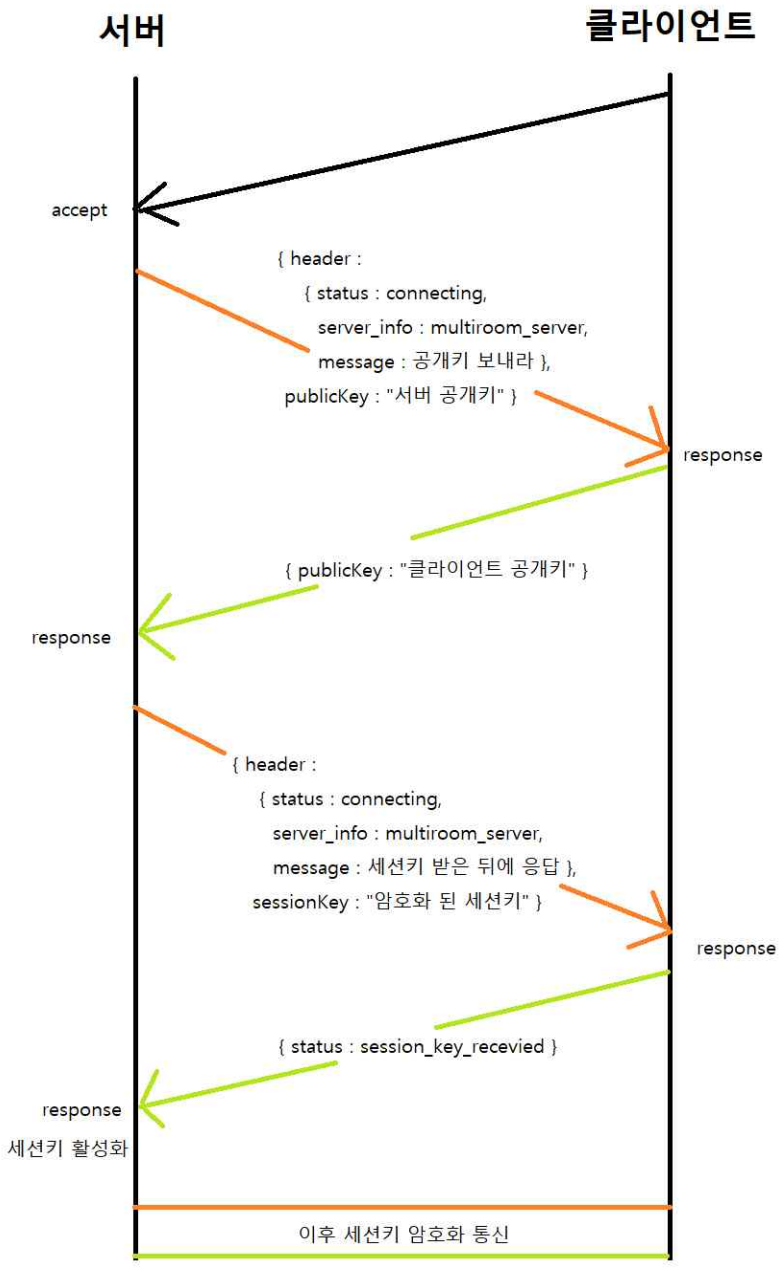
I) 초기 암호화 연결 설정 과정
-RSA 비대칭키 암호화를 사용한다.
  i) 클라이언트가 접속 시 서버는 클라이언트 측에 서버의 정보를 제공하고 공개키를 요청
  ii) 클라이언트는 서버의 정보를 받고 신뢰 가능한지 판단 후 자신의 공개키를 전달
  iii) 서버는 세션 키를 클라이언트의 공개키로 암호화하여 클라이언트에 전달
  iv) 클라이언트는 세션 키를 비밀키로 복호화하여 저장하고 서버 측에 받은 사실 응답
  v) 서버는 클라이언트의 응답을 받은 뒤 세션 키를 활성화한다. 이후 요청 대기.

II) 암호화 설정 이후 통신 과정
-AES 대칭키 암호화를 사용한다.
  i) 클라이언트는 요청할 데이터를 세션 키로 암호화하여 전송한다
  ii) 서버는 요청받은 데이터를 세션 키로 복호화하여 명령을 분석하고 처리한다.
  iii) 서버는 처리한 뒤에 응답 데이터를 클라이언트로 동일하게 암호화하여 넘긴다.
  (이 때, status와 같은 응답 상태 데이터는 암호화하지 않는다)
  iv) 클라이언트는 서버로부터 받은 응답 상태를 확인하고 이에 대해 적절히 처리한다.

III) 암호화 통신 시 전달 형태
IV(초기화 벡터)는 세션 키와 함께 JSONString의 암/복호화 과정에서 사용하고 JSONString 앞에 붙여서 클라이언트 또는 서버로 전달됨. 각 종단점에서는 세션키와 전달받은 IV를 사용해서 복호화 과정을 거친 후 명령(command) 또는 상태(status)를 해석함.

7. 부록

i) 초기 암호화 연결 설정 과정



ii) 암호화 설정 이후 통신 과정