

# Geofront 개발 후기

Python 2와 작별하고 Python 3로 개발하기

(주)스포카  
홍민희

발표자: 홍민희

# 발표자: 홍민희

- 스포카(spoqa.com)에서  
도도포인트(dodopoint.com) 개발

# 발표자: 홍민희

- 스포카(spoqa.com)에서  
도도포인트(dodopoint.com) 개발
- 자유 소프트웨어 지지자, 오픈 소스 애호가

# 발표자: 홍민희

- 스포카(spoqa.com)에서  
도도포인트(dodopoint.com) 개발
- 자유 소프트웨어 지지자, 오픈 소스 애호가
- 곧 파이썬 10년차

# 발표자: 홍민희

- 스포카(spoqa.com)에서  
도도포인트(dodopoint.com) 개발
- 자유 소프트웨어 지지자, 오픈 소스 애호가
- 곧 파이썬 10년차
- <http://hongminhee.org/>

# 이상한 발표

# 이상한 발표

- 총 슬라이드 80장



# 이상한 발표

- 총 슬라이드 80장
- 그 중에서 30장이 지오프론트 소개

# 이상한 발표

- 총 슬라이드 80장
- 그 중에서 30장이 지오프론트 소개
- 파이썬 3 얘기는 45장

# 분량 조절에 실패

# 분량 조절에 실패

- 원래는 라이트닝 토크로 파이썬 3 얘기나 하려 했으나

# 분량 조절에 실패

- 원래는 라이트닝 토크로 파이썬 3 얘기나 하려 했으나
- 시간을 넘길 것 같아서 메인 발표로 변경

# 분량 조절에 실패

- 원래는 라이트닝 토크로 파이썬 3 얘기나 하려 했으나
- 시간을 넘길 것 같아서 메인 발표로 변경
- 이번에는 시간이 남을 것 같아서 이것저것 추가

# 분량 조절에 실패

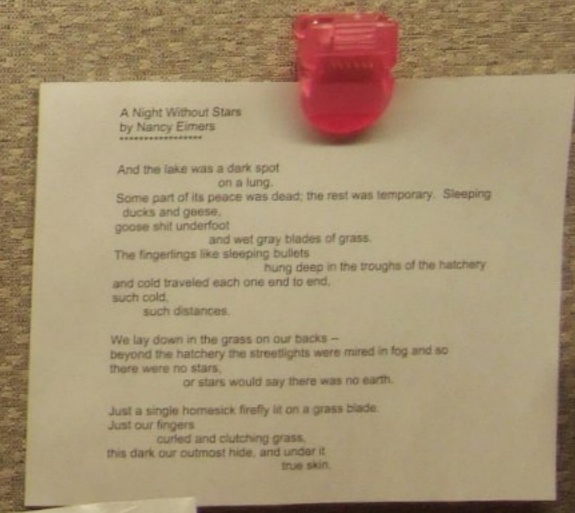
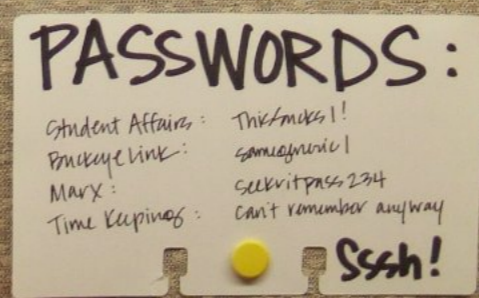
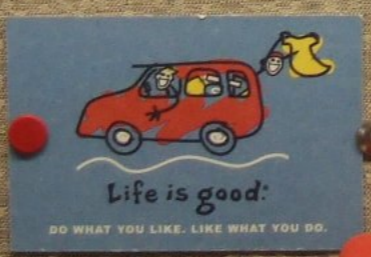
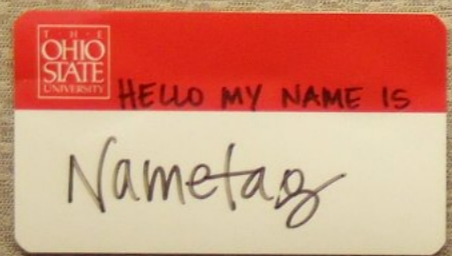
- 원래는 라이트닝 토크로 파이썬 3 얘기나 하려 했으나
- 시간을 넘길 것 같아서 메인 발표로 변경
- 이번에는 시간이 남을 것 같아서 이것저것 추가
- 발표는 하나지만 실제로는 두 개의 주제나 마찬가지

# 분량 조절에 실패

- 원래는 라이트닝 토크로 파이썬 3 얘기나 하려 했으나
- 시간을 넘길 것 같아서 메인 발표로 변경
- 이번에는 시간이 남을 것 같아서 이것저것 추가
- 발표는 하나지만 실제로는 두 개의 주제나 마찬가지
- 죄송합니다



# 1부: 지오프론트 소개



Everything will now come your way.





ADMIN / ADMIN

WEBCAM OPTIMIZED  
for low-light video chatting

Ms Puerto's Blog · Log In - Mozilla Firefox

WORDPRESS.COM

Username

Password

Remember Me

Log In

[Register](#) | [Lost your password?](#)

acer

# 팀 소유의 서버 접근 문제

# 팀 소유의 서버 접근 문제

- 보안 의식과 귀찮음 사이의 줄다리기

# 팀 소유의 서버 접근 문제

- 보안 의식과 귀찮음 사이의 줄다리기
- 대부분의 소규모 팀은 보안 의식이 거의 없다시피 하다

# 팀 소유의 서버 접근 문제

- 보안 의식과 귀찮음 사이의 줄다리기
- 대부분의 소규모 팀은 보안 의식이 거의 없다시피 하다
- 아무도 우리 같은 작은 팀의 서버를 노리지 않을 거라는 (비교적) 합리적 판단에 근거

# 팀 소유의 서버 접근 문제

- 보안 의식과 귀찮음 사이의 줄다리기
- 대부분의 소규모 팀은 보안 의식이 거의 없다시피 하다
- 아무도 우리 같은 작은 팀의 서버를 노리지 않을 거라는 (비교적) 합리적 판단에 근거
- 하지만 무차별적/기계적 공격은 팀의 규모나 비즈니스적 가치를 따지지 않는 게 함정



# 귀차니즘 대 보안 의식

# 귀차니즘 대 보안 의식

- 공유 비밀번호의 예

# 귀차니즘 대 보안 의식

- 공유 비밀번호의 예
  - 창업자 전화번호

# 귀차니즘 대 보안 의식

- 공유 비밀번호의 예
  - 창업자 전화번호
  - 창업자 이름이나 회사명을 두벌식→쿼티로 부호화 (“홍민희”→“ghdalsgml”)

# 귀차니즘 대 보안 의식

- 공유 비밀번호의 예
  - 창업자 전화번호
  - 창업자 이름이나 회사명을 두벌식→쿼티로 부호화 (“홍민희”→“ghdalsgml”)
  - 가장 안좋은 경우: “asdf”, “1234”, “1111”...

# 귀차니즘 대 보안 의식

- 공유 비밀번호의 예
  - 창업자 전화번호
  - 창업자 이름이나 회사명을 두벌식→쿼티로 부호화 (“홍민희”→“ghdalsgml”)
  - 가장 안좋은 경우: “asdf”, “1234”, “1111”...
- 차라리 비밀번호를 설정하지 말지?

# 귀차니즘 대 보안 의식

- 공유 비밀번호의 예
  - 창업자 전화번호
  - 창업자 이름이나 회사명을 두벌식→쿼티로 부호화 (“홍민희”→“ghdalsgml”)
  - 가장 안좋은 경우: “asdf”, “1234”, “1111”...
- 차라리 비밀번호를 설정하지 말지?
- 비밀번호 없애는 게 더 귀찮으니까 저렇게 하지 멍청아!

문제는 귀차니즘



# 문제는 귀차니즘

- 뛰어난 개발자들조차 귀차니즘 앞에서 굴복

# 문제는 귀차니즘

- 뛰어난 개발자들조차 귀차니즘 앞에서 굴복
- 대개 몰라서 못하는 게 아니라 알지만 귀찮아서 안하는 것

# 문제는 귀차니즘

- 뛰어난 개발자들조차 귀차니즘 앞에서 굴복
- 대개 몰라서 못하는 게 아니라 알지만 귀찮아서 안하는 것
- 보안성이 높아도 귀찮은 시스템은 선택되지 않음

# 문제는 귀차니즘

- 뛰어난 개발자들조차 귀차니즘 앞에서 굴복
- 대개 몰라서 못하는 게 아니라 알지만 귀찮아서 안하는 것
- 보안성이 높아도 귀찮은 시스템은 선택되지 않음
- 최대한 덜 귀찮게 하는 것이 그나마 보안성을 높이는 데 효과적

# 공유 인증키 방식

A

B

C

D

authoriz  
ed\_keys

authoriz  
ed\_keys

authoriz  
ed\_keys

authoriz  
ed\_keys

# 공유 인증키 방식

A

B

C

D

id\_rsa

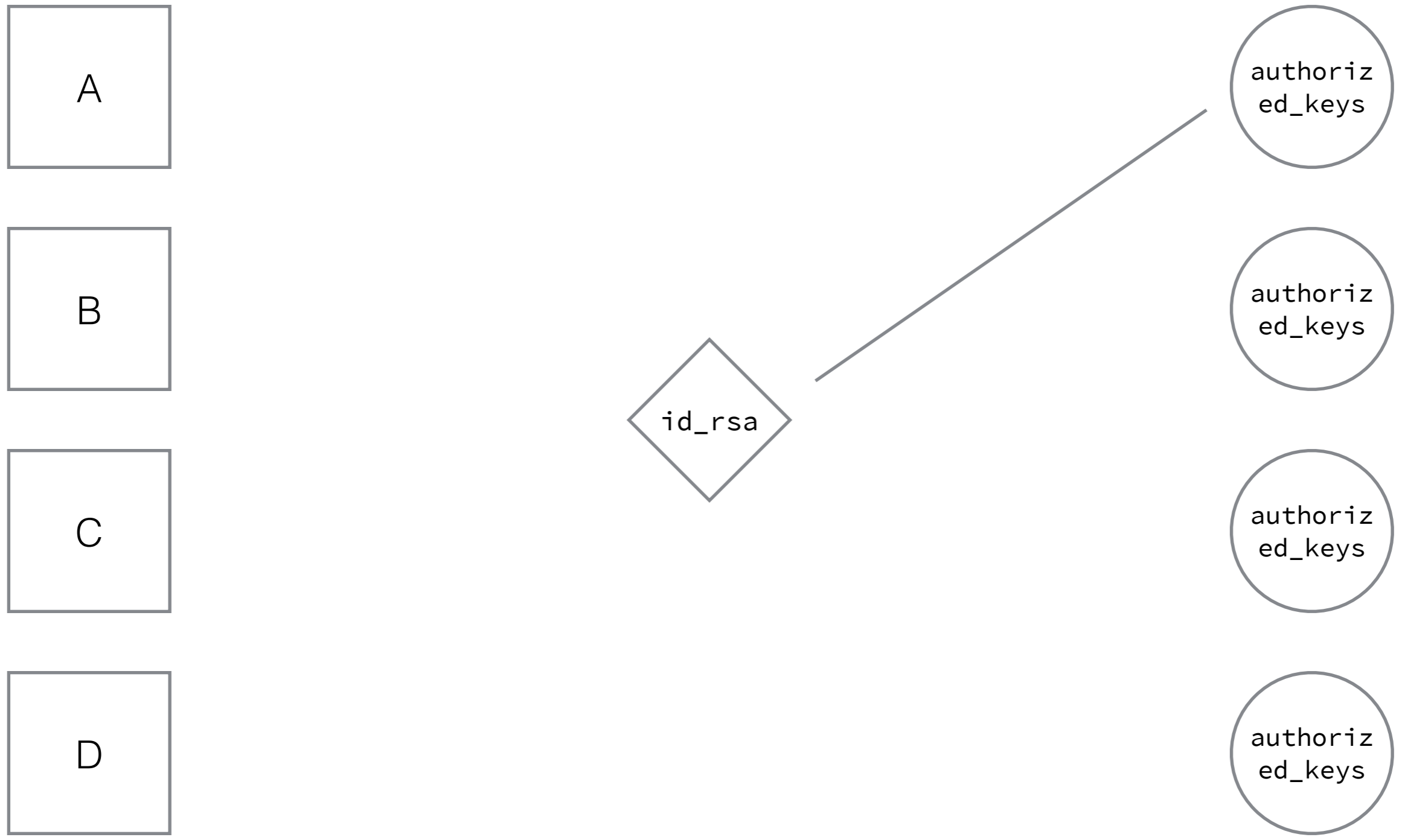
authoriz  
ed\_keys

authoriz  
ed\_keys

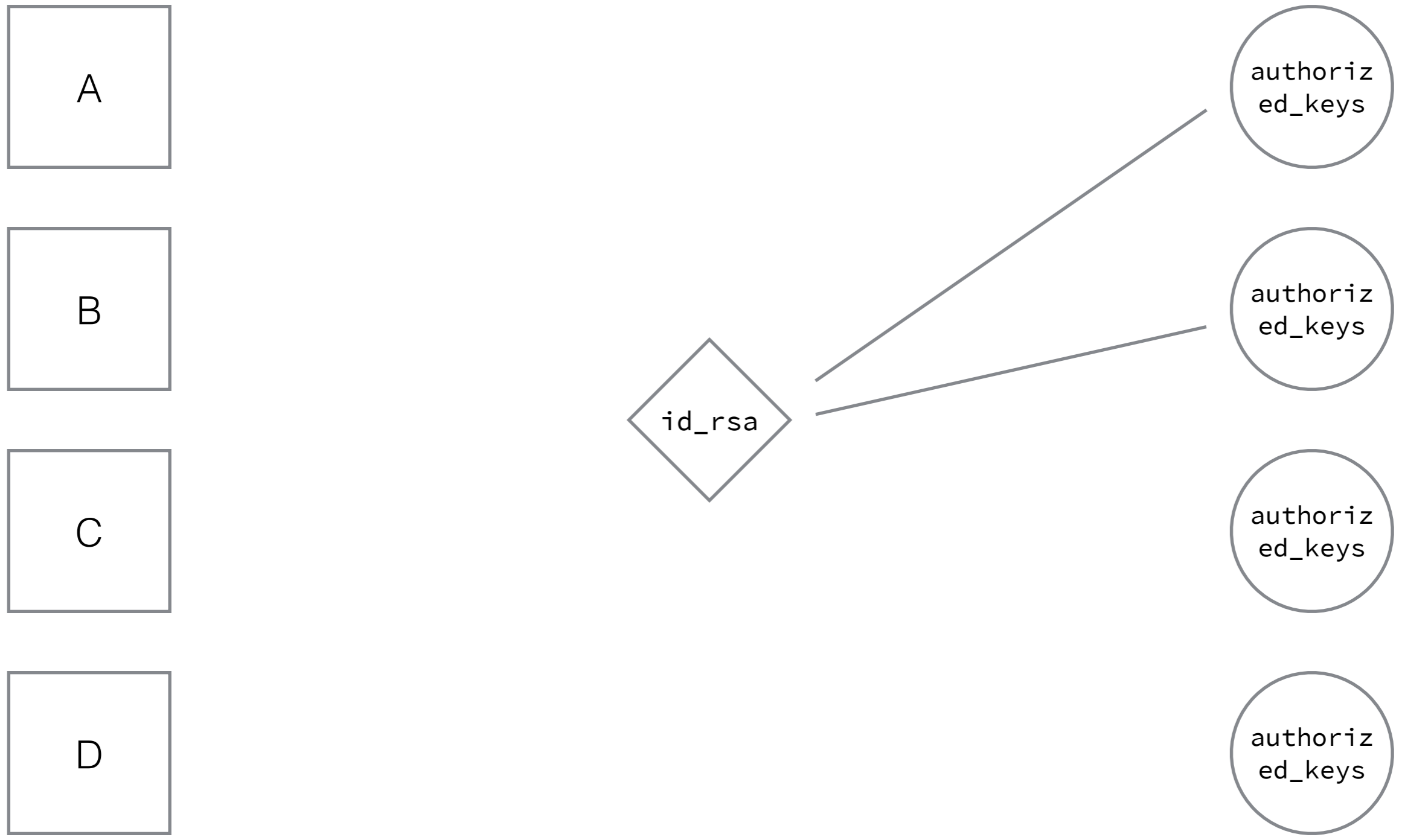
authoriz  
ed\_keys

authoriz  
ed\_keys

# 공유 인증키 방식

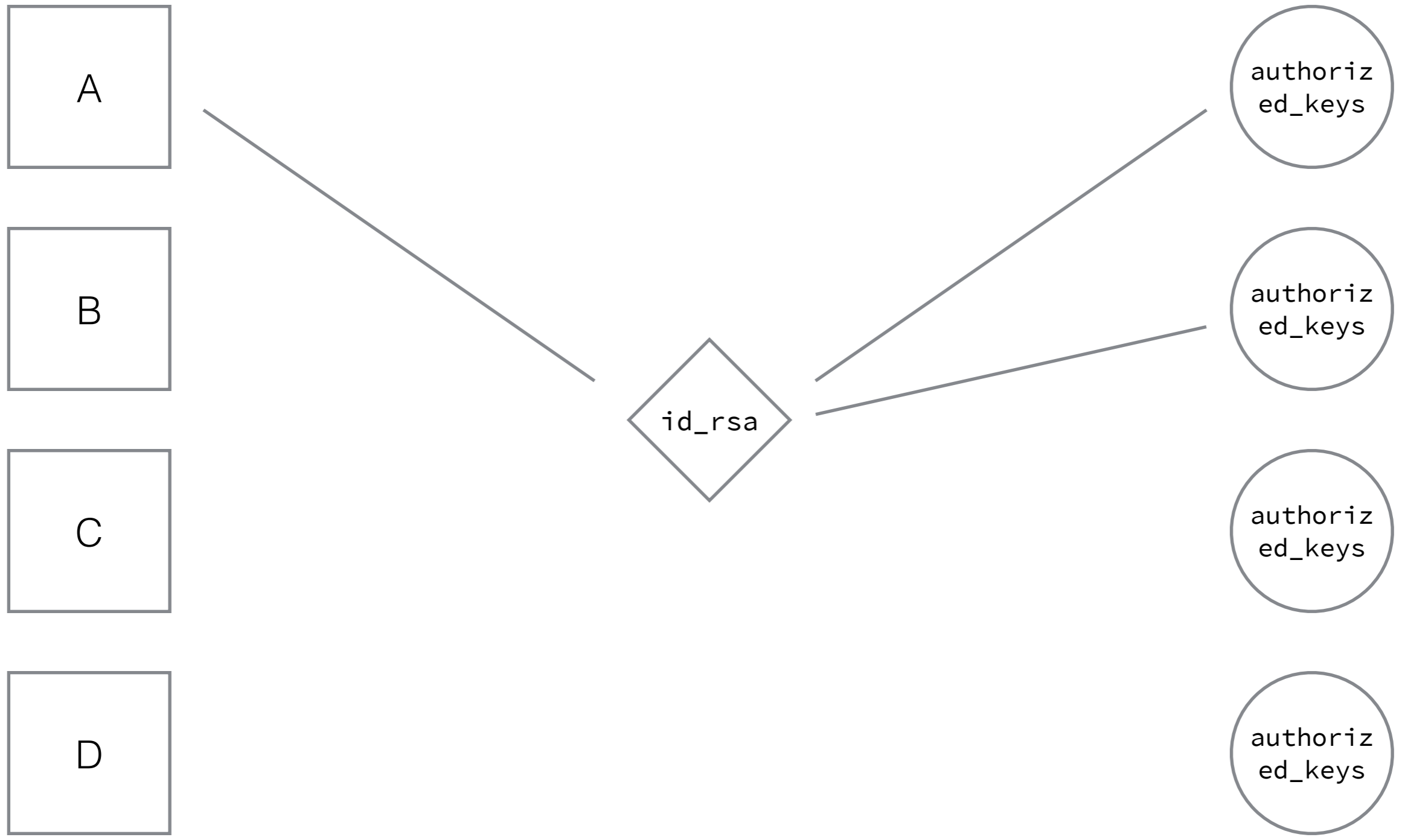


# 공유 인증키 방식

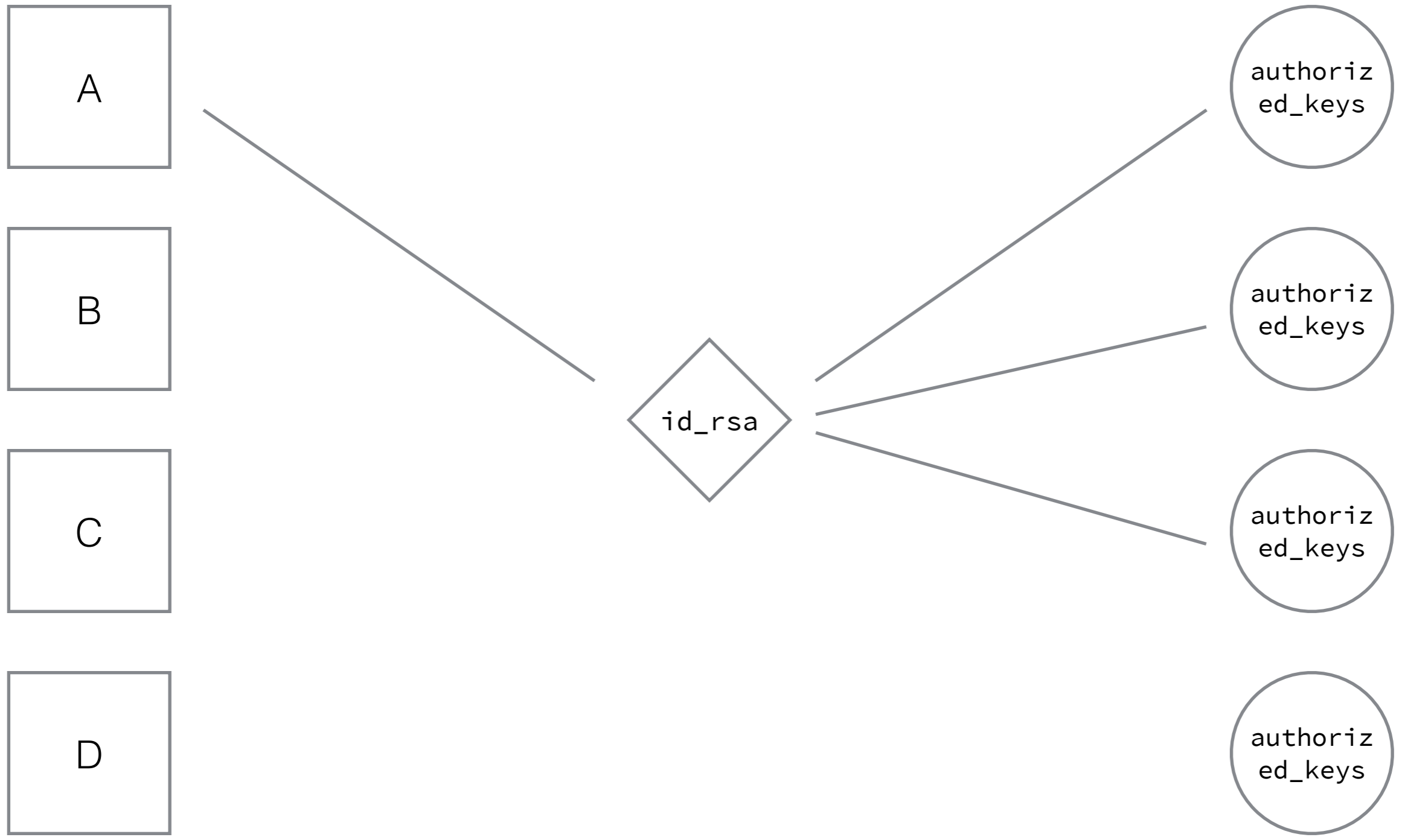




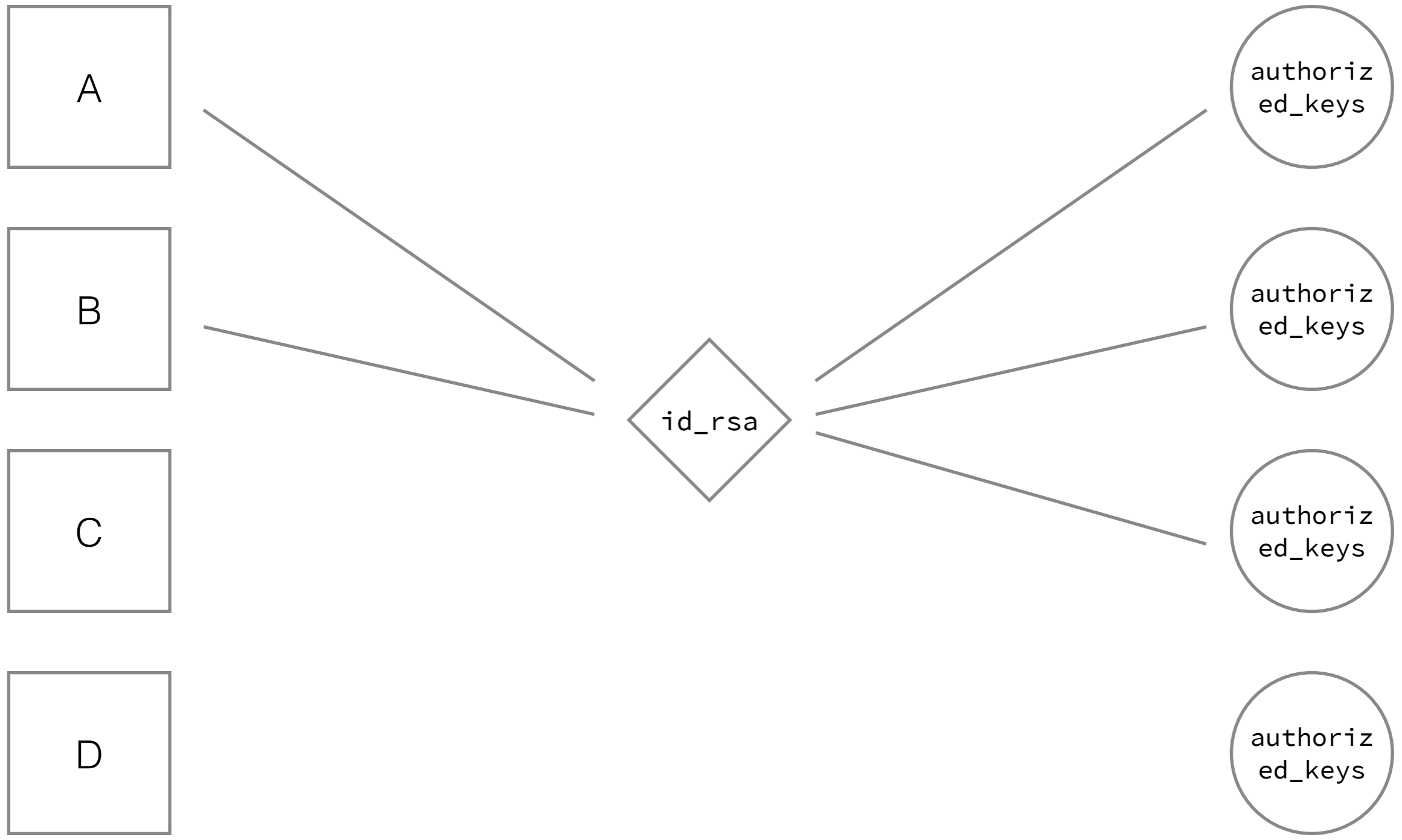
# 공유 인증키 방식



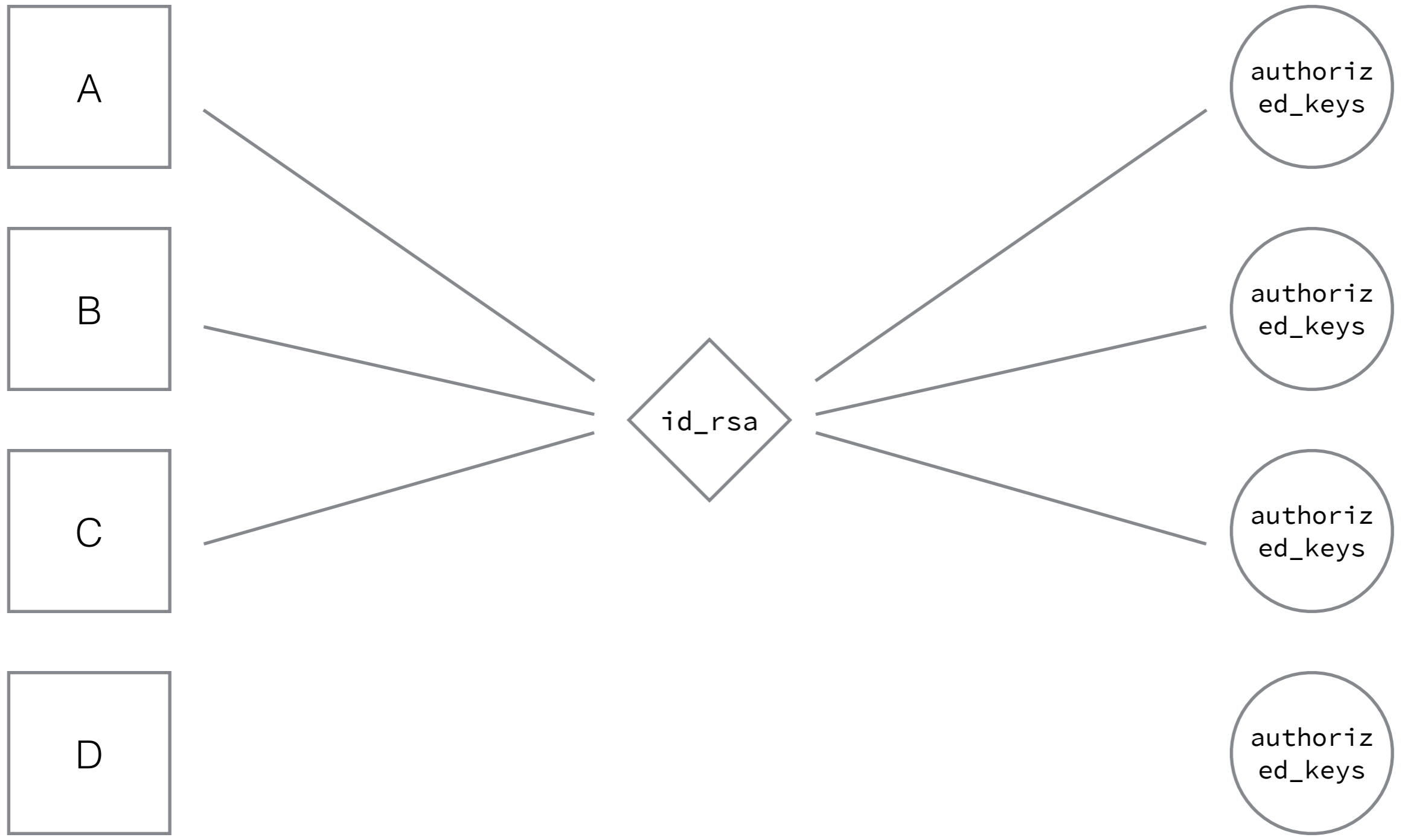
# 공유 인증키 방식



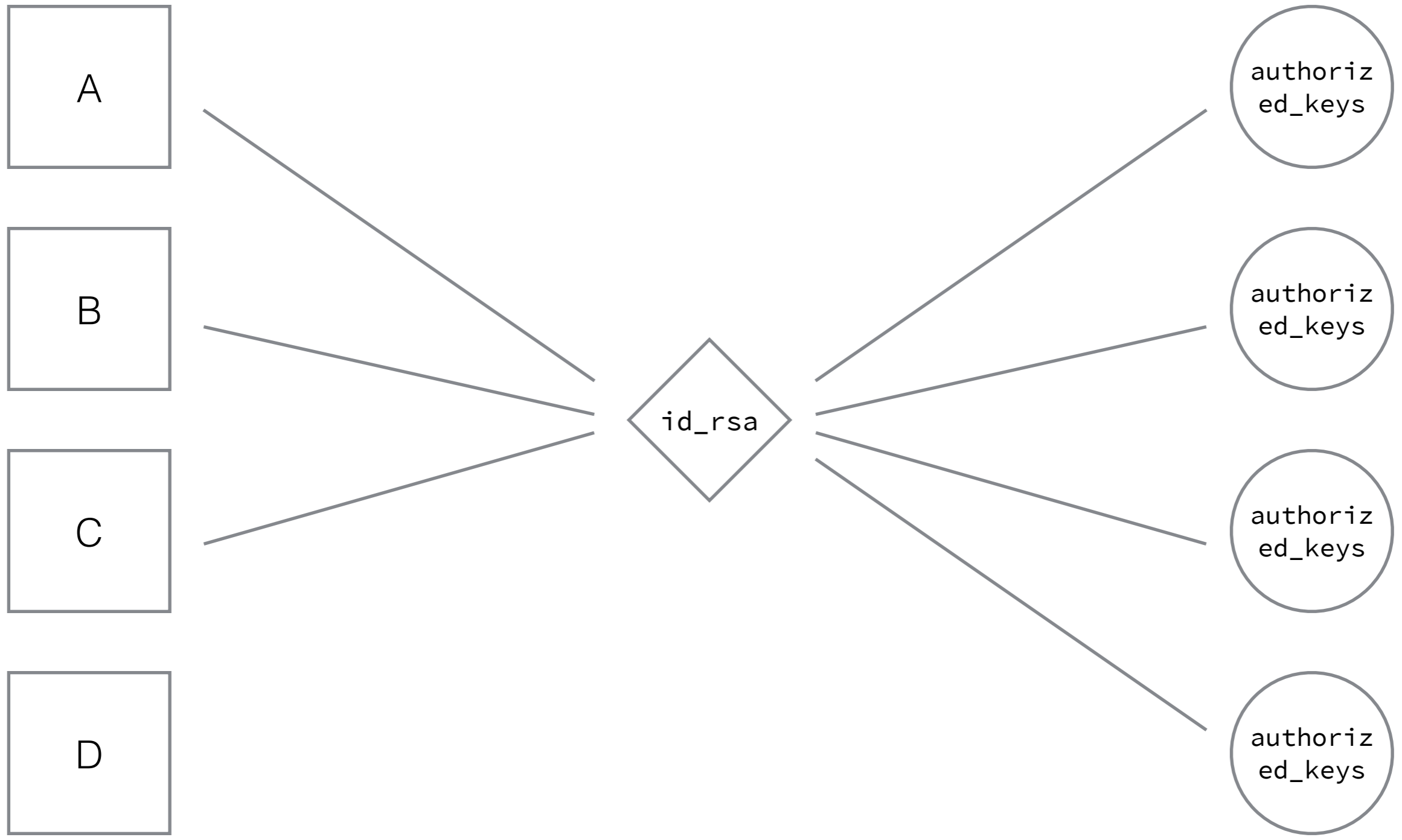
# 공유 인증키 방식



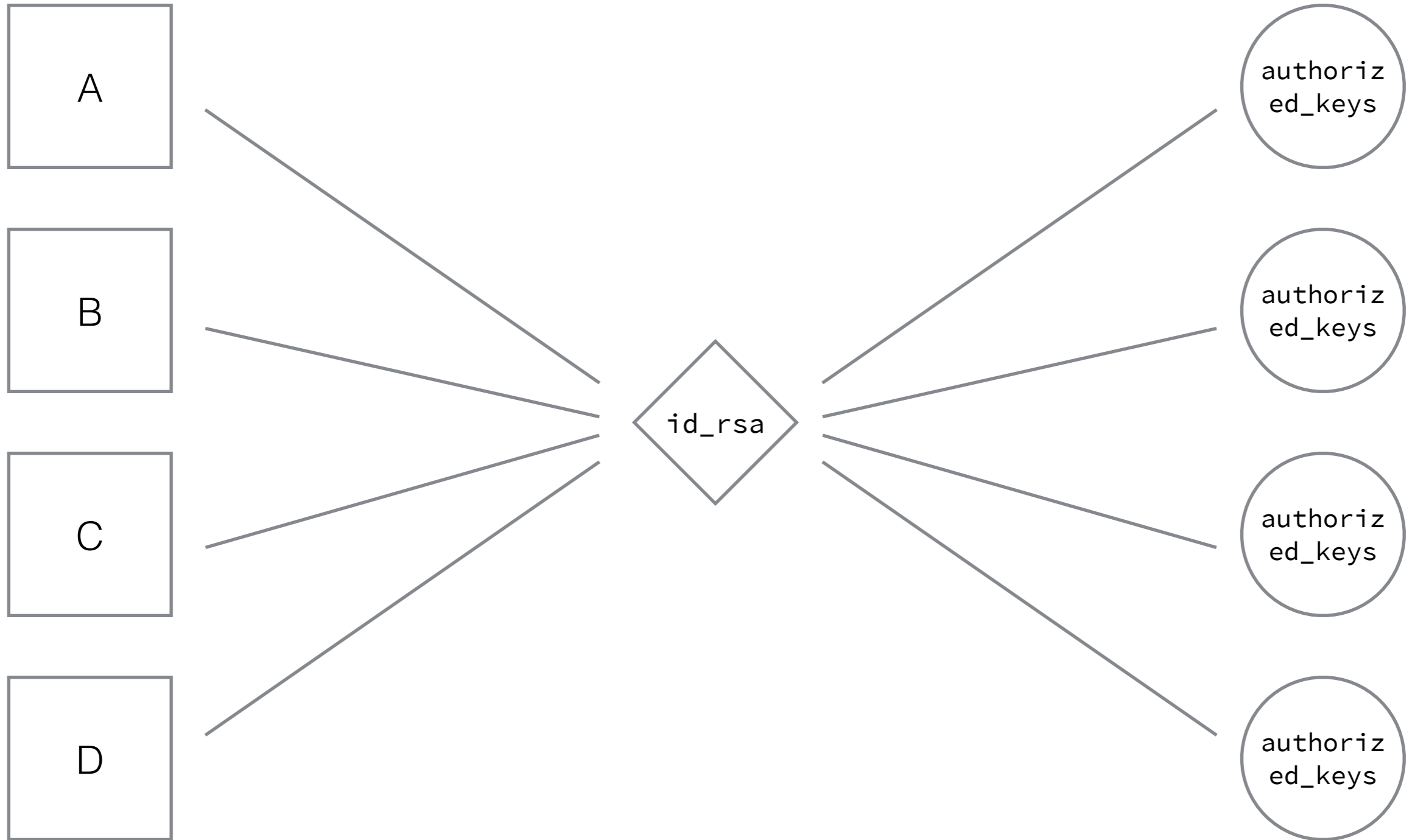
# 공유 인증키 방식



# 공유 인증키 방식



# 공유 인증키 방식



# 공유 인증키 방식

# 공유 인증키 방식

- 아주 작은 팀에서 자주 쓰이는 방식



# 공유 인증키 방식

- 아주 작은 팀에서 자주 쓰이는 방식
- 당연히(?) 패스프레이즈는 없다!

# 공유 인증키 방식

- 아주 작은 팀에서 자주 쓰이는 방식
- 당연히(?) 패스프레이즈는 없다!
- 왜냐하면...

# 공유 인증키 방식

- 아주 작은 팀에서 자주 쓰이는 방식
- 당연히(?) 패스프레이즈는 없다!
- 왜냐하면...
- 비밀번호 입력하는 것도 귀찮아서 쓰는 거니까!

# 공유 인증키 방식

- 아주 작은 팀에서 자주 쓰이는 방식
- 당연히(?) 패스프레이즈는 없다!
- 왜냐하면...
- 비밀번호 입력하는 것도 귀찮아서 쓰는 거니까!
- 나가는 사람이 있을 때마다 새 인증키를 만들어야 하지만  
그딴 귀찮은 짓을 할 리가...



SPOQA > Keys

- 개인용
- SPOQA
- 사진
- 공유
- 링크
- 이벤트

이름 ▲	유형	수정된 날짜
id_rsa	파일	2012/12/7 오후 4:02일
id_rsa.pub	파일	2012/12/7 오후 4:02일



SPOQA > Keys

- 개인용
- SPOQA
- 사진
- 공유
- 링크
- 이벤트

이름 ▲	유형	수정된 날짜
id_rsa	파일	2012/12/7 오후 4:02일
id_rsa.pub	파일	2012/12/7 오후 4:02일

※주: 연출입니다.



SPOQA > Keys

- 개인용
- SPOQA
- 사진
- 공유
- 링크
- 이벤트

이름 ▲	유형	수정된 날짜
id_rsa	파일	2012/12/7 오후 4:02일
id_rsa.pub	파일	2012/12/7 오후 4:02일

※주: 연출입니다.

스포카는 공유 인증키를 사용하지 않으며, 공유 인증키를 Dropbox에 공유한 적도 없습니다. 적어도 제가 입사한 이후에는...

# 각자 인증키 방식



# 각자 인증키 방식

- DVCS의 성공으로 전보다 훨씬 많이 보급된 상황

# 각자 인증키 방식

- DVCS의 성공으로 전보다 훨씬 많이 보급된 상황
- 주요 저장소 서비스에서 인증을 위해 사용하기 때문

# 각자 인증키 방식

- DVCS의 성공으로 전보다 훨씬 많이 보급된 상황
- 주요 저장소 서비스에서 인증을 위해 사용하기 때문
- 각자 인증키를 어차피 가지고 있으니 그걸로 인증하자!

# 각자 인증키 방식

- DVCS의 성공으로 전보다 훨씬 많이 보급된 상황
- 주요 저장소 서비스에서 인증을 위해 사용하기 때문
- 각자 인증키를 어차피 가지고 있으니 그걸로 인증하자!
- `authorized_keys`

# 각자 인증키 방식

- DVCS의 성공으로 전보다 훨씬 많이 보급된 상황
- 주요 저장소 서비스에서 인증을 위해 사용하기 때문
- 각자 인증키를 어차피 가지고 있으니 그걸로 인증하자!
- `authorized_keys`
- `ssh-copy-id`

# 각자 인증키 방식

A

B

C

D

authoriz  
ed\_keys

authoriz  
ed\_keys

authoriz  
ed\_keys

authoriz  
ed\_keys

# 각자 인증키 방식

A

B

C

D

id\_rsa

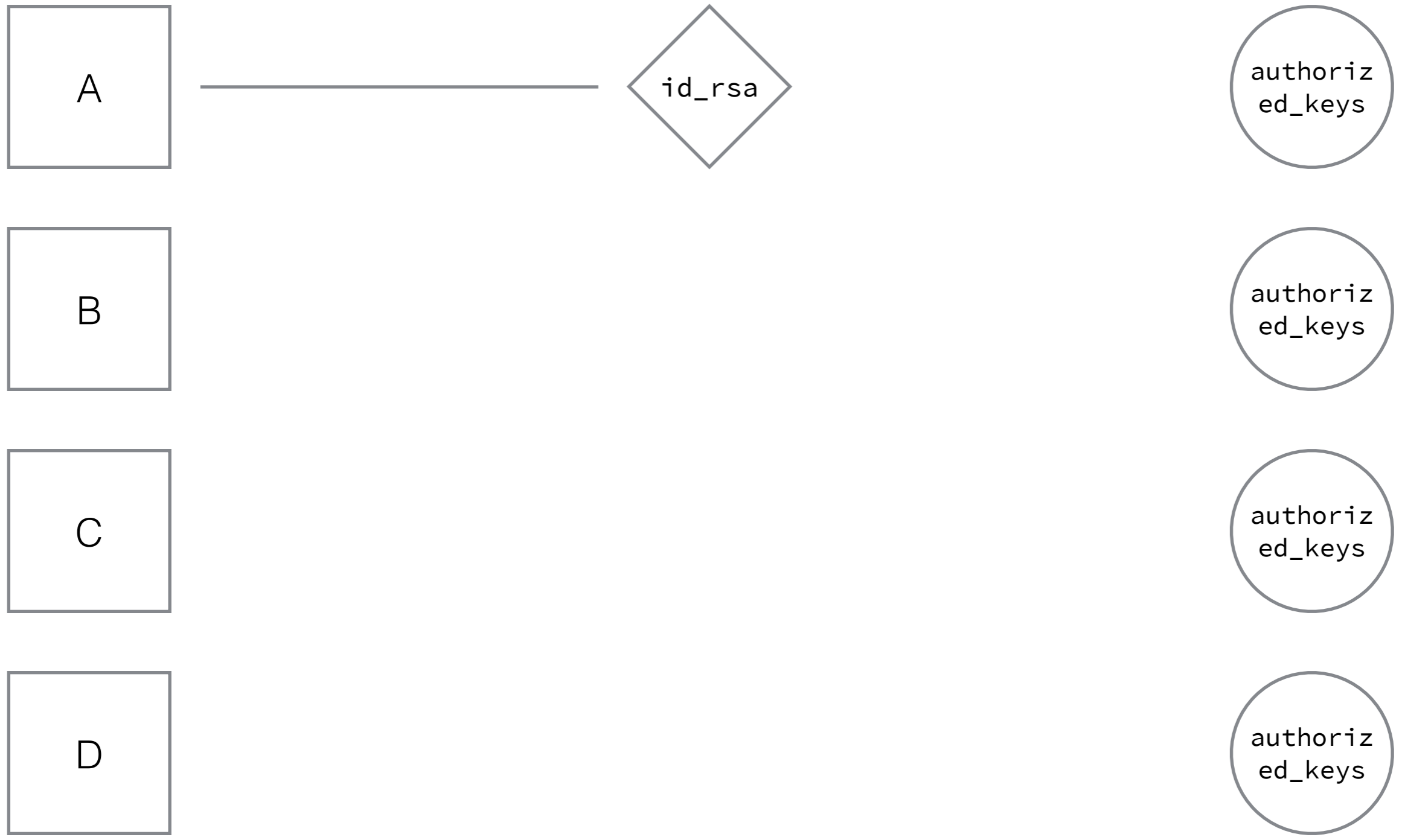
authorized\_keys

authorized\_keys

authorized\_keys

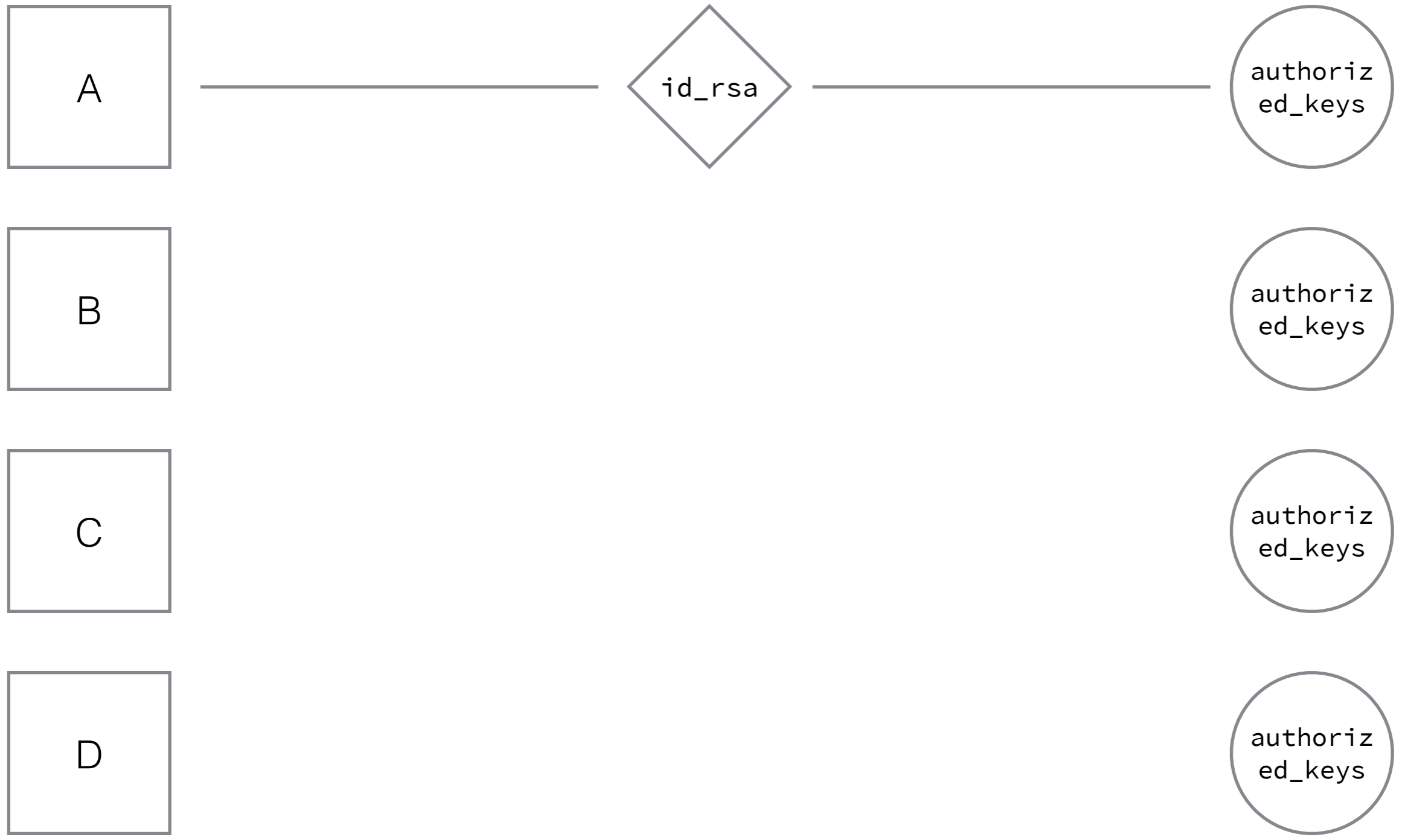
authorized\_keys

# 각자 인증키 방식

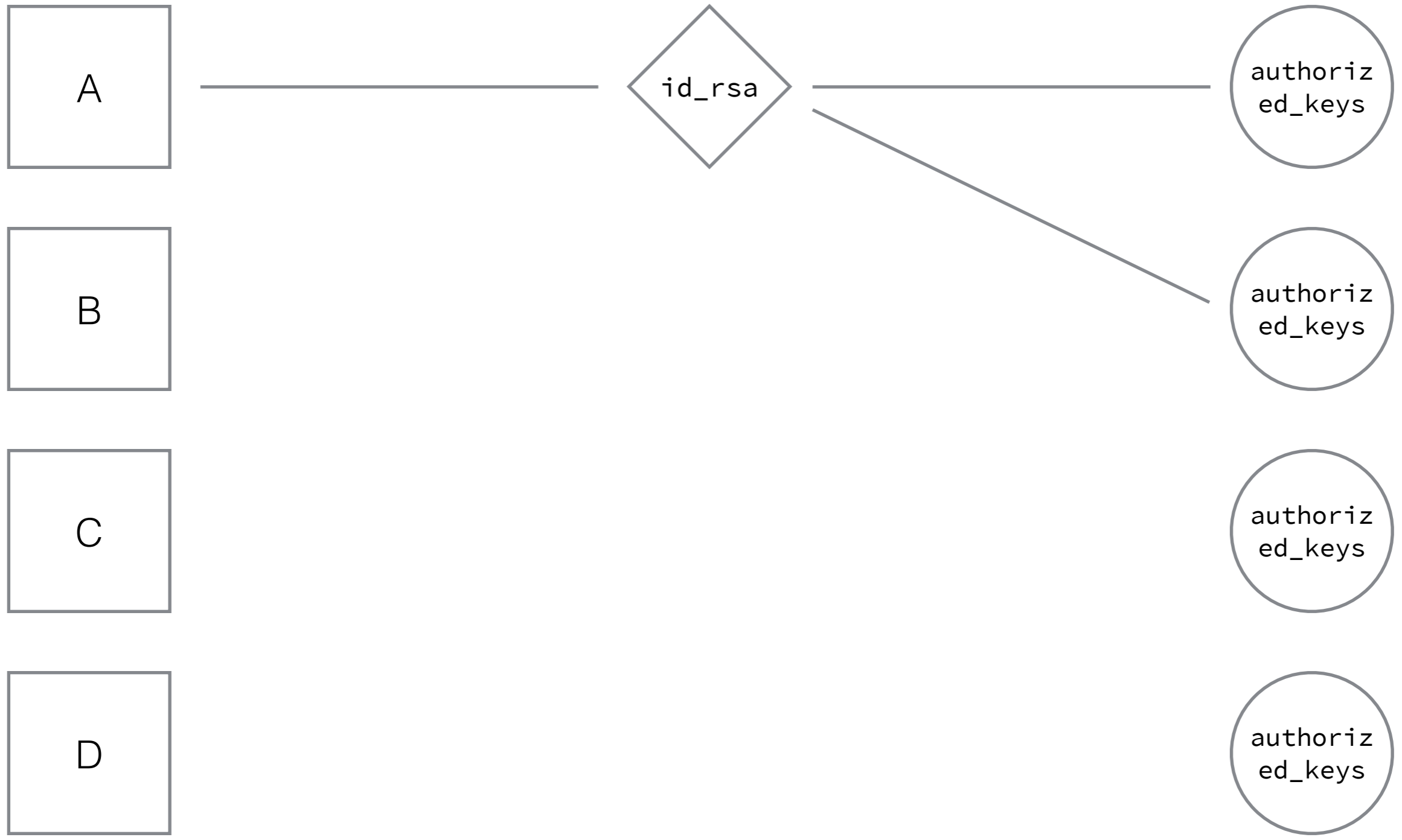




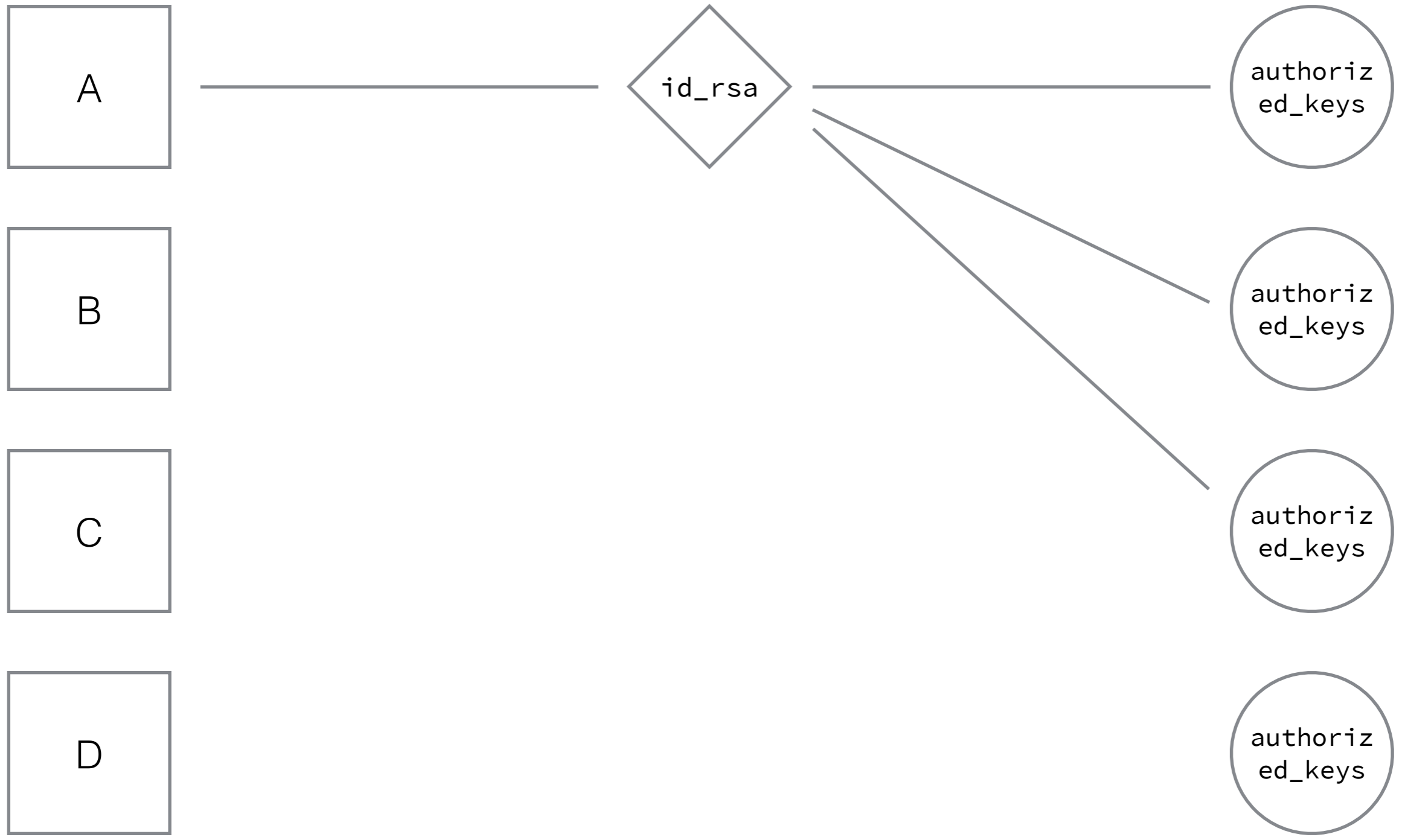
# 각자 인증키 방식



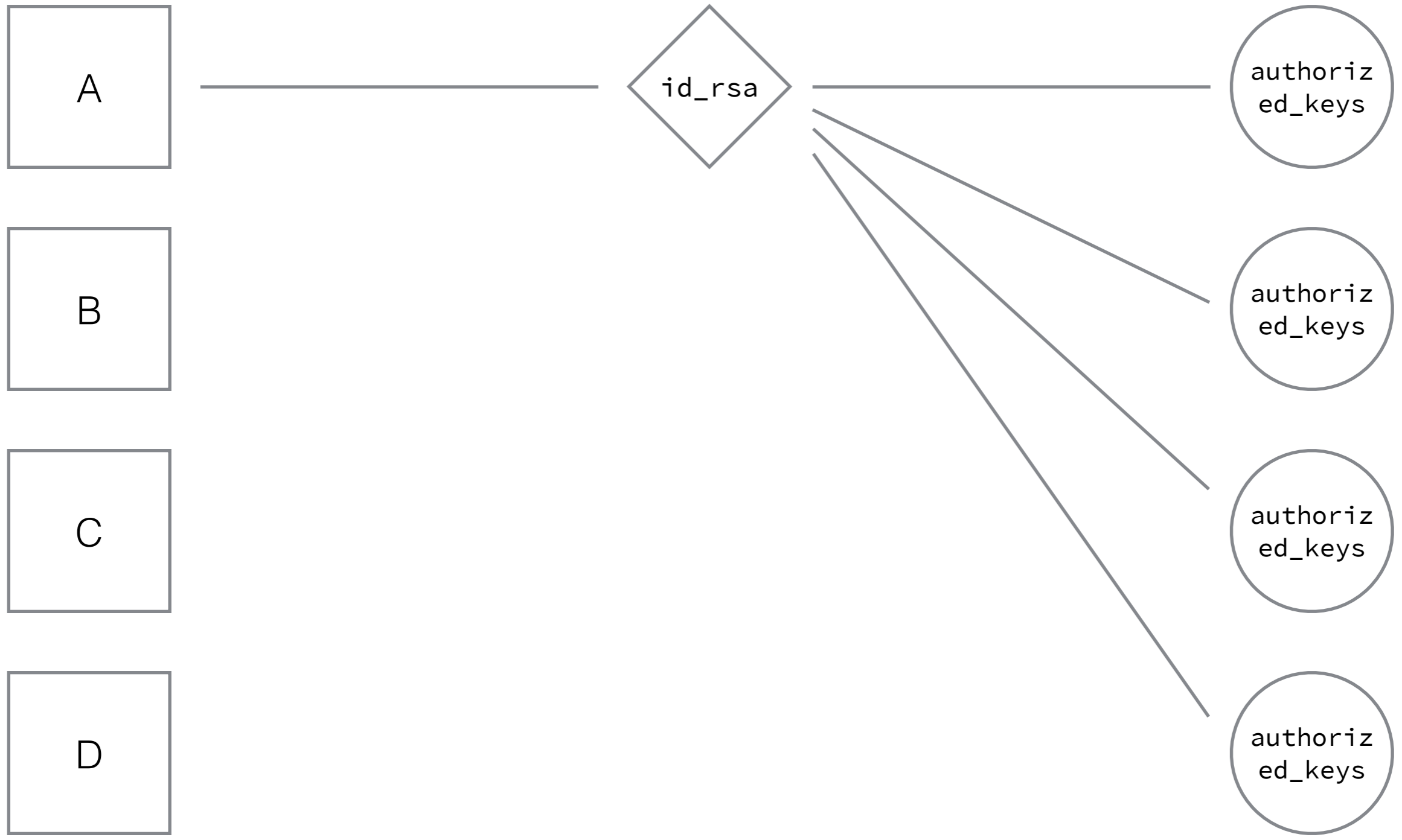
# 각자 인증키 방식



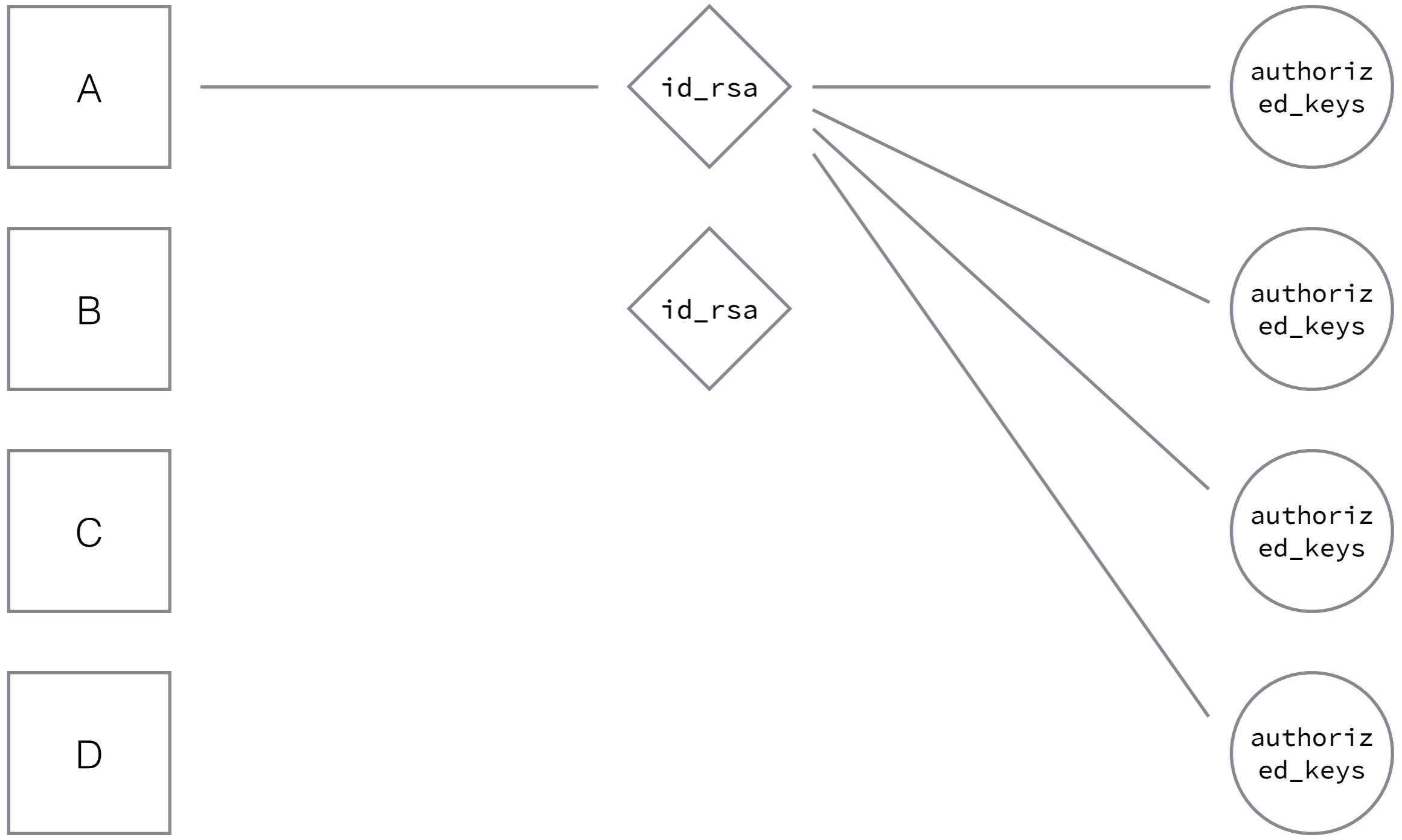
# 각자 인증키 방식



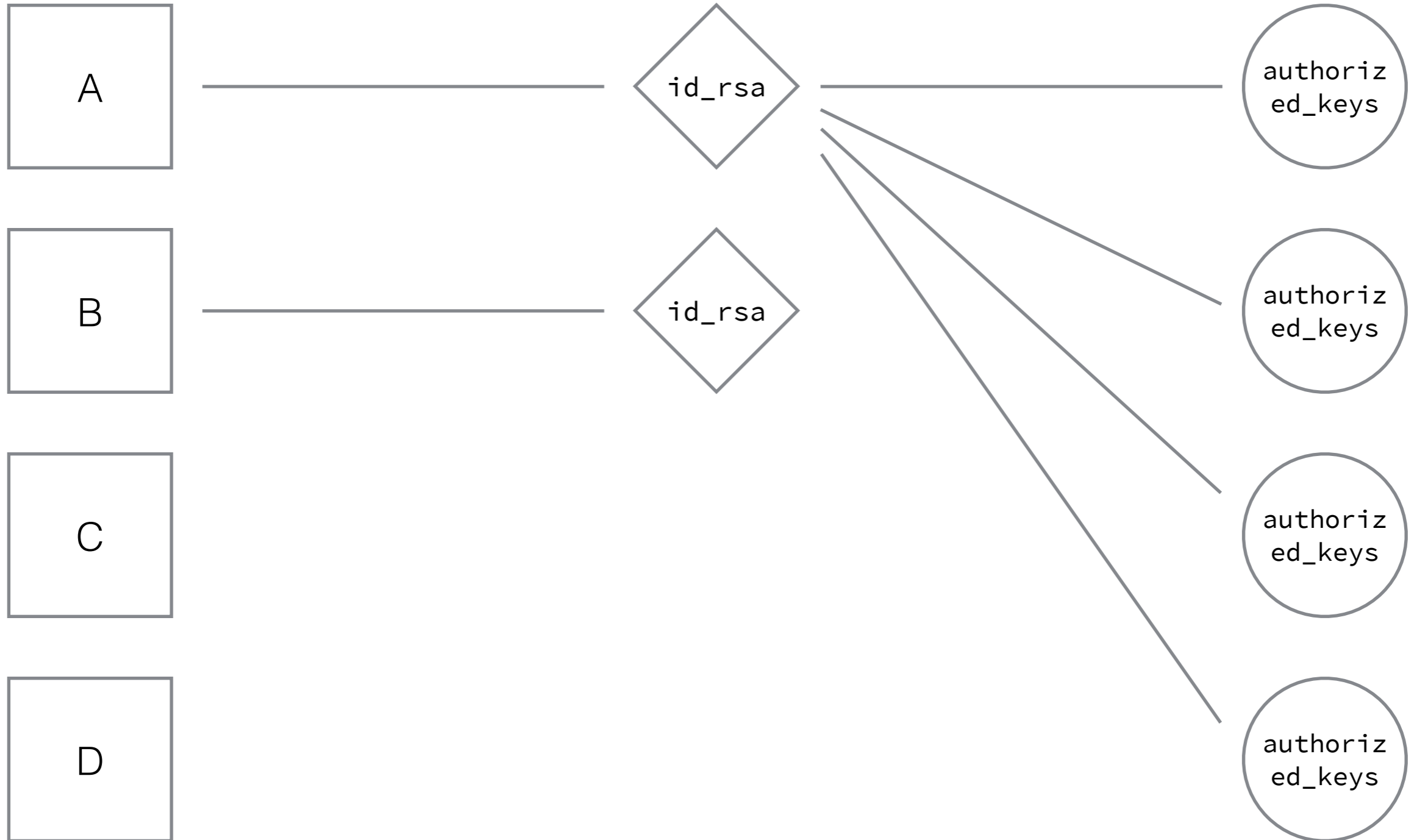
# 각자 인증키 방식



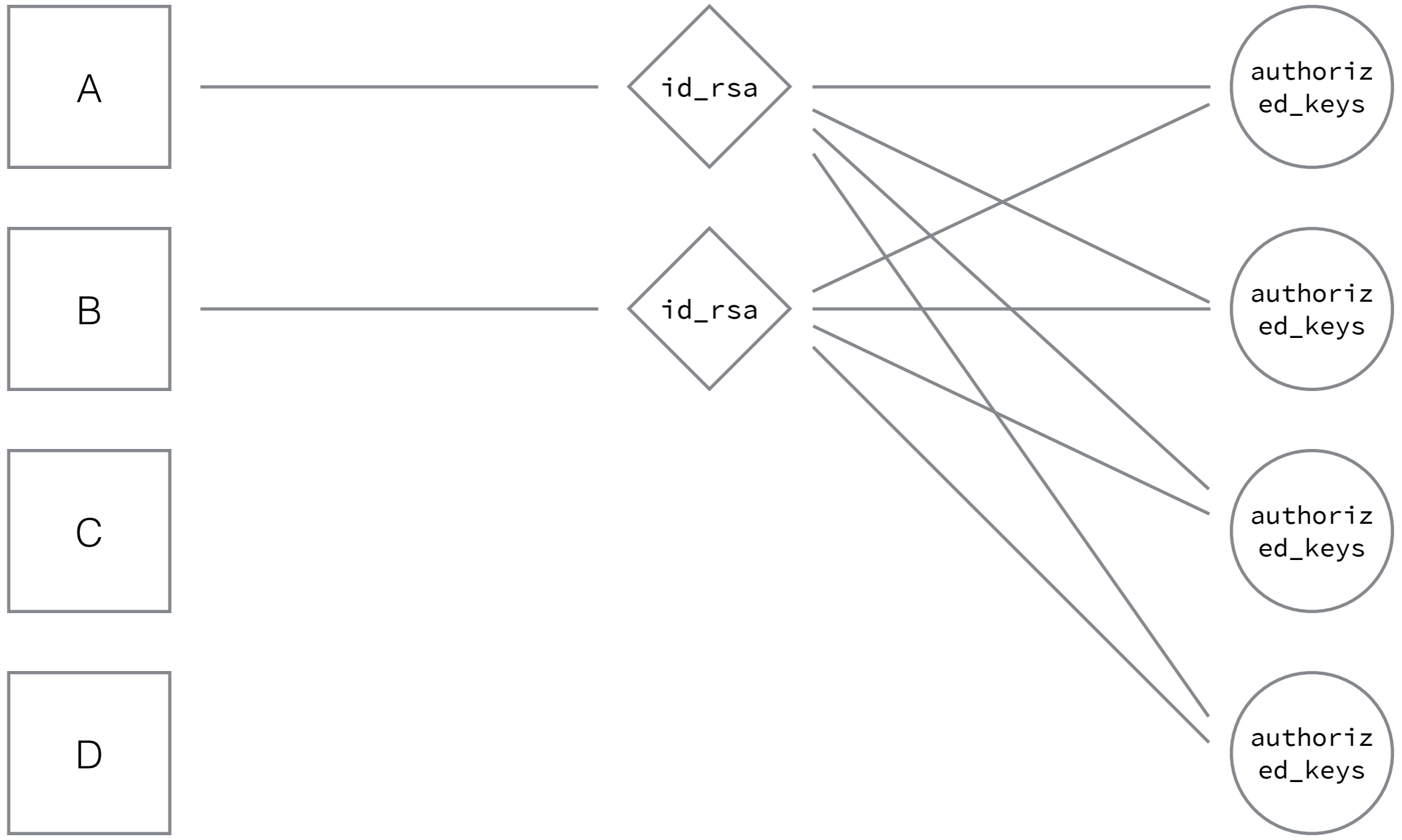
# 각자 인증키 방식



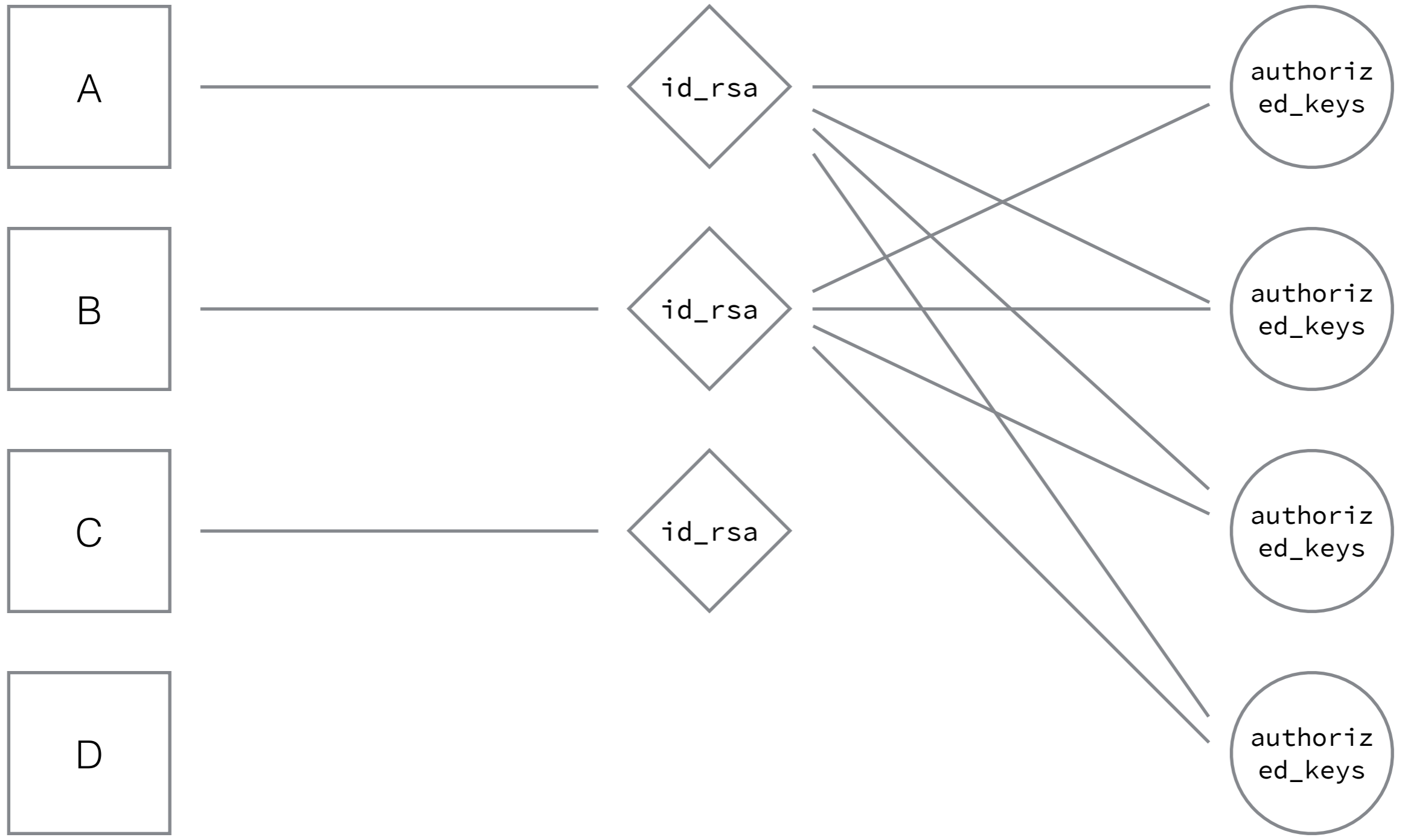
# 각자 인증키 방식



# 각자 인증키 방식

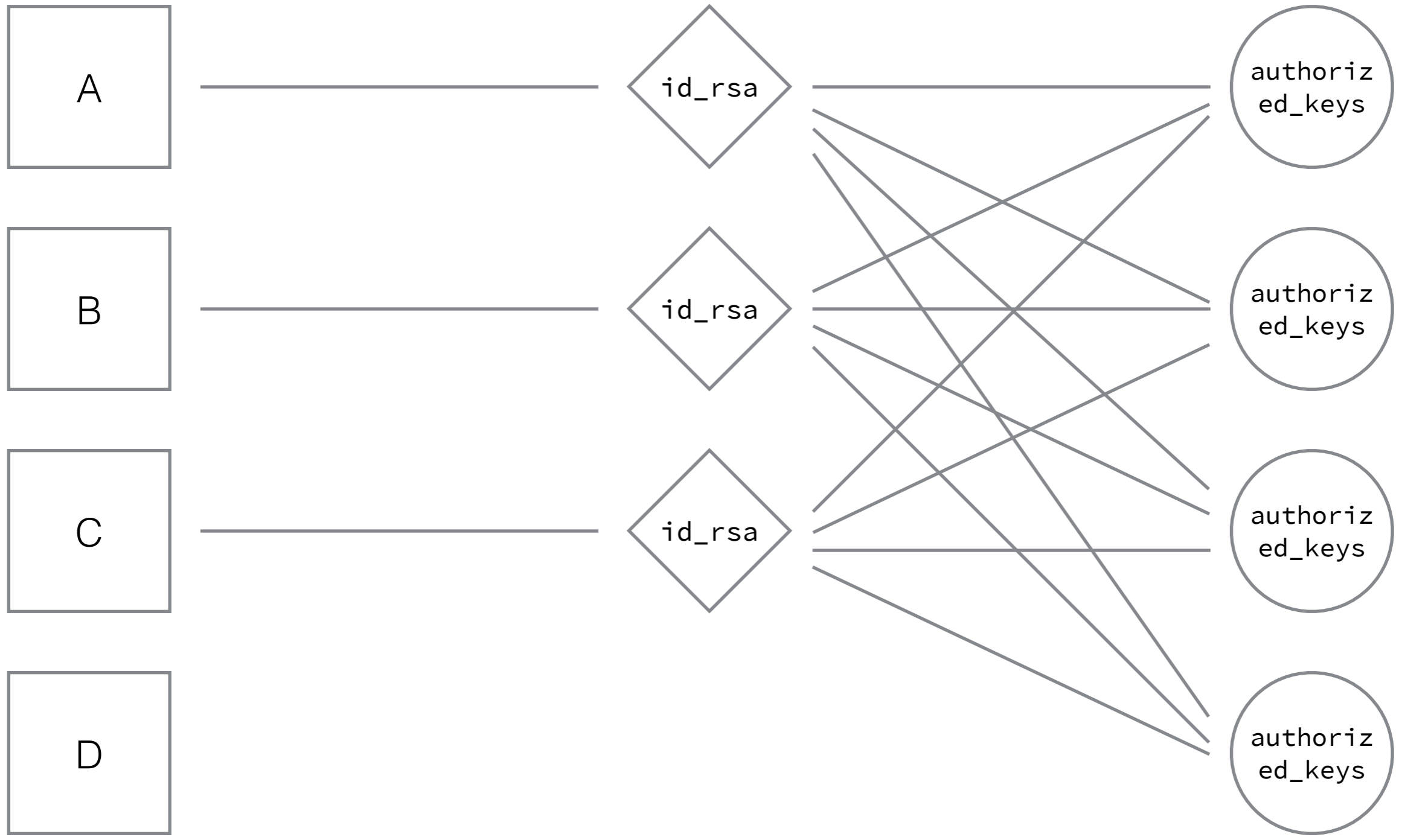


# 각자 인증키 방식

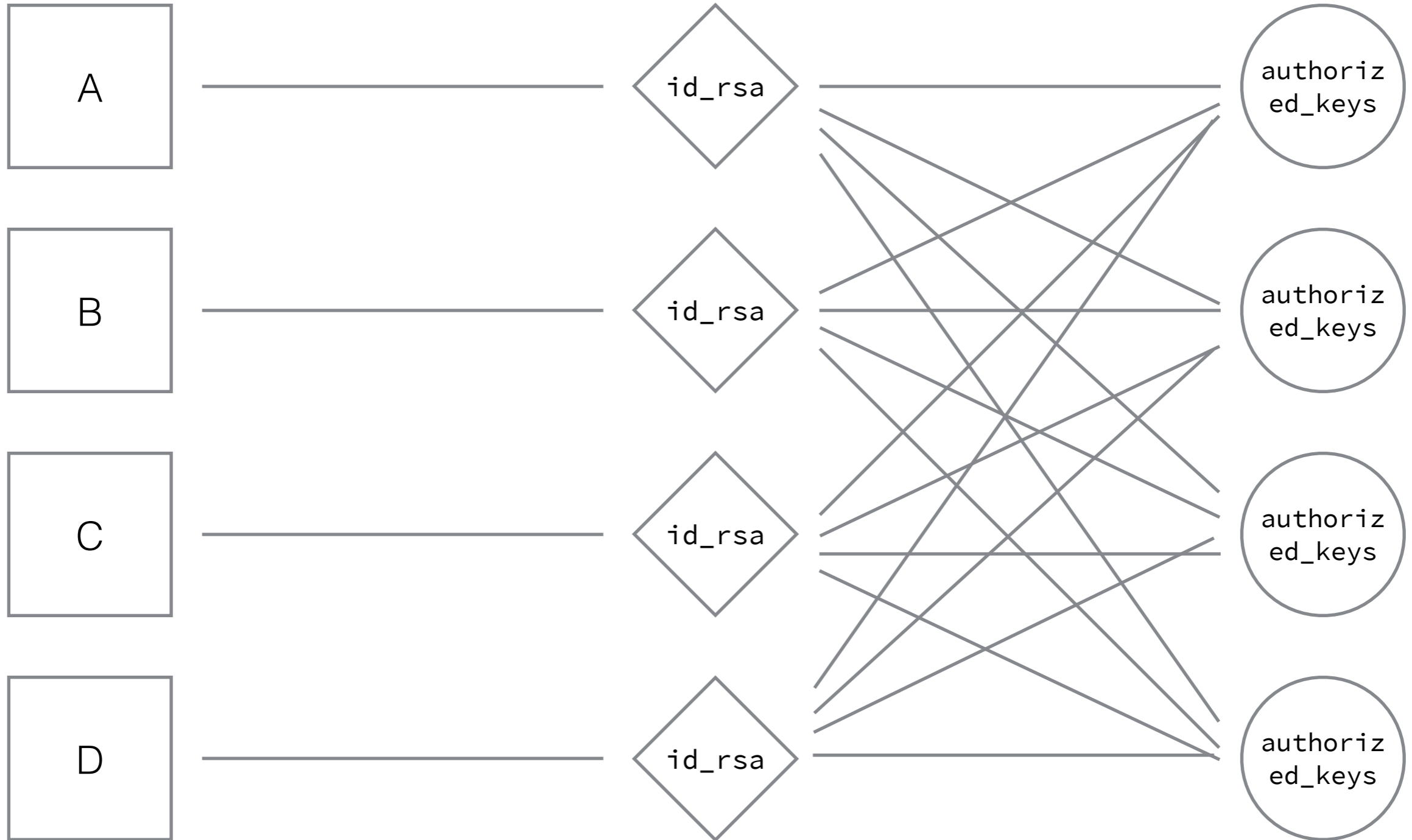




# 각자 인증키 방식



# 각자 인증키 방식



# 각자 인증키 방식

# 각자 인증키 방식

- 팀원이 새로 들어오면 어찌지?

# 각자 인증키 방식

- 팀원이 새로 들어오면 어찌지?
- 팀원이 나가면?

# 각자 인증키 방식

- 팀원이 새로 들어오면 어찌지?
- 팀원이 나가면?
- 원격지 수만큼 수정

# 각자 인증키 방식

- 팀원이 새로 들어오면 어찌지?
- 팀원이 나가면?
- 원격지 수만큼 수정
- 새로 들어온 사람의 인증키를 원격지마다 추가하는 것은 피할 수 없긴 해도....

# 각자 인증키 방식

- 팀원이 새로 들어오면 어찌지?
- 팀원이 나가면?
- 원격지 수만큼 수정
- 새로 들어온 사람의 인증키를 원격지마다 추가하는 것은 피할 수 없긴 해도...
- 나간 사람 빼는 건 귀찮으니까 나중에



# 각자 인증키 방식

# 각자 인증키 방식

- 새로운 사람, 나간 사람 생길 때마다 모든 원격지의 `authorized_keys` 수정해야 함

# 각자 인증키 방식

- 새로운 사람, 나간 사람 생길 때마다 모든 원격지의 `authorized_keys` 수정해야 함
- 근데 누가 추가하고 제거하지?

# 각자 인증키 방식

- 새로운 사람, 나간 사람 생길 때마다 모든 원격지의 `authorized_keys` 수정해야 함
- 근데 누가 추가하고 제거하지?
- “홍 님 인증키 추가좀 부탁드립니다.”

# 각자 인증키 방식

- 새로운 사람, 나간 사람 생길 때마다 모든 원격지의 `authorized_keys` 수정해야 함
- 근데 누가 추가하고 제거하지?
- “홍 님 인증키 추가좀 부탁드립니다.”
- A + 내가 해야함?



"아무것도 안하고 싶다"



"이미 아무것도 안하고 있지만 더 격렬  
하고 적극적으로 아무것도 안하고 싶다"

고민



# 고민

- 새로 온 사람 공개키를 매번 보내달라고 하기 귀찮다

# 고민

- 새로 온 사람 공개키를 매번 보내달라고 하기 귀찮다
- 사람 나갈 때 그 사람 공개키 빼야 한다는 거 자꾸 까먹는다

# 고민

- 새로 온 사람 공개키를 매번 보내달라고 하기 귀찮다
- 사람 나갈 때 그 사람 공개키 빼야 한다는 거 자꾸 까먹는다
- 그냥 한번 자동화 해놓고 나는 아무 짓도 안해도 돌아갔음 좋겠다

# 아이디어

# 아이디어

- 어, 근데 새로 온 사람도 깃헙 계정은 있지 않나?

# 아이디어

- 어, 근데 새로 온 사람도 깃헙 계정은 있지 않나?
- 깃헙에 원래 각자 공개키는 올려두고 쓰지 않나?

# 아이디어

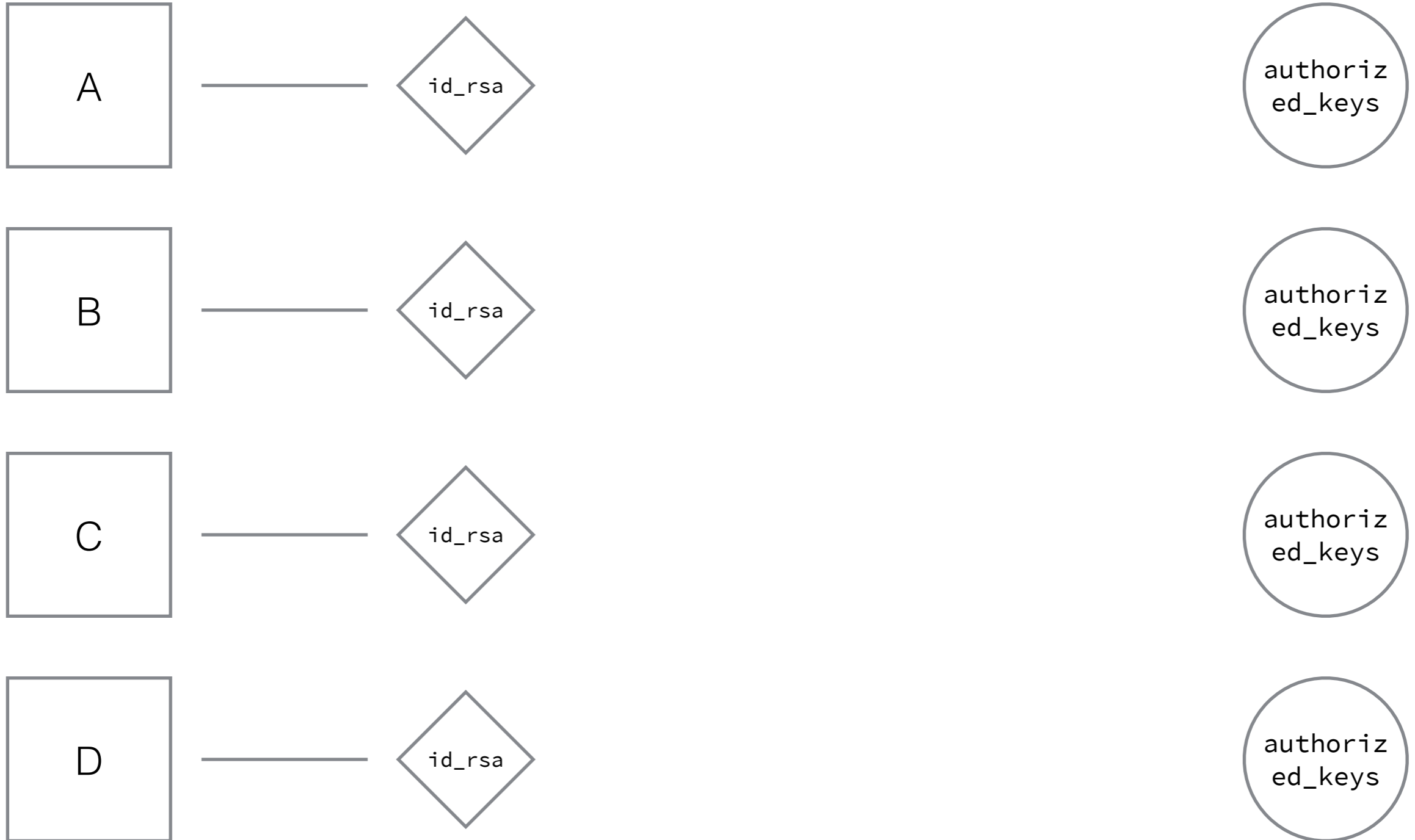
- 어, 근데 새로 온 사람도 깃헙 계정은 있지 않나?
- 깃헙에 원래 각자 공개키는 올려두고 쓰지 않나?
- 참, 우리 회사도 깃헙 쓰지?

# 아이디어

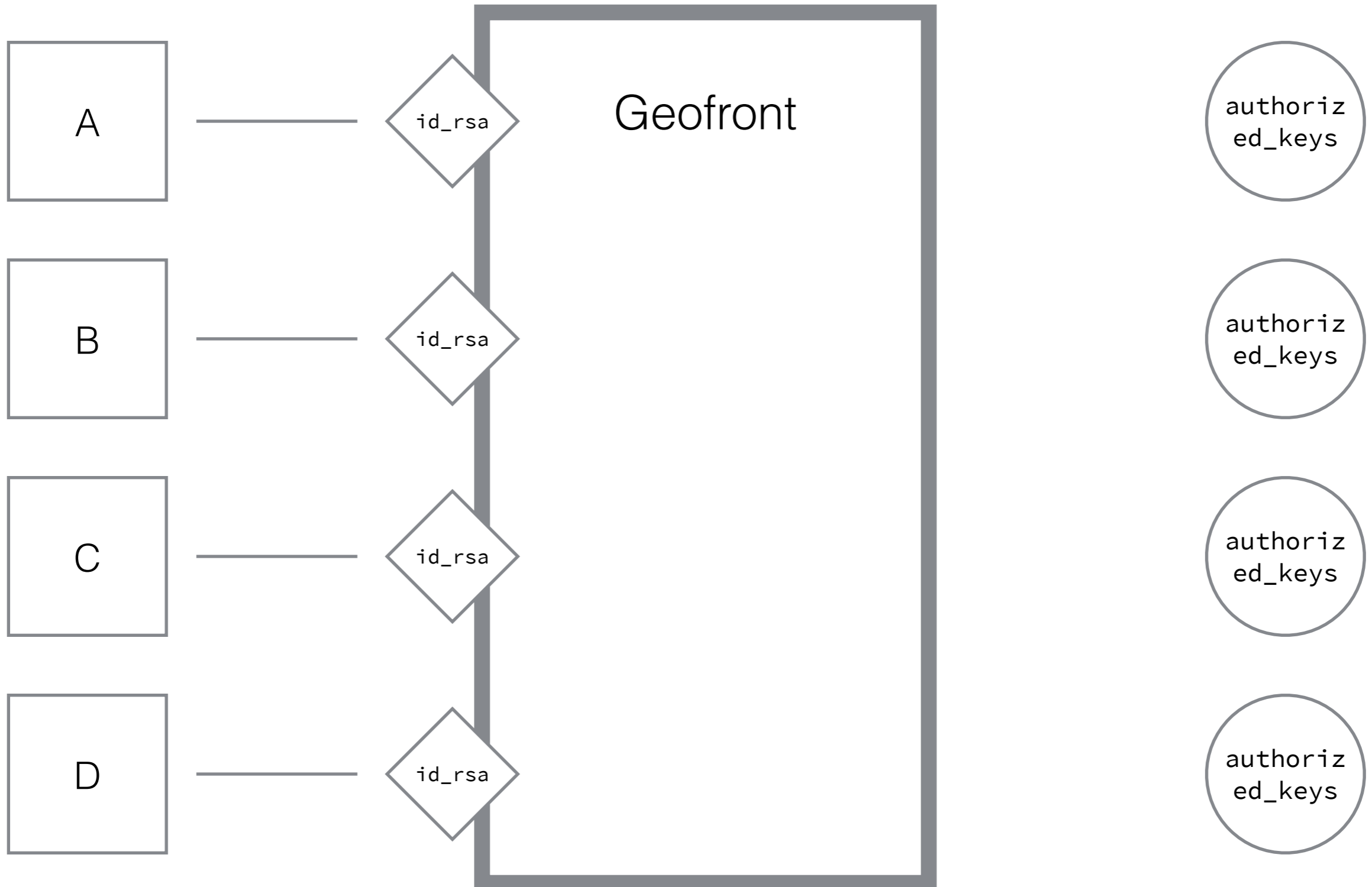
- 어, 근데 새로 온 사람도 깃헙 계정은 있지 않나?
- 깃헙에 원래 각자 공개키는 올려두고 쓰지 않나?
- 참, 우리 회사도 깃헙 쓰지?
- 깃헙 organization에 팀원 추가/제거하는 행동으로  
알아서 되게끔 못하려나?



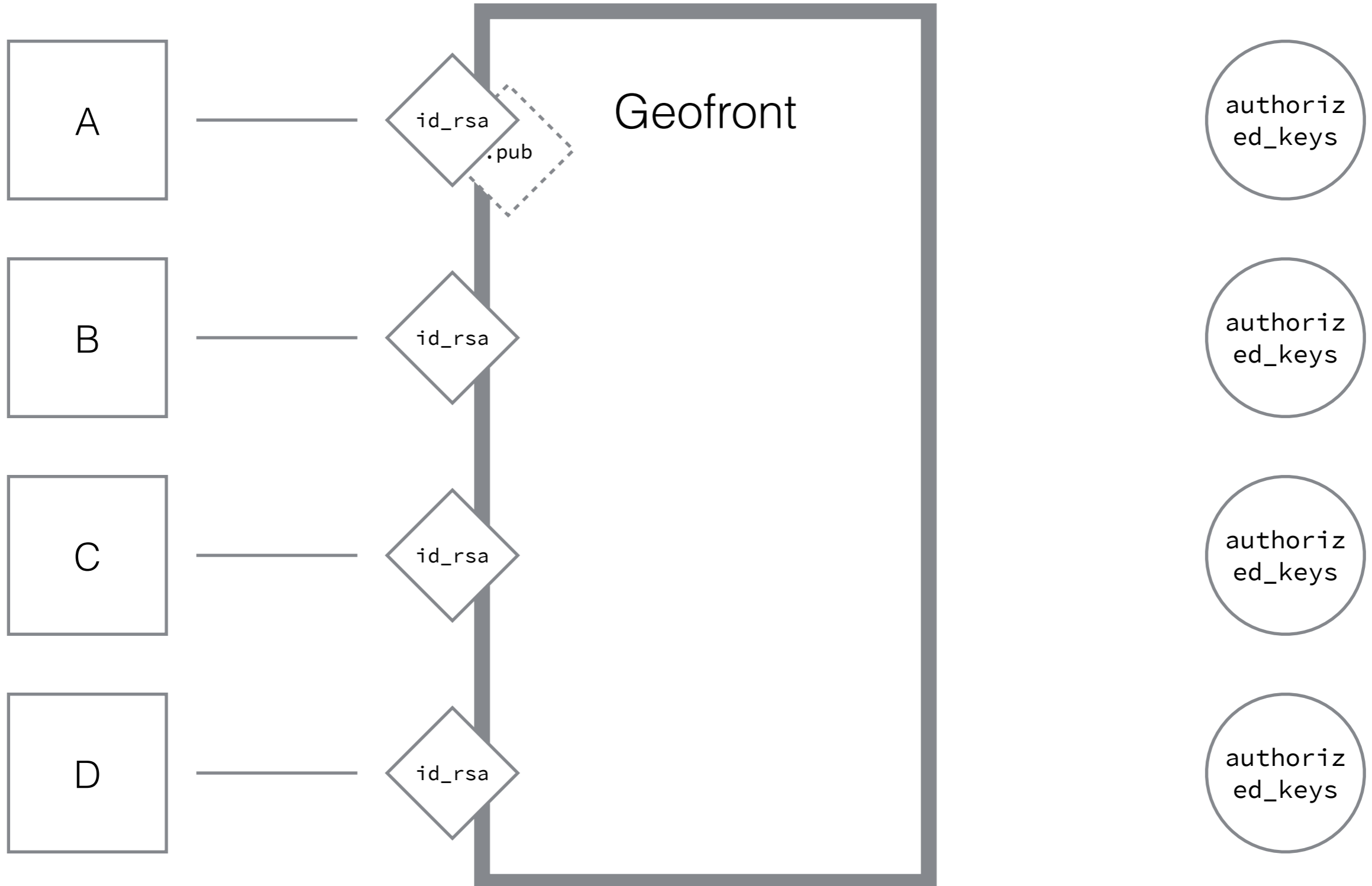
# 마스터 키



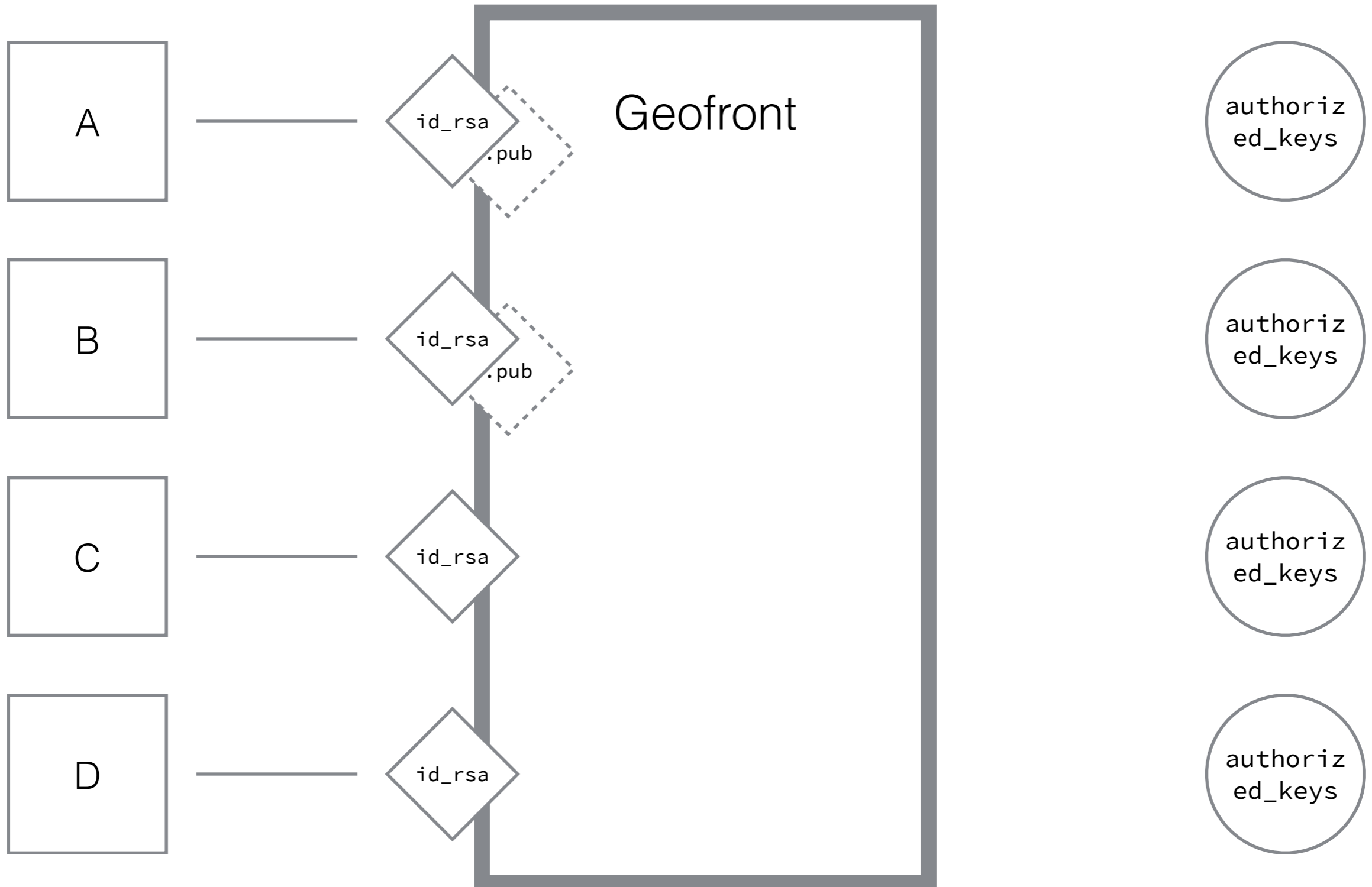
# 마스터 키



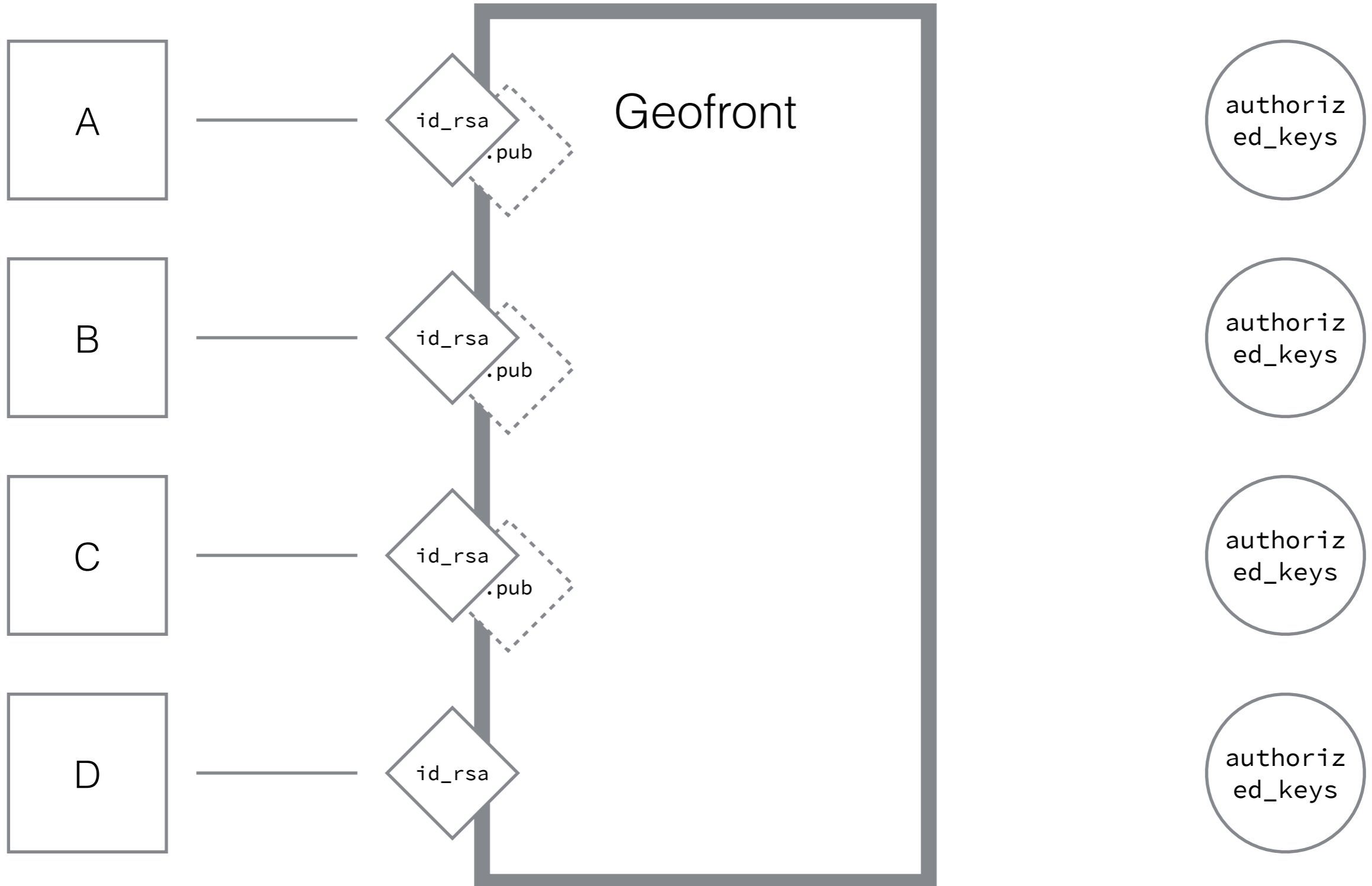
# 마스터 키



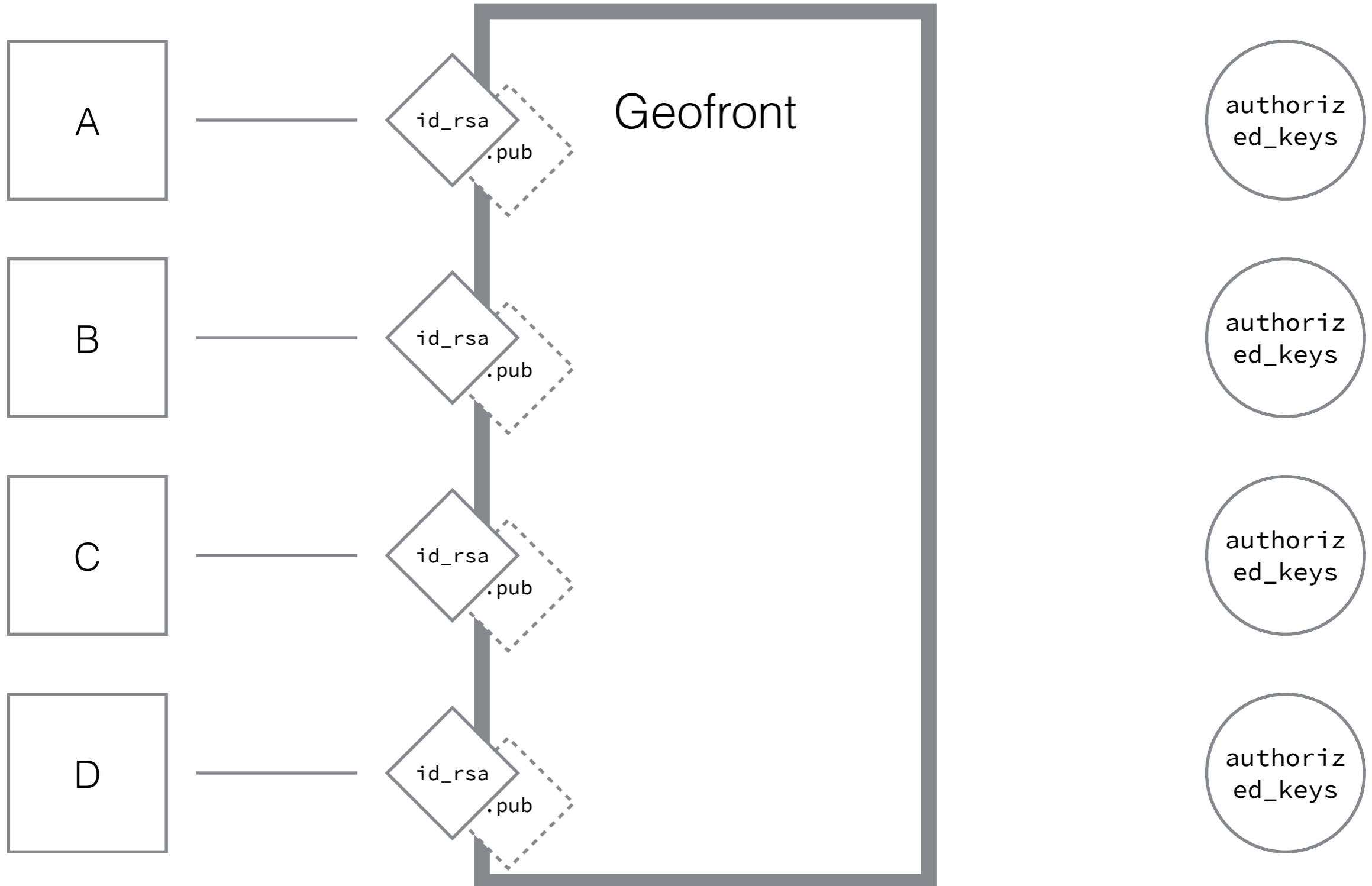
# 마스터 키



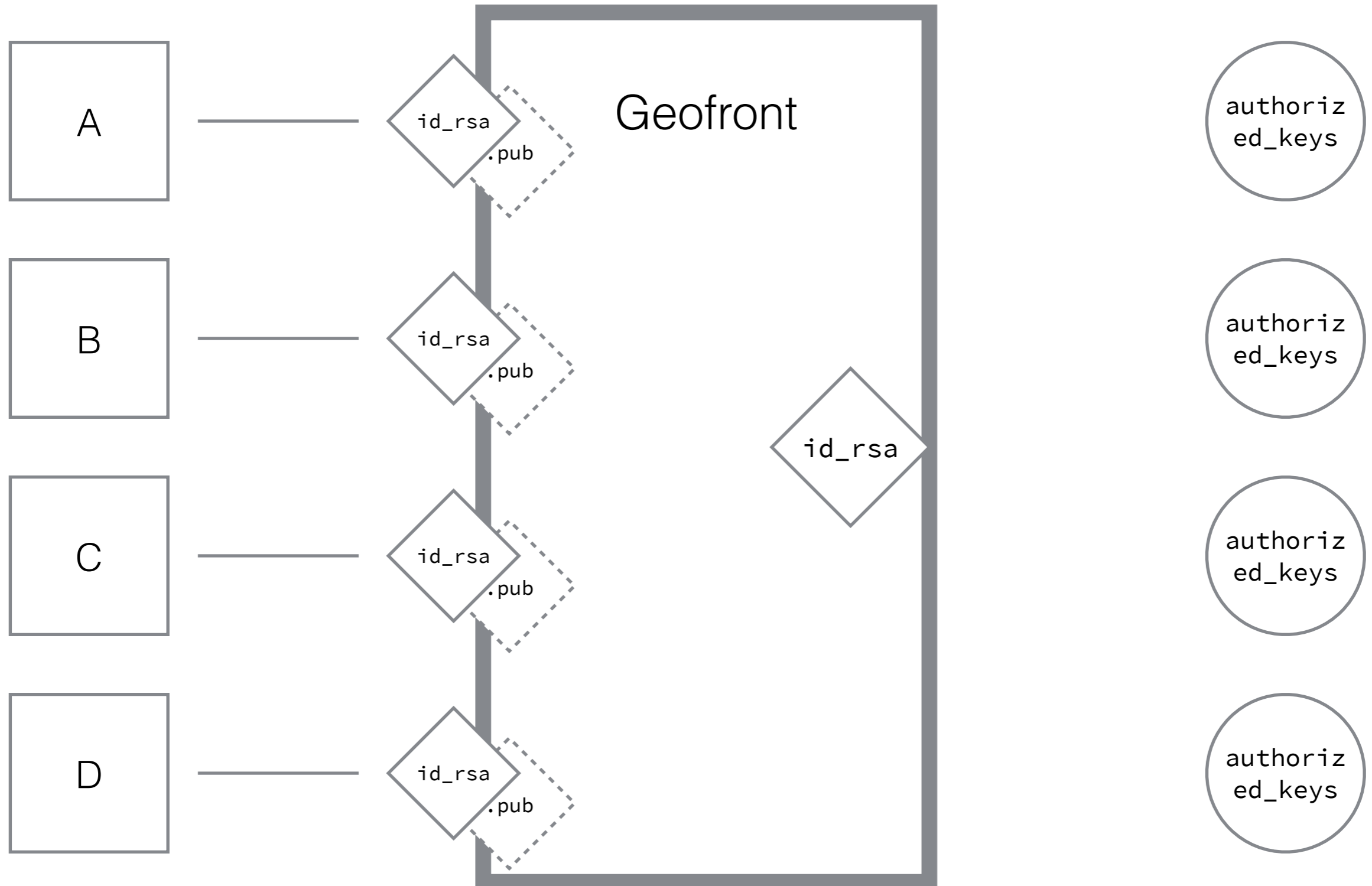
# 마스터 키



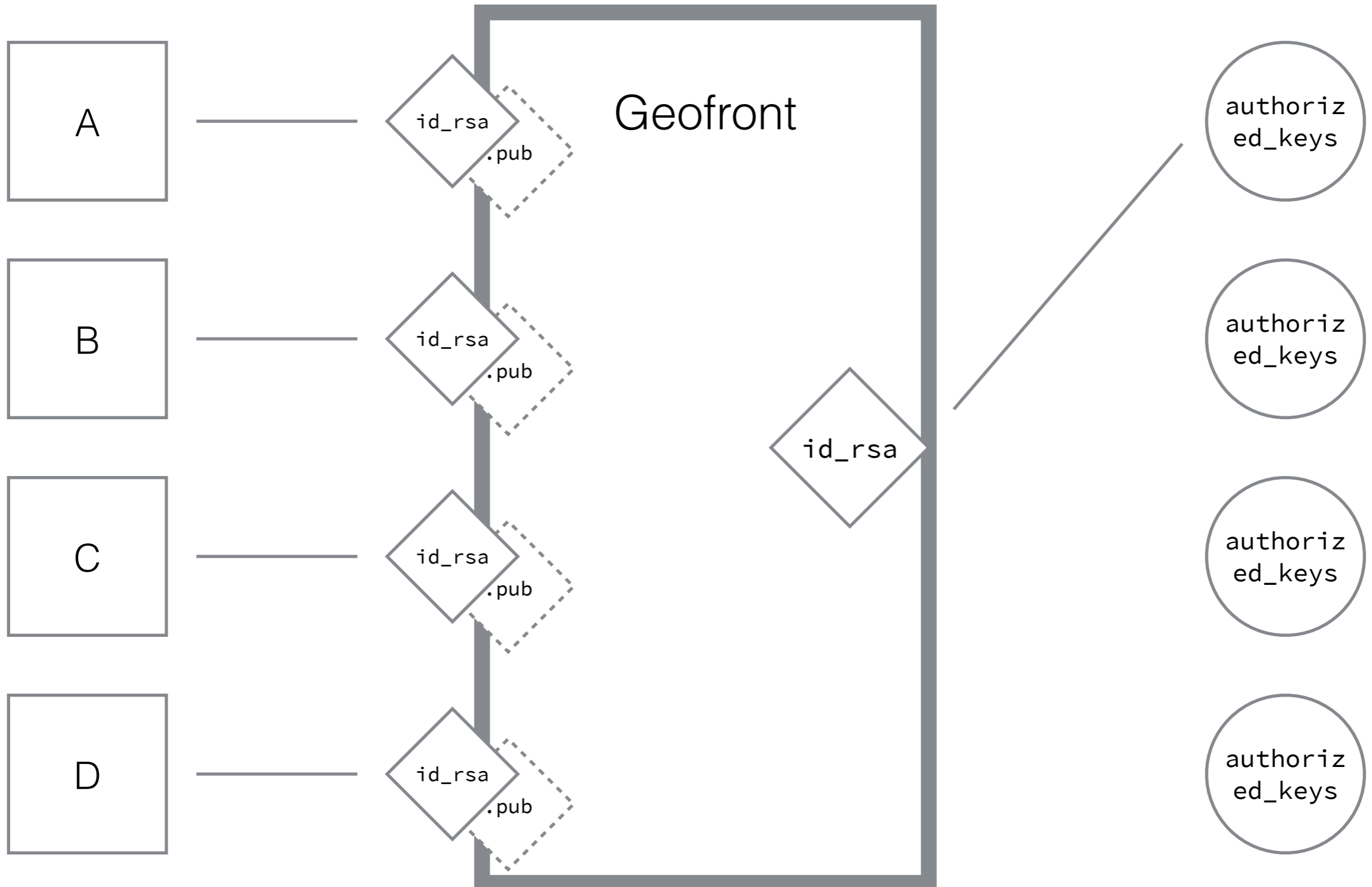
# 마스터 키



# 마스터 키

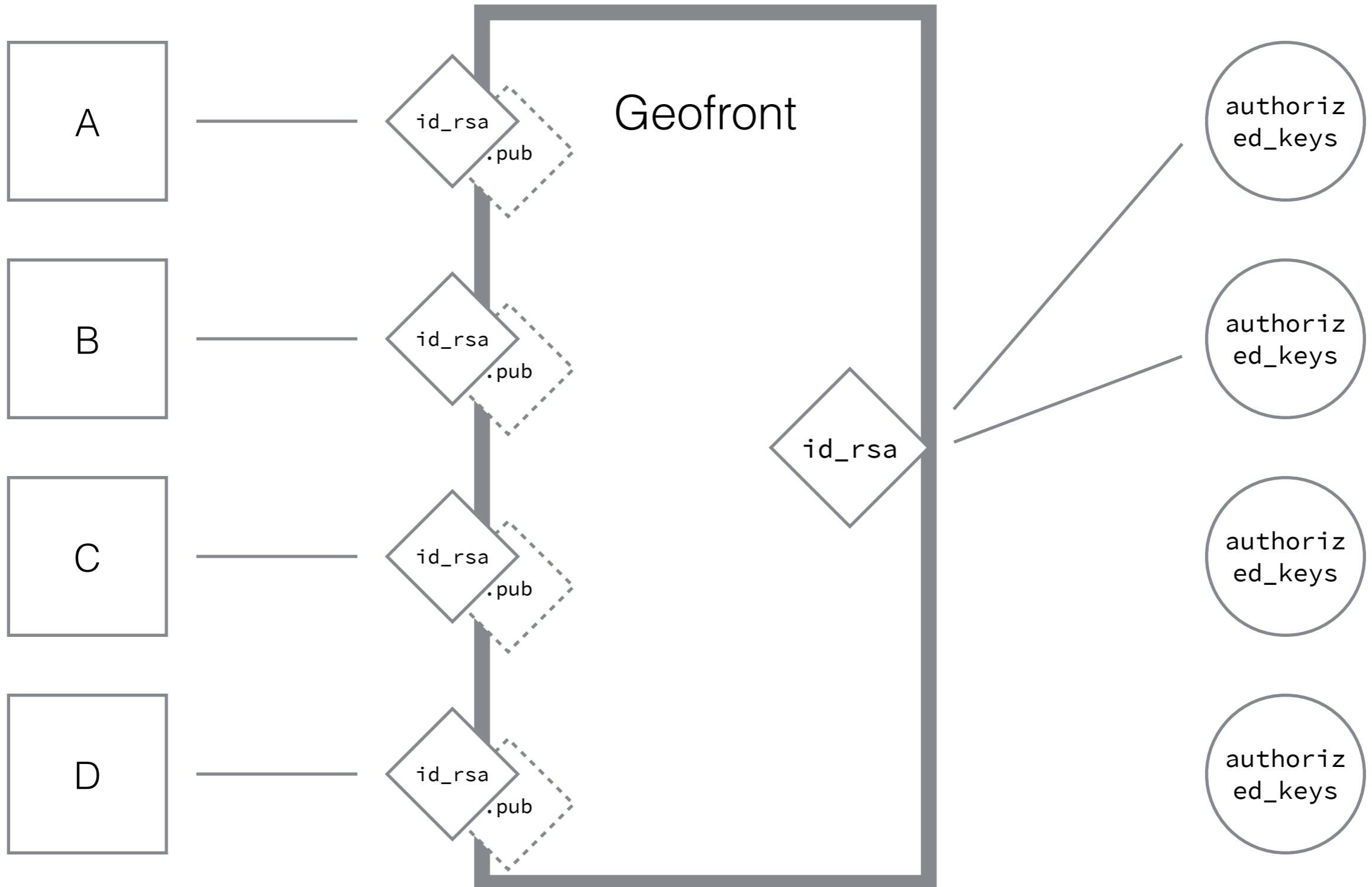


# 마스터 키

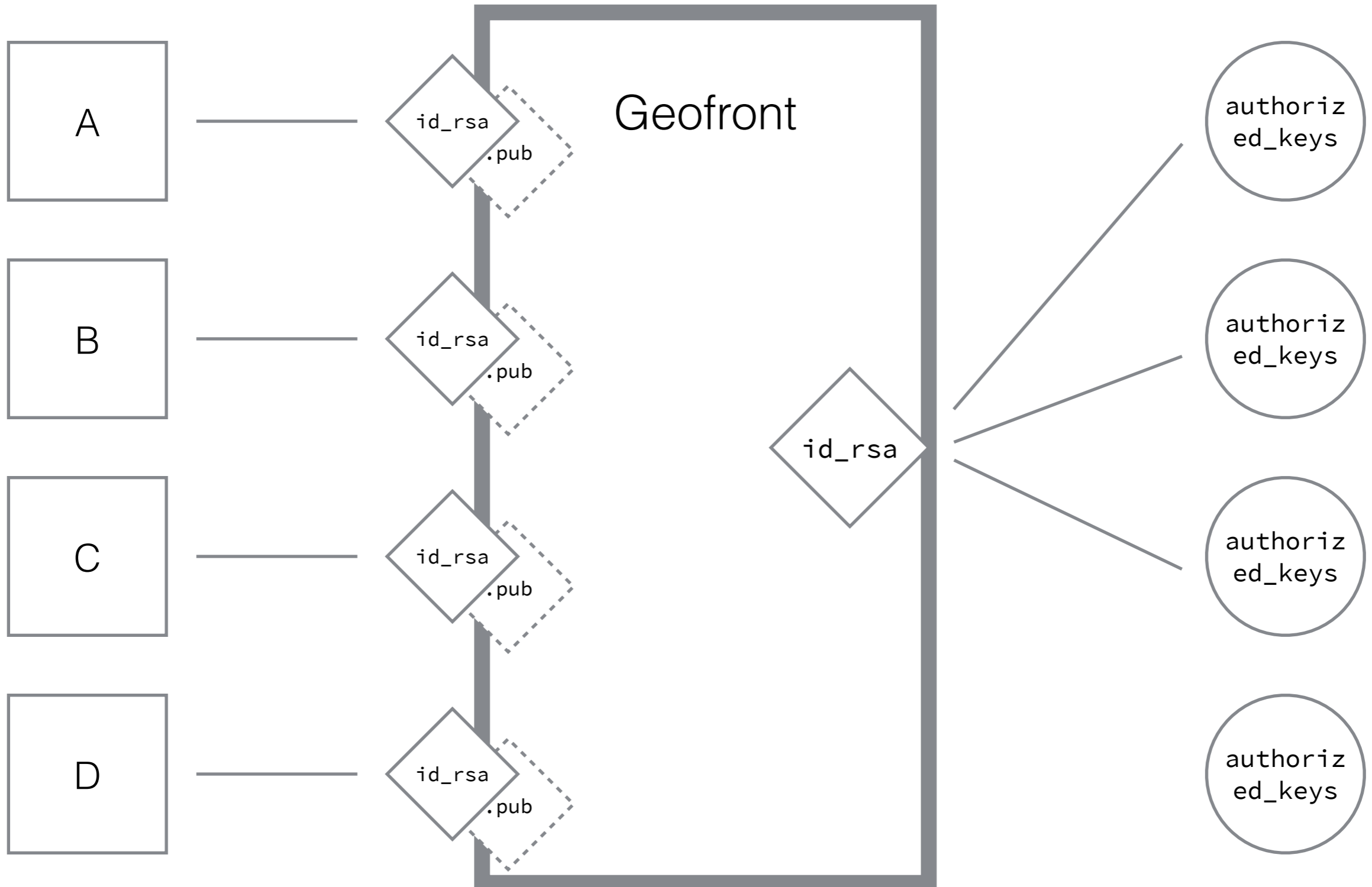




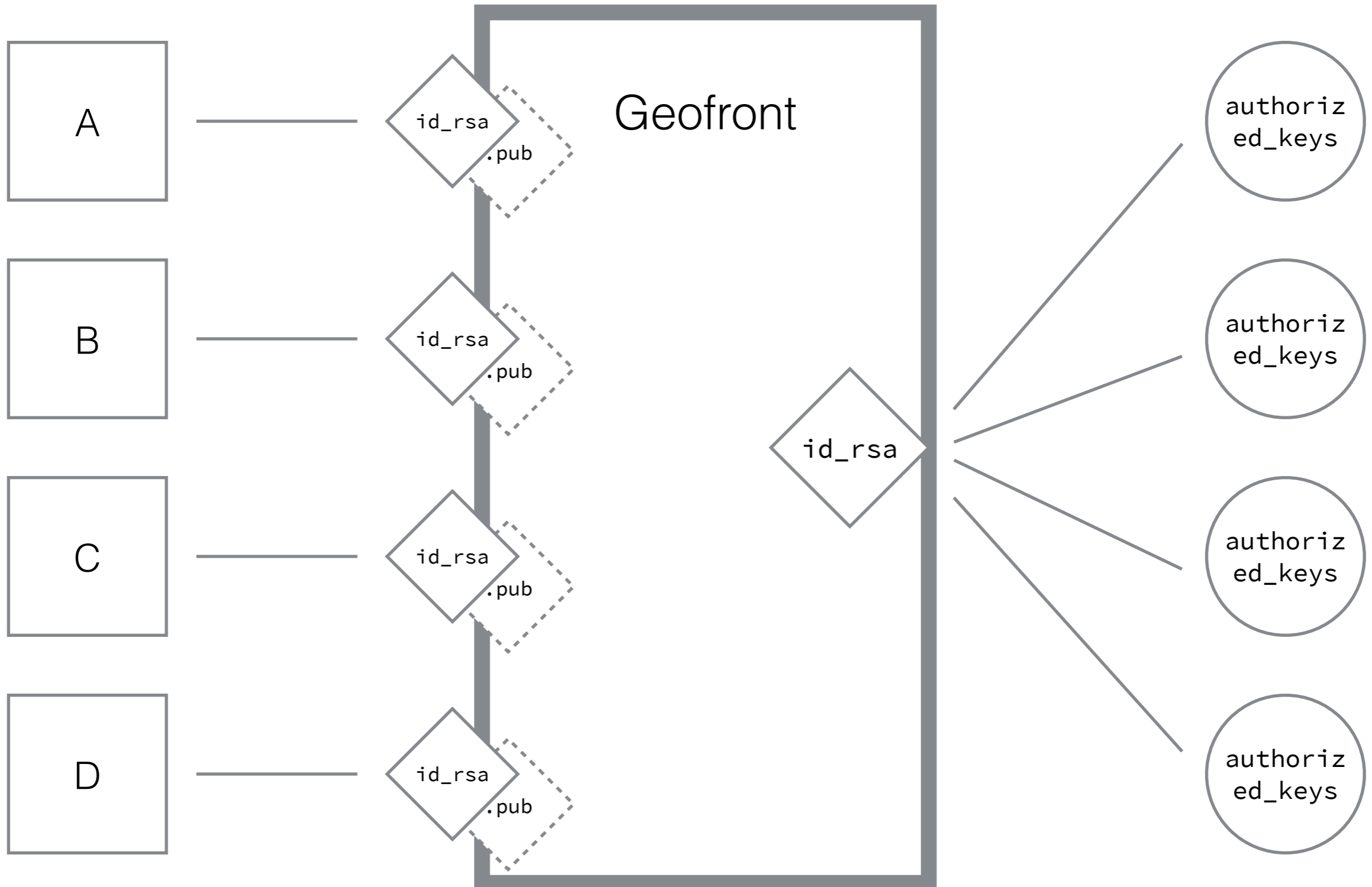
# 마스터 키



# 마스터 키



# 마스터 키



# 마스터 키 아이디어

# 마스터 키 아이디어

- 지오프론트 서버는 하나의 마스터 키를 유지

# 마스터 키 아이디어

- 지오프론트 서버는 하나의 마스터 키를 유지
- 마스터 키는 지오프론트 서버만 접근 가능한 곳에 보존

# 마스터 키 아이디어

- 지오프론트 서버는 하나의 마스터 키를 유지
- 마스터 키는 지오프론트 서버만 접근 가능한 곳에 보존
- 모든 원격지는 마스터 키만 허용하는 단 1줄짜리 `authorized_keys`만 유지

# 마스터 키 아이디어

- 지오프론트 서버는 하나의 마스터 키를 유지
- 마스터 키는 지오프론트 서버만 접근 가능한 곳에 보존
- 모든 원격지는 마스터 키만 허용하는 단 1줄짜리 `authorized_keys`만 유지
- 어디서 많이 본... 공유 키 방식?



# 마스터 키 아이디어

- 지오프론트 서버는 하나의 마스터 키를 유지
- 마스터 키는 지오프론트 서버만 접근 가능한 곳에 보존
- 모든 원격지는 마스터 키만 허용하는 단 1줄짜리 `authorized_keys`만 유지
- 어디서 많이 본... 공유 키 방식?
- 마스터 키는 주기적으로 파기 후 재생성 (기본값 하루)

# 지오프론트 CLI

# 지오프론트 CLI

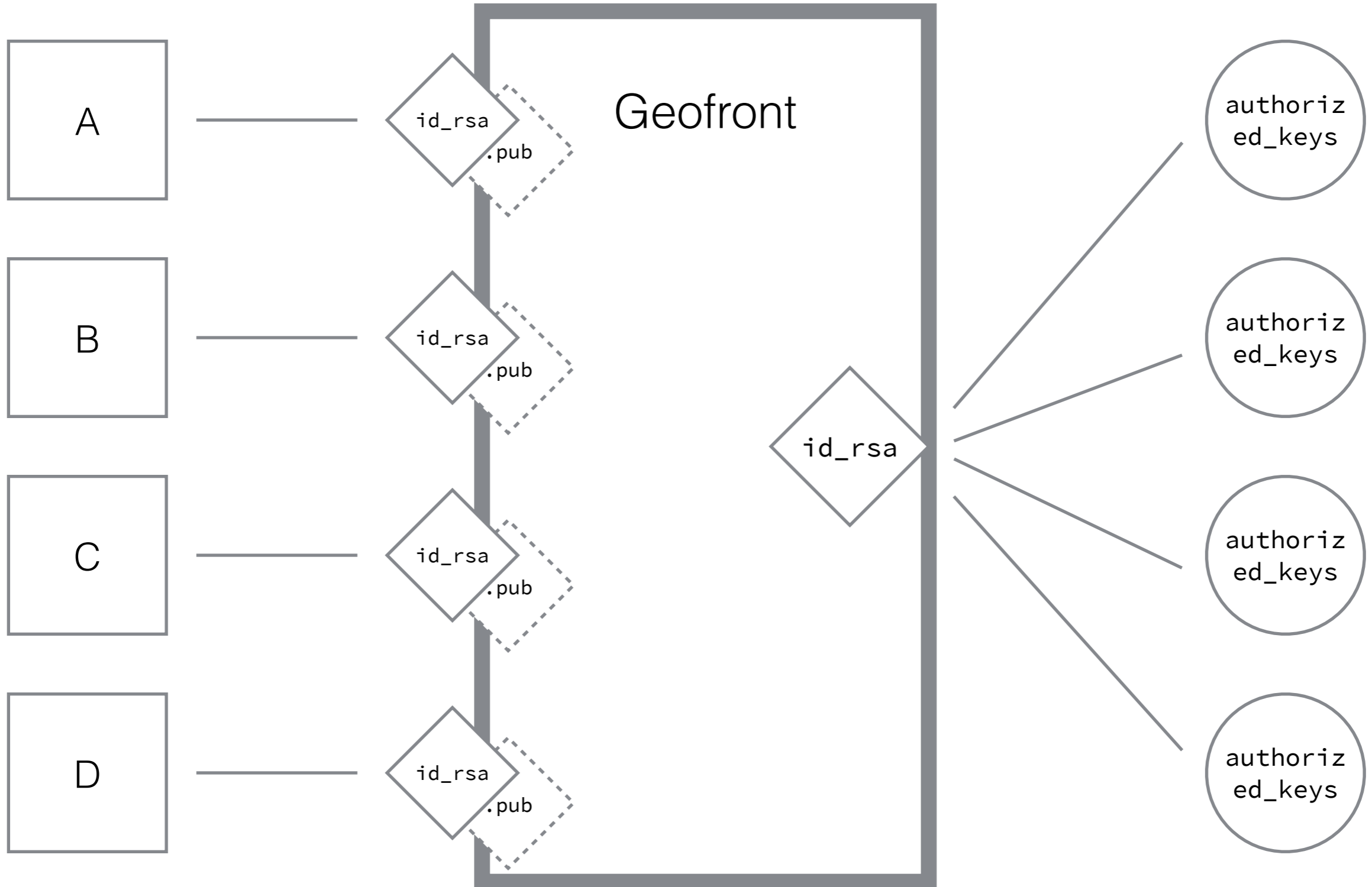
```
$ pip install --user geofront-cli
```

# 지오프론트 CLI

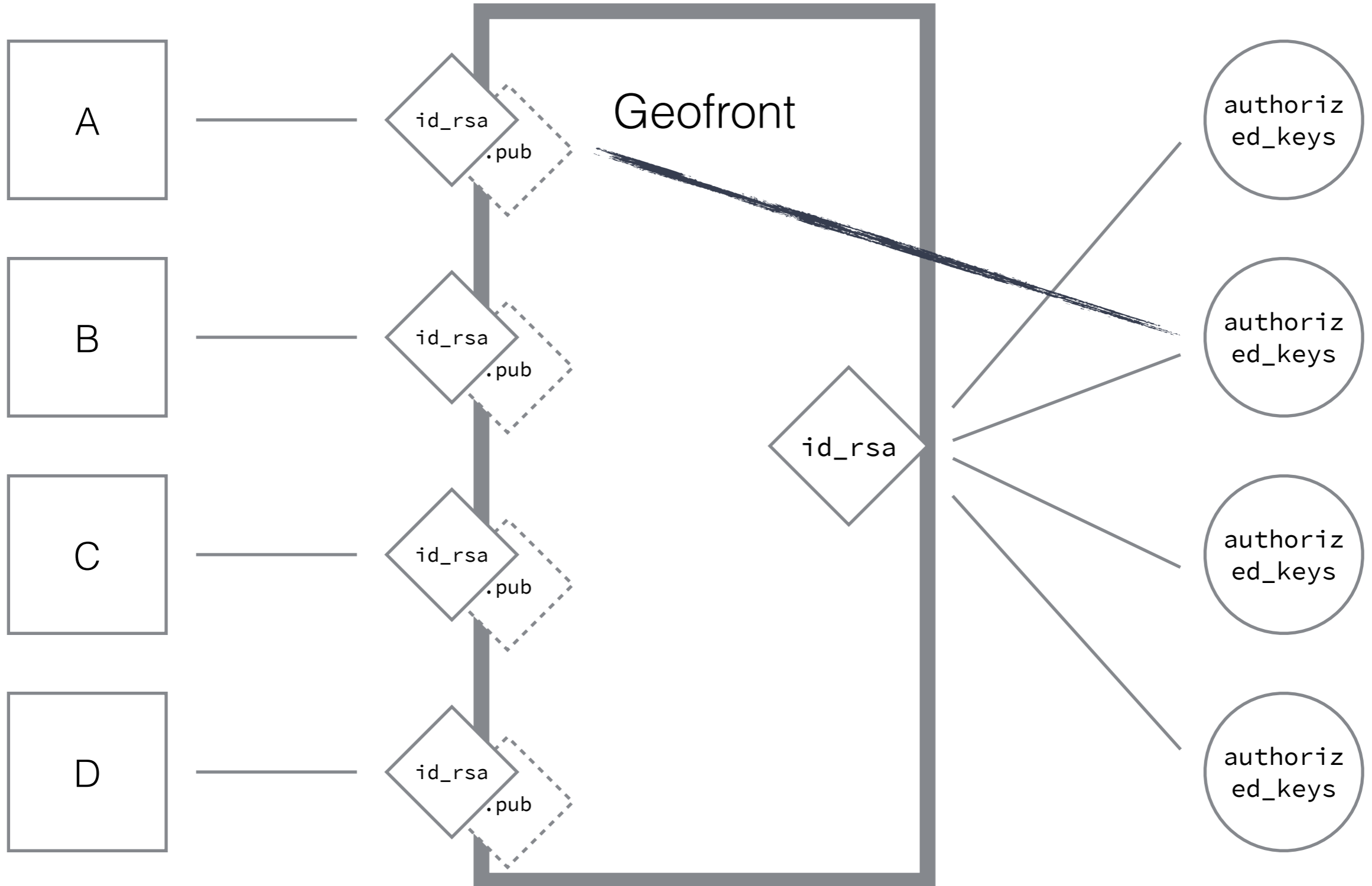
```
$ pip install --user geofront-cli
```

```
$ geofront-cli authorize server-2
```

# 한시적 허용



# 한시적 허용



# 허용된 원격지에 접속

```
$ pip install --user geofront-cli
```

```
$ geofront-cli authorize server-2
```

```
$ ssh server-2
```

# 한번에

```
$ pip install --user geofront-cli
```

```
$ geofront-cli authorize server-2
```

```
$ ssh server-2
```

```
$ geofront-cli ssh server-2 # shortcut
```



# 한시적 허용 아이디어

# 한시적 허용 아이디어

- 팀원은 지오프론트 서버에 팀의 일원임을 인증

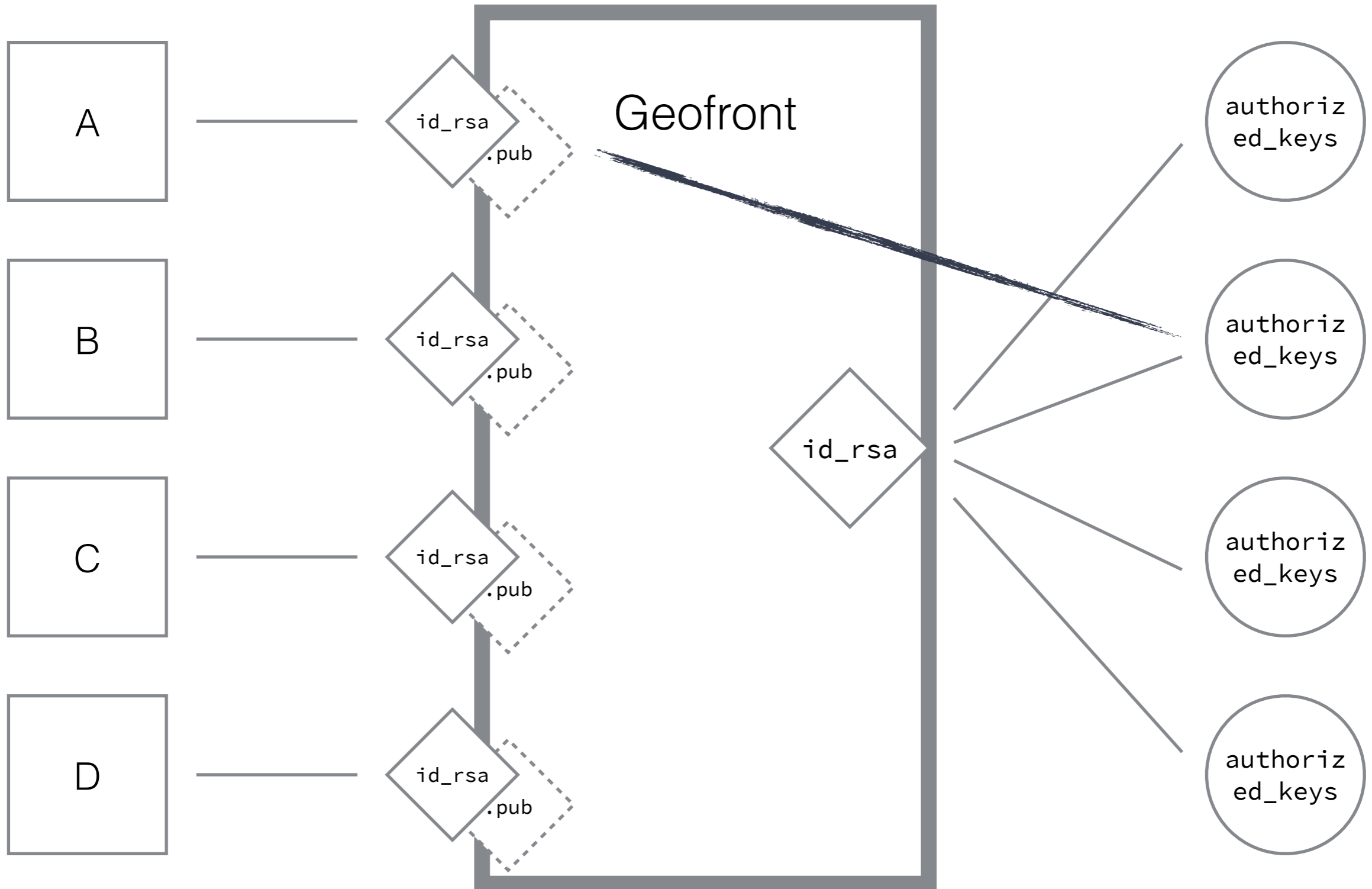
# 한시적 허용 아이디어

- 팀원은 지오프론트 서버에 팀의 일원임을 인증
- 인증된 팀원의 공개키를 접속할 원격지의 `authorized_keys`에 한시적으로 추가

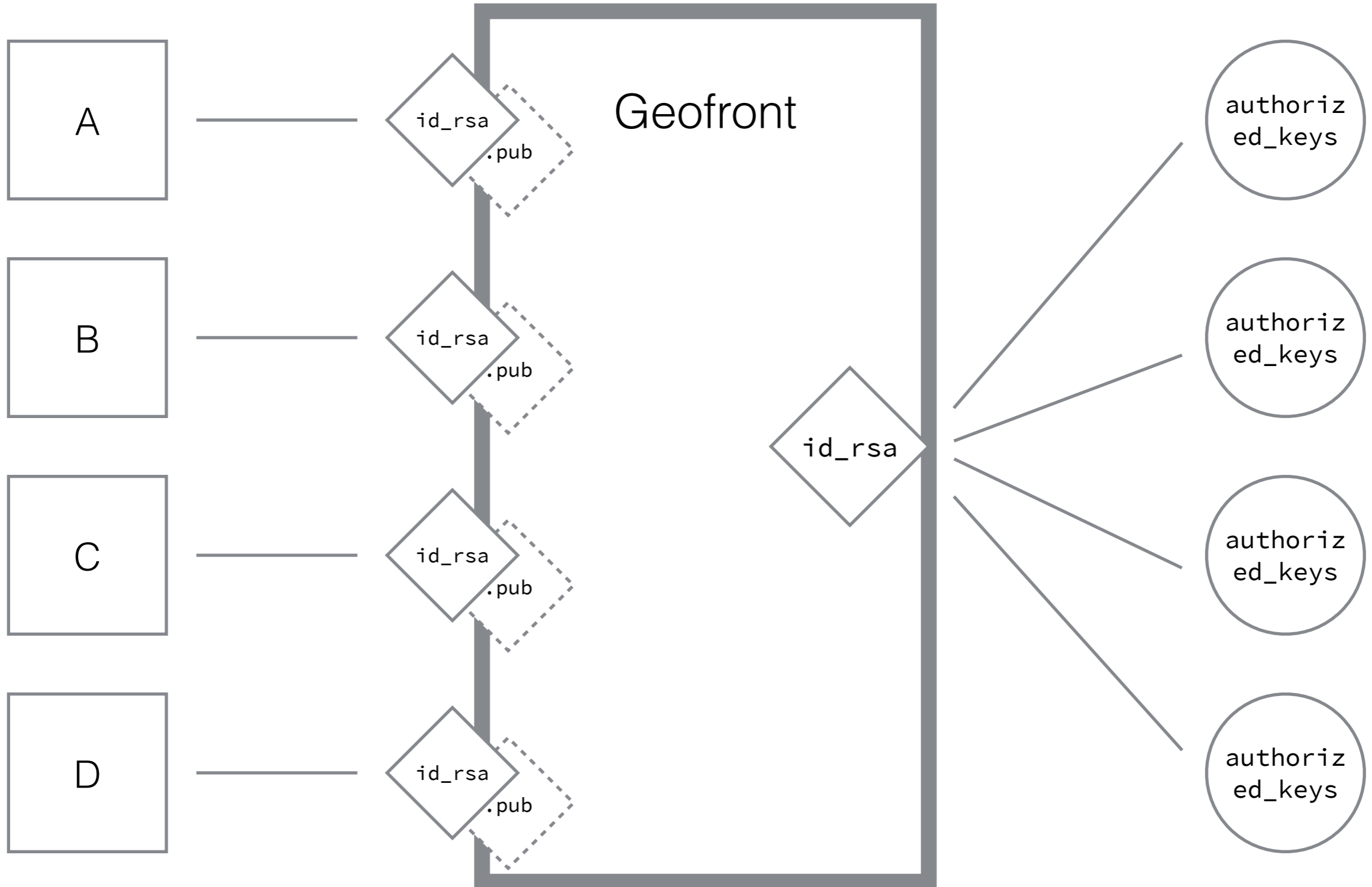
# 한시적 허용 아이디어

- 팀원은 지오프론트 서버에 팀의 일원임을 인증
- 인증된 팀원의 공개키를 접속할 원격지의 `authorized_keys`에 한시적으로 추가
- 해당 공개키는 약 60초 후에 다시 빠짐

# 한시적 허용



# 한시적 허용



# 팀원 인증

# 팀원 인증

- 팀이 일원이라는 것을 어떻게 증명하지?



# 팀원 인증

- 팀이 일원이라는 것을 어떻게 증명하지?
- 깃헙 OAuth로 로그인해서 우리 팀 organization 계정에 들어있는지 API로 확인

# 팀원 인증

- 팀이 일원이라는 것을 어떻게 증명하지?
- 깃헙 OAuth로 로그인해서 우리 팀 organization 계정에 들어있는지 API로 확인
- 팀에서 빠지면 인증도 자동으로 안됨!

# 팀원 인증

- 팀이 일원이라는 것을 어떻게 증명하지?
- 깃헙 OAuth로 로그인해서 우리 팀 organization 계정에 들어있는지 API로 확인
- 팀에서 빠지면 인증도 자동으로 안됨!
- 팀원 인증 방식은 인터페이스와 구현이 분리되어 있음!

# 팀원 인증

- 팀이 일원이라는 것을 어떻게 증명하지?
- 깃헙 OAuth로 로그인해서 우리 팀 organization 계정에 들어있는지 API로 확인
- 팀에서 빠지면 인증도 자동으로 안됨!
- 팀원 인증 방식은 인터페이스와 구현이 분리되어 있음!
- 빗버킷/깃랩/구글 앱스/자체 로그인 등 구현 가능

# 공개키 등록

# 공개키 등록

- 각자 공개키를 지오프론트에 직접 올리나?

# 공개키 등록

- 각자 공개키를 지오프론트에 직접 올리나?
- 그럴 수도 있음!

# 공개키 등록

- 각자 공개키를 지오프론트에 직접 올리나?
- 그럴 수도 있음!
- 하지만 깃헙 연동을 설정하면 이미 깃헙 계정에 등록된 공개키도 사용할 수 있음



# 공개키 등록

- 각자 공개키를 지오프론트에 직접 올리나?
- 그럴 수도 있음!
- 하지만 깃헙 연동을 설정하면 이미 깃헙 계정에 등록된 공개키도 사용할 수 있음
- 신규 팀원 입장에서는 깃헙 organization 계정에 추가만 되면 아무 설정도 할 게 없음!

# 공개키 등록

- 각자 공개키를 지오프론트에 직접 올리나?
- 그럴 수도 있음!
- 하지만 깃헙 연동을 설정하면 이미 깃헙 계정에 등록된 공개키도 사용할 수 있음
- 신규 팀원 입장에서는 깃헙 organization 계정에 추가만 되면 아무 설정도 할 게 없음!
- 기존 팀원 입장에서도 깃헙 organization 계정에 해당 신규 팀원 추가만 하면 다른 건 해줄 필요 없음!

# 원격지 추가

# 원격지 추가

- 원격지 목록을 관리하는 인터페이스와 구현이 분리돼있음

# 원격지 추가

- 원격지 목록을 관리하는 인터페이스와 구현이 분리돼있음
- AWS 인스턴스 목록 그대로 사용하는 구현체도 있음

# 원격지 추가

- 원격지 목록을 관리하는 인터페이스와 구현이 분리돼있음
- AWS 인스턴스 목록 그대로 사용하는 구현체도 있음
- 사실은 Libcloud 써서 유클라우드도 되지롱...!

# 원격지 추가

- 원격지 목록을 관리하는 인터페이스와 구현이 분리돼있음
- AWS 인스턴스 목록 그대로 사용하는 구현체도 있음
- 사실은 Libcloud 써서 유클라우드도 되지롱...!
- 난 안 쓸 거지만...

# 원격지 추가

- 원격지 목록을 관리하는 인터페이스와 구현이 분리돼있음
- AWS 인스턴스 목록 그대로 사용하는 구현체도 있음
- 사실은 Libcloud 써서 유클라우드도 되지롱...!
- 난 안 쓸 거지만...
- 하드 코딩 목록으로 관리하는 구현체도 있음



# 원격지 추가

- 원격지 목록을 관리하는 인터페이스와 구현이 분리돼있음
- AWS 인스턴스 목록 그대로 사용하는 구현체도 있음
- 사실은 Libcloud 써서 유클라우드도 되지롱...!
- 난 안 쓸 거지만...
- 하드 코딩 목록으로 관리하는 구현체도 있음
- 직접 구현 클래스 짜서 사용해도 됨

# 관련 링크

# 관련 링크

- <https://github.com/spoqa/geofront>

# 관련 링크

- <https://github.com/spoqa/geofront>
- <https://geofront.readthedocs.org/>

# 관련 링크

- <https://github.com/spoqa/geofront>
- <https://geofront.readthedocs.org/>
- <http://spoqa.github.io/2014/07/09/geofront.html>

# pip로 설치 가능

```
$ pip install Geofront
```

# 2부: 지오프론트 개발 후기

파이썬 2 성불기





전적

# 전적

- '08 웹 프레임워크 (파이썬 3 전용) — **실패!**

# 전적

- '08 웹 프레임워크 (파이썬 3 전용) — **실패!**
- '13 Wand (파이썬 2 전용 → 2/3 동시 지원)

# 전적

- '08 웹 프레임워크 (파이썬 3 전용) — **실패!**
- '13 Wand (파이썬 2 전용 → 2/3 동시 지원)
- '13 alembic (파이썬 2 전용 → 2/3 동시 지원)

# 전적

- '08 웹 프레임워크 (파이썬 3 전용) — **실패!**
- '13 Wand (파이썬 2 전용 → 2/3 동시 지원)
- '13 alembic (파이썬 2 전용 → 2/3 동시 지원)
- '13 SQLAlchemy-ImageAttach (2/3 동시 지원)

# 전적

- '08 웹 프레임워크 (파이썬 3 전용) — **실패!**
- '13 Wand (파이썬 2 전용 → 2/3 동시 지원)
- '13 alembic (파이썬 2 전용 → 2/3 동시 지원)
- '13 SQLAlchemy-ImageAttach (2/3 동시 지원)
- '13 Earth Reader (2/3 동시 지원)

# 전적

- '08 웹 프레임워크 (파이썬 3 전용) — 실패!
- '13 Wand (파이썬 2 전용 → 2/3 동시 지원)
- '13 alembic (파이썬 2 전용 → 2/3 동시 지원)
- '13 SQLAlchemy-ImageAttach (2/3 동시 지원)
- '13 Earth Reader (2/3 동시 지원)
- '14 libsass-python (파이썬 2 전용 → 2/3 동시 지원)

전적



# 전적

- 파이썬 2 전용 → 2/3 동시 지원 포팅: 3번

# 전적

- 파이썬 2 전용 → 2/3 동시 지원 포팅: 3번
- 처음부터 2/3 동시 지원: 2번

# 전적

- 파이썬 2 전용 → 2/3 동시 지원 포팅: 3번
- 처음부터 2/3 동시 지원: 2번
- 파이썬 3 전용: 1번 — 실패!

# 파이썬 3 기피증

# 파이썬 3 기피증

- 2008년에 크게 실패한 뒤로 파이썬 3는 쳐다도 안봤다

# 파이썬 3 기피증

- 2008년에 크게 실패한 뒤로 파이썬 3는 쳐다도 안봤다
- 그런데 2012년부터 Wand 이슈 트래커에서 파이썬 3 패치가 (여러번) 올라옴

# 파이썬 3 기피증

- 2008년에 크게 실패한 뒤로 파이썬 3는 쳐다도 안봤다
- 그런데 2012년부터 Wand 이슈 트래커에서 파이썬 3 패치가 (여러번) 올라옴
- libsass-python 이슈 트래커에서도 파이썬 3 지원 요청이 올라옴



This repository Search

Explore Gist Blog Help



dahlia

+ ▾



dahlia / wand

Unwatch 34

Unstar 374

Fork 93

# Python 3 support #15

Edit New issue

**Closed** johnsoft wants to merge 2 commits into dahlia:master from johnsoft:py3k-fixes

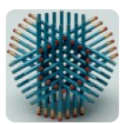
Conversation 2 Commits 2 Files changed 2



johnsoft commented on 14 Mar 2012

Not sure how you feel about Python 3, but here's a patch if you're willing to support it. There's probably some setup.py/2to3 mojo that needs added before it's 100% ready, but I install packages by hand so I can't really help with that.

- johnsoft added some commits on 13 Mar 2012
- Fixed creating an image from a blob in py3k. 017dfe9
  - Formats are now represented as str instead of bytes in Python 3. 3c5c2fa



dahlia commented on 17 Mar 2012

Owner

Labels

enhance

Milestone

0.3.0

Assignee

No one assigned

Notifications

Unsubscribe




# Py3k fixes #80

Edit New issue

**Closed** johnsoft wants to merge 6 commits into dahlia:master from johnsoft:py3k-fixes

Conversation 12 Commits 6 Files changed 3

+12 -15

 johnsoft commented on 8 Dec 2012  
Everybody loves Python 3

Labels  
None yet

Milestone  
0.4.0

Assignee  
No one assigned

Notifications  
**Unsubscribe**

You're receiving notifications

- johnsoft added some commits on 20 Nov 2012
- Fix syntax errors in Python 3 a32383a
  - Improve error reporting when the MagickWand library can't be found 554614c
  - Fix creating an image from a file in Python 3 96dc826
  - Fix creating an image from a blob in Python 3 37b4344
  - Fix Image.transform() string handling in Python 3 b94348d
  - Fix saving an image to a file in Python 3 1ea41e0



This repository Search

Explore Gist Blog Help



dahlia



dahlia / **libsass-python**

forked from [sass/libsass](#)

Unwatch 9

Unstar 77

Fork 164

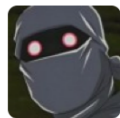
# Python 3 support #7

Edit

New issue

**Closed**

FSX opened this issue on 9 Jan 2013 · 4 comments



FSX commented on 9 Jan 2013

Hey, I was wondering if support for Python 3 is planned. It currently does not install, the unit tests do not run and the dependencies needed for the unit tests don't support Python 3.

I'm attempting to make it work in Python 3 and so far I've managed to make the package install and run the compile function (only with byte strings). Next step is unicode. 🐛

The dependencies need to be replaced since they don't support Python 3. Attest is supposed to work, but it's dependency, progressbar doesn't work. I don't know what Werkzeug's status is.

- Package installation
- Importing the Sass module
- Compile Sass text (only byte strings)
- Use unicode strings instead of byte strings

Labels



None yet

Milestone



No milestone

Assignee



No one assigned

Notifications

Unsubscribe

You're receiving notifications because you modified the open/close state.

# 파이썬 3 호환성 작업

# 파이썬 3 호환성 작업

- 꾸준한 수요로 지원을 안 할 수가 없었음

# 파이썬 3 호환성 작업

- 꾸준한 수요로 지원을 안 할 수가 없었음
- 파이썬 2/3 동시 지원하는 코드를 짜는 노하우 축적

# 파이썬 3 호환성 작업

- 꾸준한 수요로 지원을 안 할 수가 없었음
- 파이썬 2/3 동시 지원하는 코드를 짜는 노하우 축적
- 파이썬 3 기피증이 조금씩 사라짐

# 파이썬 3 호환성 작업

- 꾸준한 수요로 지원을 안 할 수가 없었음
- 파이썬 2/3 동시 지원하는 코드를 짜는 노하우 축적
- 파이썬 3 기피증이 조금씩 사라짐
- '13년부터는 아예 자발적으로 파이썬 3 호환 작업을 시작

왜 기피했더라



# 왜 기피했더라

- 파이썬 3에 대해 점차 재평가하게 됨

# 왜 기피했더라

- 파이썬 3에 대해 점차 재평가하게 됨
- 애초에 파이썬 3를 기피하게 됐던 이유를 되돌아봄

# 트라우마

# 트라우마

- 2008년의 실패가 트라우마가 되었음

# 트라우마

- 2008년의 실패가 트라우마가 되었음
  - **WSGI 프로토콜**이 파이썬 3에서 어떻게 되어야 하는지에 대한 합의가 없던 시절 (PEP 3333 없었음)

# 트라우마

- 2008년의 실패가 트라우마가 되었음
  - **WSGI 프로토콜**이 파이썬 3에서 어떻게 되어야 하는지에 대한 합의가 없던 시절 (PEP 3333 없었음)
  - 파이썬 3.0 시절의 **표준 라이브러리**는 혼돈의 카오스

# 트라우마

- 2008년의 실패가 트라우마가 되었음
  - **WSGI 프로토콜**이 파이썬 3에서 어떻게 되어야 하는지에 대한 합의가 없던 시절 (PEP 3333 없었음)
  - 파이썬 3.0 시절의 **표준 라이브러리**는 혼돈의 카오스
  - URL 파싱과 HTTP 메시지 파싱을 손으로 짜야 했음

# 트라우마

- 2008년의 실패가 트라우마가 되었음
  - WSGI 프로토콜이 파이썬 3에서 어떻게 되어야 하는지에 대한 합의가 없던 시절 (PEP 3333 없었음)
  - 파이썬 3.0 시절의 표준 라이브러리는 혼돈의 카오스
  - URL 파싱과 HTTP 메시지 파싱을 손으로 짜야 했음
  - 파이썬 3를 쓰는 고객도 없었음



# 생태계의 위기

# 생태계의 위기

- 너무나 부족했던 서드파티 라이브러리

# 생태계의 위기

- 너무나 부족했던 서드파티 라이브러리
- PyPI에서 파이썬 3 지원하는 패키지가 손에 꼽혔을 정도

# 생태계의 위기

- 너무나 부족했던 서드파티 라이브러리
- PyPI에서 파이썬 3 지원하는 패키지가 손에 꼽혔을 정도
- 3.3에서 유니코드 리터럴 (u'...' )이 부활하기 전까지 2/3 호환 코드 짜기는 무척 힘들었음

# 생태계의 위기

- 너무나 부족했던 서드파티 라이브러리
- PyPI에서 파이썬 3 지원하는 패키지가 손에 꼽혔을 정도
- 3.3에서 유니코드 리터럴 (u'...' )이 부활하기 전까지 2/3 호환 코드 짜기는 무척 힘들었음
- PyPy 역시 파이썬 3를 최근까지도 구현 안했음

**생필품 부족**

# 생필품 부족

- numpy, scipy

# 생필품 부족

- numpy, scipy
- Django, South



# 생필품 부족

- numpy, scipy
- Django, South
- Werkzeug, Flask

# 생필품 부족

- numpy, scipy
- Django, South
- Werkzeug, Flask
- pycrypto, paramiko

# 생필품 부족

- numpy, scipy
- Django, South
- Werkzeug, Flask
- pycrypto, paramiko
- kombu, celery

# 생필품 부족

- numpy, scipy
- Django, South
- Werkzeug, Flask
- pycrypto, paramiko
- kombu, celery
- PIL

# 플랫폼

# 플랫폼

- 주요 리눅스 배포판에서 파이썬 2가 기본이었거나

# 플랫폼

- 주요 리눅스 배포판에서 파이썬 2가 기본이었거나
- 아예 파이썬 3 패키지가 없는 배포판도

# 플랫폼

- 주요 리눅스 배포판에서 파이썬 2가 기본이었거나
- 아예 파이썬 3 패키지가 없는 배포판도
- 맥 역시 파이썬 3를 번들하지 않음



# 플랫폼

- 주요 리눅스 배포판에서 파이썬 2가 기본이었거나
- 아예 파이썬 3 패키지가 없는 배포판도
- 맥 역시 파이썬 3를 번들하지 않음
- 구글 앱 엔진 등의 PaaS도 파이썬 3를 지원 안했음

# 나 자신과의 싸움

# 나 자신과의 싸움

- 내가 찢던 레거시는 또 어쩔 거여

지오프론트는...?

# 지오프론트는...?

- 지오프론트 서버는 파이썬 3 전용

# 지오프론트는...?

- 지오프론트 서버는 파이썬 3 전용
- 지오프론트 CLI 클라이언트는 파이썬 2.6 이상, 3.2 이상

# 지오프론트는...?

- 지오프론트 서버는 파이썬 3 전용
- 지오프론트 CLI 클라이언트는 파이썬 2.6 이상, 3.2 이상
- 어찌다 그런 생각을...

# 파이썬 3의 현재



# 파이썬 3의 현재

- 파이썬 3가 예정된 미래라는 데에 커뮤니티가 합의하는 추세

# 파이썬 3의 현재

- 파이썬 3가 예정된 미래라는 데에 커뮤니티가 합의하는 추세
- 희망하는 미래가 아니라 예정된 미래

# 파이썬 3의 현재

- 파이썬 3가 예정된 미래라는 데에 커뮤니티가 합의하는 추세
- 희망하는 미래가 아니라 예정된 미래
- 파이썬 2는 명백하게 늙어가고 있음

# 파이썬 3의 현재

- 파이썬 3가 예정된 미래라는 데에 커뮤니티가 합의하는 추세
- 희망하는 미래가 아니라 예정된 미래
- 파이썬 2는 명백하게 늙어가고 있음
- 파이썬 2에서는 점차 더이상 새로운 것이 나오지 않음

# 파이썬 3의 현재

# 파이썬 3의 현재

- 반면 파이썬 3의 생태계는 점차 커지고 있을 뿐만 아니라

# 파이썬 3의 현재

- 반면 파이썬 3의 생태계는 점차 커지고 있을 뿐만 아니라
- 커지는 가속도 역시 증가

# 파이썬 3의 현재

- 반면 파이썬 3의 생태계는 점차 커지고 있을 뿐만 아니라
- 커지는 가속도 역시 증가
- 기존 파이썬 3 문제의 거의 대부분은 생태계의 문제



# 파이썬 3의 현재

- 반면 파이썬 3의 생태계는 점차 커지고 있을 뿐만 아니라
- 커지는 가속도 역시 증가
- 기존 파이썬 3 문제의 거의 대부분은 생태계의 문제
- 그런데 생태계가 빠르게 발전하고 있다?

# 파이썬 3의 현재

- 반면 파이썬 3의 생태계는 점차 커지고 있을 뿐만 아니라
- 커지는 가속도 역시 증가
- 기존 파이썬 3 문제의 거의 대부분은 생태계의 문제
- 그런데 생태계가 빠르게 발전하고 있다?
- 파이썬 3에 대한 평가도 시시각각 재평가하지 않으면

# 파이썬 3의 현재

- 반면 파이썬 3의 생태계는 점차 커지고 있을 뿐만 아니라
- 커지는 가속도 역시 증가
- 기존 파이썬 3 문제의 거의 대부분은 생태계의 문제
- 그런데 생태계가 빠르게 발전하고 있다?
- 파이썬 3에 대한 평가도 시시각각 재평가하지 않으면
- 매달 파이썬 3의 생태계의 사정은 달라짐

# 2009년의 파이썬 3

- numpy, scipy
- Django, South
- Werkzeug, Flask
- pycrypto, paramiko
- celery
- PIL

# 2010년의 파이썬 3

- numpy (1.5.0), scipy
- Django, South
- Werkzeug, Flask
- pycrypto, paramiko
- celery
- PIL

# 2011년의 파이썬 3

- numpy (1.5.0), scipy (0.9.0)
- Django, South
- Werkzeug, Flask
- pycrypto (2.4), paramiko
- celery (2.4.0)
- PIL

# 2013년의 파이썬 3

- numpy (1.5.0), scipy (0.9.0)
- Django (1.5), South (0.8)
- Werkzeug (0.9), Flask (0.10)
- pycrypto (2.4), paramiko
- celery (2.4.0)
- Pillow (2.0.0)

# 2014년의 파이썬 3

- numpy (1.5.0), scipy (0.9.0)
- Django (1.5), South (0.8)
- Werkzeug (0.9), Flask (0.10)
- pycrypto (2.4), paramiko (1.13.0)
- celery (2.4.0)
- Pillow (2.0.0)



Python 3 Wall of Superpo... x +

https://python3wos.appspot.com

# PYTHON 3 WALL OF SUPERPOWERS

Python 3.0 was released December 3, 2008.  
As listed on PyPI - packages in **red** don't support python 3, packages in **green** do. Hopefully one day everything will be green.  
 Status: 165/200 Updated: 2014-08-24 02:17:08 UTC-0400

Package	Downloads
distribute	2872889
setuptools	2522799
requests	2187554
boto	2039646
pip	1832107
lxml	1781810
zlib	1626275
nose	1574876
python-defaults	1749328
simplejson py3k	1142554
pytz	1141284
jpeg2	1028548
Django py3k	882212
xmltodict	872881
xl py3k	848736
PyMySQL	827183
MarkupSafe	818895
coverage	783229
double	752114
Flask	724148
paramiko	718873
jika	687383
FlaskDeploy	627817
pymongo	622432
pyamf	603289
redis	578235
httplib2	571818
salabam	568819
pycrypto	551273
pycrypto2	521982
Werkzeug	548818
mock	542785
SQLAlchemy	538181
Pygments	520289
Flask	498402
MySQL-python	482382
pep8	481814
colorama	482884
statsinterface	423283
rsa	422237
kombu	422236
carbon	421784
superior	424372
Fabio	424372
mail2	418883
greenlet	408777
graphite-web	404321
lxml	381419
South	3887128
ecdsa	380380
anyjson	3783123
Spinn	3618487
celery	3482188
pycups	342380
honey	323813
PratyTable	3171146
WebOb	2783748
gunicorn	3182881
icu801	322388
Moku	298730
Flite	288472
setuptools-g2	288181

# 플랫폼, 2014

# 플랫폼, 2014

- 주요 리눅스 배포판에서 모두 파이썬 3를 패키지로 지원

# 플랫폼, 2014

- 주요 리눅스 배포판에서 모두 파이썬 3를 패키지로 지원
- 젠투, 아치 등은 아예 파이썬 3가 기본

# 플랫폼, 2014

- 주요 리눅스 배포판에서 모두 파이썬 3를 패키지로 지원
- 젠투, 아치 등은 아예 파이썬 3가 기본
- 아직 구글 앱 엔진은 파이썬 2.7만 지원하지만, 헤로쿠는 파이썬 3를 100% 지원함

# 플랫폼, 2014

- 주요 리눅스 배포판에서 모두 파이썬 3를 패키지로 지원
- 젠투, 아치 등은 아예 파이썬 3가 기본
- 아직 구글 앱 엔진은 파이썬 2.7만 지원하지만, 헤로쿠는 파이썬 3를 100% 지원함
- 아쉽지만 맥에서는 여전히 파이썬 2.5, 2.6, 2.7만 번들

# 플랫폼, 2014

- 주요 리눅스 배포판에서 모두 파이썬 3를 패키지로 지원
- 젠투, 아치 등은 아예 파이썬 3가 기본
- 아직 구글 앱 엔진은 파이썬 2.7만 지원하지만, 헤로쿠는 파이썬 3를 100% 지원함
- 아쉽지만 맥에서는 여전히 파이썬 2.5, 2.6, 2.7만 번들
- pyenv 등으로 설치하면 문제 L L

# 플랫폼, 2014

- 주요 리눅스 배포판에서 모두 파이썬 3를 패키지로 지원
- 젠투, 아치 등은 아예 파이썬 3가 기본
- 아직 구글 앱 엔진은 파이썬 2.7만 지원하지만, 헤로쿠는 파이썬 3를 100% 지원함
- 아쉽지만 맥에서는 여전히 파이썬 2.5, 2.6, 2.7만 번들
- pyenv 등으로 설치하면 문제 L L
- PyPy도 이제 파이썬 3 구현했음!



# 생태계 외의 문제

# 생태계 외의 문제

- 파이썬 3 문제의 대부분이 생태계였다면...

# 생태계 외의 문제

- 파이썬 3 문제의 대부분이 생태계였다면...
- 생태계 문제가 사라진 지금은 대체 어떤 문제가 남았을까?

# 생태계 외의 문제

- 파이썬 3 문제의 대부분이 생태계였다면...
- 생태계 문제가 사라진 지금은 대체 어떤 문제가 남았을까?
- 아마 그것은 내 자신이 싸질러 둔 똥코드?

# 생태계 외의 문제

- 파이썬 3 문제의 대부분이 생태계였다면...
- 생태계 문제가 사라진 지금은 대체 어떤 문제가 남았을까?
- 아마 그것은 내 자신이 싸질러 둔 똥코드?
- 그래도 새로 싸는 똥은 파이썬 3로 똥칠할 수 있다!

# 지오프론트의 경우

# 지오포론트의 경우

- 신규 프로젝트

# 지오프론트의 경우

- 신규 프로젝트
- 즉 파이썬 2로 쓰여진 레거시 코드 없음



# 지오프론트의 경우

- 신규 프로젝트
- 즉 파이썬 2로 쓰여진 레거시 코드 없음
- 서버이므로 최대한 여러 환경에서 돌아야 할 필요도 없음

# 지오프론트의 경우

- 신규 프로젝트
- 즉 파이썬 2로 쓰여진 레거시 코드 없음
- 서버이므로 최대한 여러 환경에서 돌아야 할 필요도 없음
- (지오프론트 CLI 클라는 파이썬 2.6, 2.7도 지원)

# 지오프론트의 경우

- 신규 프로젝트
- 즉 파이썬 2로 쓰여진 레거시 코드 없음
- 서버이므로 최대한 여러 환경에서 돌아야 할 필요도 없음
- (지오프론트 CLI 클라는 파이썬 2.6, 2.7도 지원)
- 의존하는 라이브러리는 어떨까?

**모두 파이썬 3 지원**

# 모두 파이썬 3 지원

- `paramiko` — SSH 라이브러리

# 모두 파이썬 3 지원

- paramiko — SSH 라이브러리
- Werkzeug, Flask — 웹 라이브러리

# 모두 파이썬 3 지원

- `paramiko` — SSH 라이브러리
- `Werkzeug`, `Flask` — 웹 라이브러리
- `apache-libcloud` — IaaS 클라이언트 라이브러리

# 모두 파이썬 3 지원

- `paramiko` — SSH 라이브러리
- `Werkzeug`, `Flask` — 웹 라이브러리
- `apache-libcloud` — IaaS 클라이언트 라이브러리
- `waitress` — WSGI 서버 (웹 서버)



# 모두 파이썬 3 지원

- `paramiko` — SSH 라이브러리
- `Werkzeug`, `Flask` — 웹 라이브러리
- `apache-libcloud` — IaaS 클라이언트 라이브러리
- `waitress` — WSGI 서버 (웹 서버)
- `pytest` — 단위 테스트 프레임워크

# 모두 파이썬 3 지원

- **paramiko** — SSH 라이브러리
- **Werkzeug, Flask** — 웹 라이브러리
- **apache-libcloud** — IaaS 클라이언트 라이브러리
- **waitress** — WSGI 서버 (웹 서버)
- **pytest** — 단위 테스트 프레임워크
- **Sphinx** — 문서화 도구

**실제로 썼던 신박한 기능들**

# 실제로 썼던 신박한 기능들

- PEP 405 — 빌트인 가상 환경 (pyvenv)

# 실제로 썼던 신박한 기능들

- PEP 405 — 빌트인 가상 환경 (pyvenv)
- PEP 453 — 빌트인 pip

# 실제로 썼던 신박한 기능들

- PEP 405 — 빌트인 가상 환경 (pyvenv)
- PEP 453 — 빌트인 pip
- PEP 3107 — 함수 어노테이션

# 실제로 썼던 신박한 기능들

- PEP 405 — 빌트인 가상 환경 (pyvenv)
- PEP 453 — 빌트인 pip
- PEP 3107 — 함수 어노테이션
- PEP 433 — `functools.singledispatch`

# 실제로 썼던 신박한 기능들

- PEP 405 — 빌트인 가상 환경 (pyvenv)
- PEP 453 — 빌트인 pip
- PEP 3107 — 함수 어노테이션
- PEP 433 — `functools.singledispatch`
- `datetime.timezone`



# PEP 405 — 빌트인 가상 환경

# PEP 405 — 빌트인 가상 환경

- 파이썬 처음 세팅할 때 맨날 하던 짓

# PEP 405 — 빌트인 가상 환경

- 파이썬 처음 세팅할 때 맨날 하던 짓
- `easy_install pip`

# PEP 405 — 빌트인 가상 환경

- 파이썬 처음 세팅할 때 맨날 하던 짓
- `easy_install pip`
- `pip install virtualenv`

# PEP 405 — 빌트인 가상 환경

- 파이썬 처음 세팅할 때 맨날 하던 짓
- `easy_install pip`
- `pip install virtualenv`
- `virtualenv project-name-env`

# PEP 405 — 빌트인 가상 환경

- 파이썬 처음 세팅할 때 맨날 하던 짓
- `easy_install pip`
- `pip install virtualenv`
- `virtualenv project-name-env`
- 이제 그냥 내장된 `pyvenv` 명령어 쓰면 됨

# PEP 405 — 빌트인 가상 환경

- 파이썬 처음 세팅할 때 맨날 하던 짓
- `easy_install pip`
- `pip install virtualenv`
- `virtualenv project-name-env`
- 이제 그냥 내장된 `pyenv` 명령어 쓰면 됨
- `pyenv project-name-env`

# PEP 453 — 빌트인 pip



# PEP 453 — 빌트인 pip

- 더이상 파이썬 처음 세팅할 때 `setuptools`, `pip` 설치 안해도 됨

# PEP 453 — 빌트인 pip

- 더이상 파이썬 처음 세팅할 때 `setuptools`, `pip` 설치 안해도 됨
- Trivia: `pip`는 내부적으로 `requests`, `html5lib` 등을 사용

# PEP 453 — 빌트인 pip

- 더이상 파이썬 처음 세팅할 때 `setuptools`, `pip` 설치 안해도 됨
- Trivia: `pip`는 내부적으로 `requests`, `html5lib` 등을 사용
- `pip._vendor.{requests,html5lib}` 구경해보세요

# PEP 3107 — 함수 어노테이션

```
class Team:
    """Backend interface for team membership auth..."""
    @typed
    def authenticate(self,
                     auth_nonce: str,
                     requested_redirect_url: str,
                     wsgi_environ: collections.abc.Mapping
                     ) -> Identity:
        """Second step of authentication process, ..."""
```

PEP 433: *singledispatch*

# PEP 433: singledispatch

- 제너릭 메서드, 혹은 제너릭 함수라고 부르는 것

# PEP 433: singledispatch

- 제너릭 메서드, 혹은 제너릭 함수라고 부르는 것
- 내가 구현하지 않은 클래스 계층 구조에 메서드를 추가하고 싶을 때 사용

# PEP 433: singledispatch

- 제너릭 메서드, 혹은 제너릭 함수라고 부르는 것
- 내가 구현하지 않은 클래스 계층 구조에 메서드를 추가하고 싶을 때 사용
- 혹은, 클래스 자체의 인터페이스로 넣기 애매한 기능들을 따로 함수로 빼면서도 다형성을 그대로 사용할 수 있음



# PEP 433: singledispatch

- 제너릭 메서드, 혹은 제너릭 함수라고 부르는 것
- 내가 구현하지 않은 클래스 계층 구조에 메서드를 추가하고 싶을 때 사용
- 혹은, 클래스 자체의 인터페이스로 넣기 애매한 기능들을 따로 함수로 빼면서도 다형성을 그대로 사용할 수 있음
- 다만, 다중 디스패치는 지원 안함 (자주 필요한 건 아님)

# PEP 433: singledispatch

```
@singledispatch

def get_metadata(driver: NodeDriver,
                 node: Node) -> collections.abc.Mapping:
    return driver.ex_get_metadata(node)

@get_metadata.register(GCENodeDriver)
def gce_get_metadata(driver: GCENodeDriver,
                    node: Node
                    ) -> collections.abc.Mapping:
    return node.extra['metadata']
```

`datetime.timezone`

# datetime.timezone

- 표준 datetime 모듈에는 여러 불편한 점이 많은데

# datetime.timezone

- 표준 datetime 모듈에는 여러 불편한 점이 많은데
- 사실 그 중 많은 부분은 파이썬 3에서 이미 몇년 전에 고쳐진 사항들

# datetime.timezone

- 표준 datetime 모듈에는 여러 불편한 점이 많은데
- 사실 그 중 많은 부분은 파이썬 3에서 이미 몇년 전에 고쳐진 사항들
- 파이썬 2가 늙어가고 있다는 것을 이걸 사용하면서 느낌

# datetime.timezone

- 표준 datetime 모듈에는 여러 불편한 점이 많은데
- 사실 그 중 많은 부분은 파이썬 3에서 이미 몇년 전에 고쳐진 사항들
- 파이썬 2가 늙어가고 있다는 것을 이걸 사용하면서 느낌
- 그 중 하나가 시간대 인터페이스(tzinfo)만 제공할 뿐 시간대의 구현은 제공하지 않는다는 것

# datetime.timezone

- 표준 datetime 모듈에는 여러 불편한 점이 많은데
- 사실 그 중 많은 부분은 파이썬 3에서 이미 몇년 전에 고쳐진 사항들
- 파이썬 2가 늙어가고 있다는 것을 이걸 사용하면서 느낌
- 그 중 하나가 시간대 인터페이스(tzinfo)만 제공할 뿐 시간대의 구현은 제공하지 않는다는 것
- 심지어 UTC조차도!



`datetime.timezone`

# datetime.timezone

- 그런데 파이썬 3에서는 이미 고쳐져 있었던 것이다

# datetime.timezone

- 그런데 파이썬 3에서는 이미 고쳐져 있었던 것이다
- `datetime.timezone.utc`

# datetime.timezone

- 그런데 파이썬 3에서는 이미 고쳐져 있었던 것이다
- `datetime.timezone.utc`
- `timezone(timedelta(hours=9), 'KST')`

# datetime.timezone

- 그런데 파이썬 3에서는 이미 고쳐져 있었던 것이다
- `datetime.timezone.utc`
- `timezone(timedelta(hours=9), 'KST')`
- 매 프로젝트마다 `myproject.tz` 모듈 만들던 것  
더이상 안해도 됨

**실제로 썼던 신박한 기능들**

# 실제로 썼던 신박한 기능들

- PEP 3132 — 언패킹에서 \* 연산자 사용

# 실제로 썼던 신박한 기능들

- PEP 3132 — 언패킹에서 \* 연산자 사용
- PEP 344 — 안쪽 예외 보여주기



# 실제로 썼던 신박한 기능들

- PEP 3132 — 언패킹에서 \* 연산자 사용
- PEP 344 — 안쪽 예외 보여주기
- PEP 435 — 열거형

# 실제로 썼던 신박한 기능들

- PEP 3132 — 언패킹에서 \* 연산자 사용
- PEP 344 — 안쪽 예외 보여주기
- PEP 435 — 열거형
- `urllib.request.Request.method`

PEP 3132: 언패킹에서 \* 연산자

# PEP 3132: 언패킹에서 \* 연산자

- 파이썬 2에서도 함수 인자에는 \* 연산자를 쓸 수 있음

# PEP 3132: 언패킹에서 \* 연산자

- 파이썬 2에서도 함수 인자에는 \* 연산자를 쓸 수 있음
- 이제는 튜플 언패킹 등에서도 똑같이 쓸 수 있다는 것

# PEP 3132: 언패킹에서 \* 연산자

- 파이썬 2에서도 함수 인자에는 \* 연산자를 쓸 수 있음
- 이제는 튜플 언패킹 등에서도 똑같이 쓸 수 있다는 것
- `fst, snd, *rest = get_list()`

# PEP 3132: 언패킹에서 \* 연산자

- 파이썬 2에서도 함수 인자에는 \* 연산자를 쓸 수 있음
- 이제는 튜플 언패킹 등에서도 똑같이 쓸 수 있다는 것
- `fst, snd, *rest = get_list()`
- `fst, snd, *mid, last = get_list()`

# PEP 344 — 안쪽 예외 표시

```
def a():  
    raise ValueError('something went wrong')  
  
def b():  
    try:  
        a()  
    except ValueError:  
        raise RuntimeError('crashed')  
  
b()
```



# PEP 344 — 안쪽 예외 표시

# PEP 344 — 안쪽 예외 표시

- 예외를 잡아서 다른 형태의 예외 타입으로 다시 던질 때

# PEP 344 — 안쪽 예외 표시

- 예외를 잡아서 다른 형태의 예외 타입으로 다시 던질 때
- 파이썬 2에서는 에러 메시지에 바깥쪽의 스택 트레이스만 표시됨

# PEP 344 — 안쪽 예외 표시

- 예외를 잡아서 다른 형태의 예외 타입으로 다시 던질 때
- 파이썬 2에서는 에러 메시지에 바깥쪽의 스택 트레이스만 표시됨
- 디버그할 때 해당 스택까지 들어가서 안쪽에서 실제로는 어떤 예외가 난 건지 직접 찾아봐야함

# PEP 344 — 안쪽 예외 표시

- 예외를 잡아서 다른 형태의 예외 타입으로 다시 던질 때
- 파이썬 2에서는 에러 메시지에 바깥쪽의 스택 트레이스만 표시됨
- 디버그할 때 해당 스택까지 들어가서 안쪽에서 실제로는 어떤 예외가 난 건지 직접 찾아봐야함
- 파이썬 3에서는 안쪽에서 난 오류도 같이 보여줌!

# PEP 344 — 안쪽 예외 표시

```
Traceback (most recent call last):
```

```
File "t.py", line 6, in b
```

```
    a()
```

```
File "t.py", line 2, in a
```

```
    raise ValueError('something went wrong')
```

```
ValueError: something went wrong
```

```
During handling of the above exception, another exception occurred:
```

```
Traceback (most recent call last):
```

```
File "t.py", line 10, in <module>
```

```
    b()
```

```
File "t.py", line 8, in b
```

```
    raise RuntimeError('crashed')
```

```
RuntimeError: crashed
```

# PEP 435 — 열거형

```
class KeyType(enum.Enum):  
    """SSH key types."""  
    ecdsa_ssh2_nistp256 = 'ecdsa-sha2-nistp256'  
    ecdsa_ssh2_nistp384 = 'ecdsa-sha2-nistp384'  
    ecdsa_ssh2_nistp521 = 'ecdsa-sha2-nistp521'  
    ssh_dss = 'ssh-dss'  
    ssh_rsa = 'ssh-rsa'
```

# PEP 435 — 열거형

```
>>> list(KeyType)

[KeyType.ecdsa_ssh2_nistp256, KeyType.ecdsa_ssh2_nistp384,
KeyType.ecdsa_ssh2_nistp521, KeyType.ssh_dss,
KeyType.ssh_rsa]

>>> KeyType.ssh_rsa

KeyType.ssh_rsa

>>> KeyType.ssh_rsa.name

'ssh_rsa'

>>> KeyType.ssh_rsa.value

'ssh-rsa'
```



**Request.method**

# Request.method

- (파이썬 3에서 urllib2가 urllib.request로 이름 변경됨)

# Request.method

- (파이썬 3에서 urllib2가 urllib.request로 이름 변경됨)
- 기존에는 urllib2에서 GET, POST 메서드만 사용 가능

# Request.method

- (파이썬 3에서 urllib2가 urllib.request로 이름 변경됨)
- 기존에는 urllib2에서 GET, POST 메서드만 사용 가능
- 이제는 PUT, DELETE 등 다른 메서드도 사용 가능!

# Request.method

- (파이썬 3에서 urllib2가 urllib.request로 이름 변경됨)
- 기존에는 urllib2에서 GET, POST 메서드만 사용 가능
- 이제는 PUT, DELETE 등 다른 메서드도 사용 가능!
- PUT 하나 보내자고 requests 갈아서 쓰던 짓은 안녕

# Request.method

- (파이썬 3에서 urllib2가 urllib.request로 이름 변경됨)
- 기존에는 urllib2에서 GET, POST 메서드만 사용 가능
- 이제는 PUT, DELETE 등 다른 메서드도 사용 가능!
- PUT 하나 보내자고 requests 깎아서 쓰던 짓은 안녕
- 그냥 파이썬 2가 정상이지 아니었던 것 같지만

**파이썬 2, 이제 성불시킵시다**

# 파이썬 2, 이제 성불시킵시다

- 파이썬 2는 점점 더 파이썬 세상의 혜택을 못 보게 될 것



# 파이썬 2, 이제 성불시킵시다

- 파이썬 2는 점점 더 파이썬 세상의 혜택을 못 보게 될 것
- 이미 파이썬 2는 많은 불이익을 받고 있음

# 파이썬 2, 이제 성불시킵시다

- 파이썬 2는 점점 더 파이썬 세상의 혜택을 못 보게 될 것
- 이미 파이썬 2는 많은 불이익을 받고 있음
- 예: PEP 466 — Python 2.7.x을 위한 네트워크 보안 강화

# 파이썬 2, 이제 성불시킵시다

- 파이썬 2는 점점 더 파이썬 세상의 혜택을 못 보게 될 것
- 이미 파이썬 2는 많은 불이익을 받고 있음
- 예: PEP 466 — Python 2.7.x을 위한 네트워크 보안 강화
- 성장이 없는 파이썬 2, 이제 놓아줍시다!

spoqa

# WE'RE HIRING

즐겁게 개발하는 기쁨을 함께 나눌 동료를 찾습니다  
<http://spoqa.github.io/job.html>

