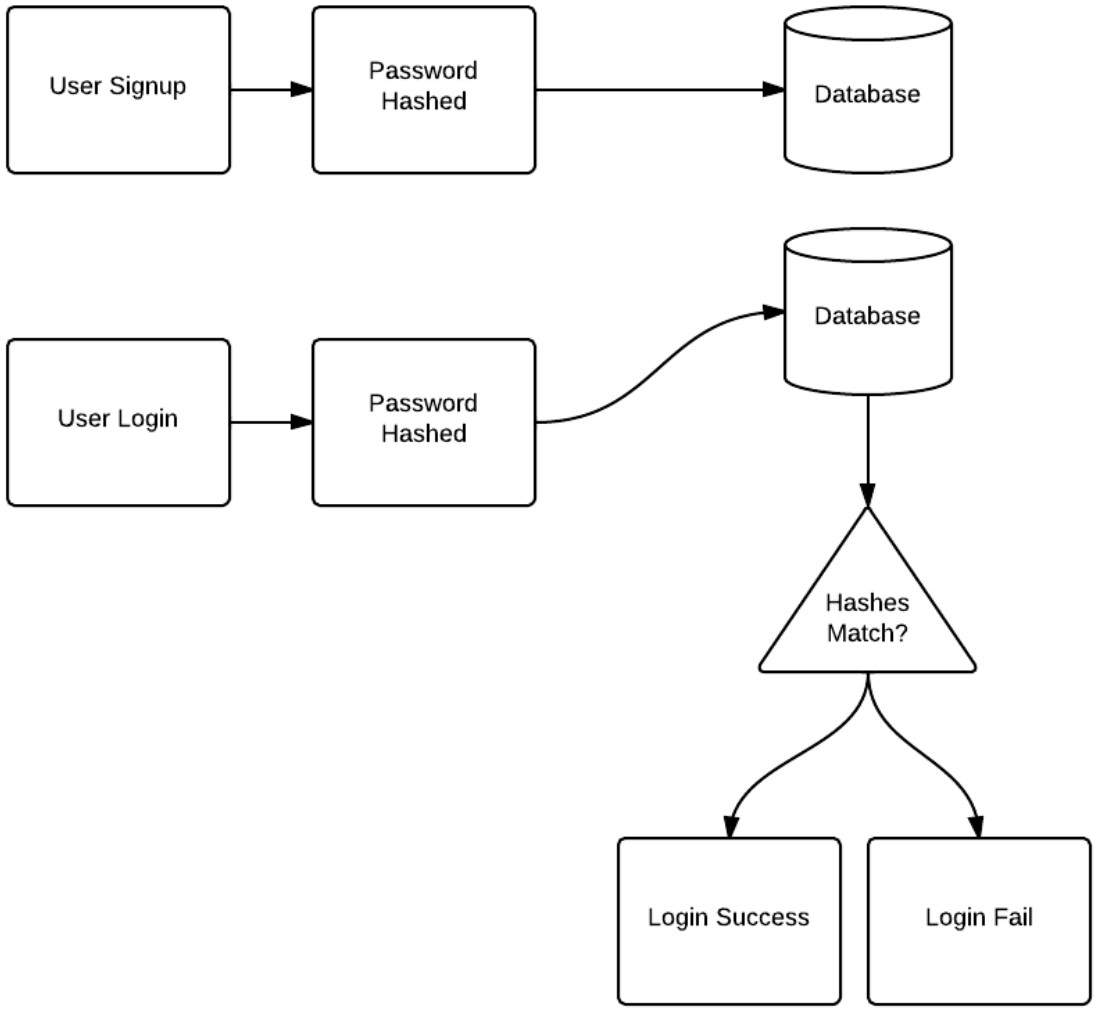
SALT

박재훈

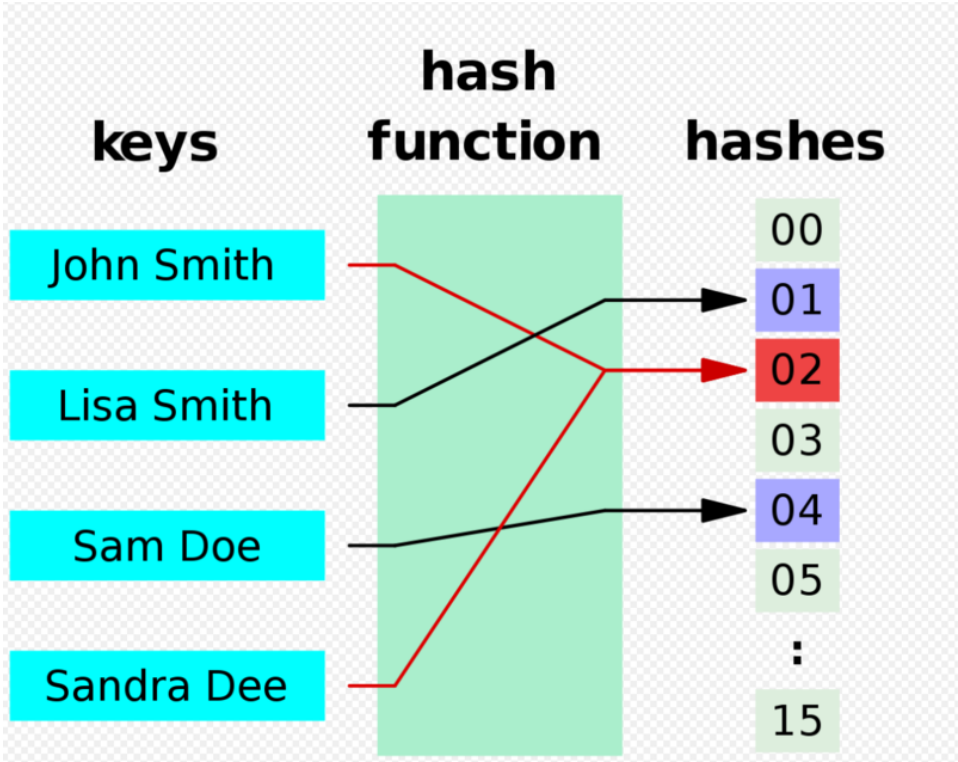
https://youtu.be/Wdp14qNVsoo

사용자 가입 및 인증


User Signup
Password Hashed
Database
User Login
Password Hashed
Database
Hashes Match?
Login Success
Login Fail

단방향 해시 함수

- 해시 함수의 원래 용도는 해시 테이블을 위한 것.
- 사용자의 비밀번호 관리는 대부분 단방향 해시 함수를 통해서 관리됨.
- 해시값을 안다고 해서 원래 값을 알 수는 없음.
- SHA256, SHA512, RipeMD 등이 대표적.



단방향 해시 함수

1비트만 달라져도 전체 값이 완전히 바뀜. (눈사태 효과)

- SHA256)
- 사용자 패스워드가 ‘hello1’일 경우,
  해시값 : 91e9240f415223982edc345532630710e94a7f52cd5f48f5ee1afc555078f0ab
- 사용자 패스워드가 ‘hello2’일 경우,
  해시값 : 87298cc2f31fba73181ea2a9e6ef10dce21ed95e98bdac9c4e1504ea16f486e4

단방향 해시 함수

- input이 동일하면 output도 항상 동일함.

- 이 원리를 이용하여 공격할 수 있음.

단어 사전 공격

- 비밀번호로 쓰일 만한 단어들을 조합하여 문자열을 생성.

- 효율이 좋지 않음.

무차별 대입 방법

- 비밀번호 길이에 맞춰 알파벳, 숫자 등을 조합하여 무작위 문자열을 만든 뒤 공격.

- 마찬가지로 효율이 좋지 않지만, 언젠가는 성공.

룩업 테이블

- 비밀번호 사전으로부터 해시값들을 추출해놓고, 그 안에서 비밀번호를 검색.

- 초당 백 개 정도의 비밀번호를 검색할 수 있으며 데이터가 수십억 개가 넘더라도 사용 가능.

역 룩업 테이블

- 비밀번호가 동일한 (해시값이 동일한) 사용자들끼리 그룹핑.

- 같은 비밀번호를 사용하는 사용자가 많기에 효율적.

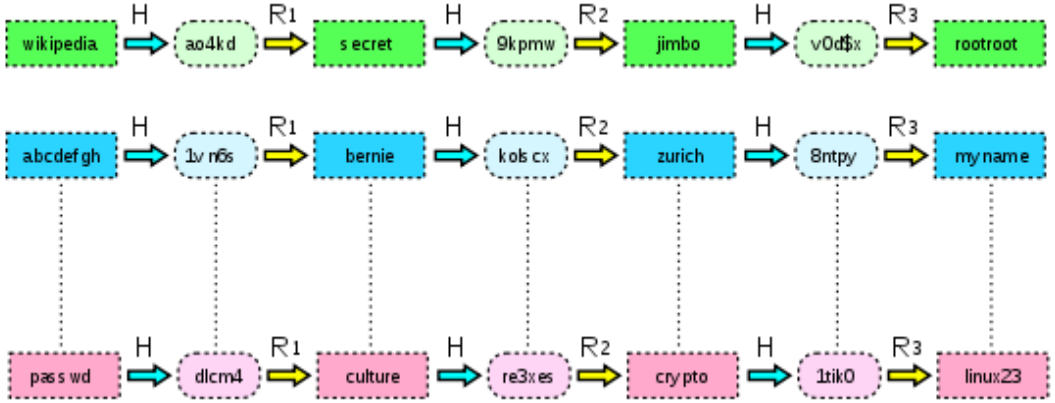
레인보우 테이블

1. 사용자가 비밀번호로 쓸 것 같은 단어들을 미리 선정해 놓음.

2. 그 단어들에 대한 해시값을 구함.

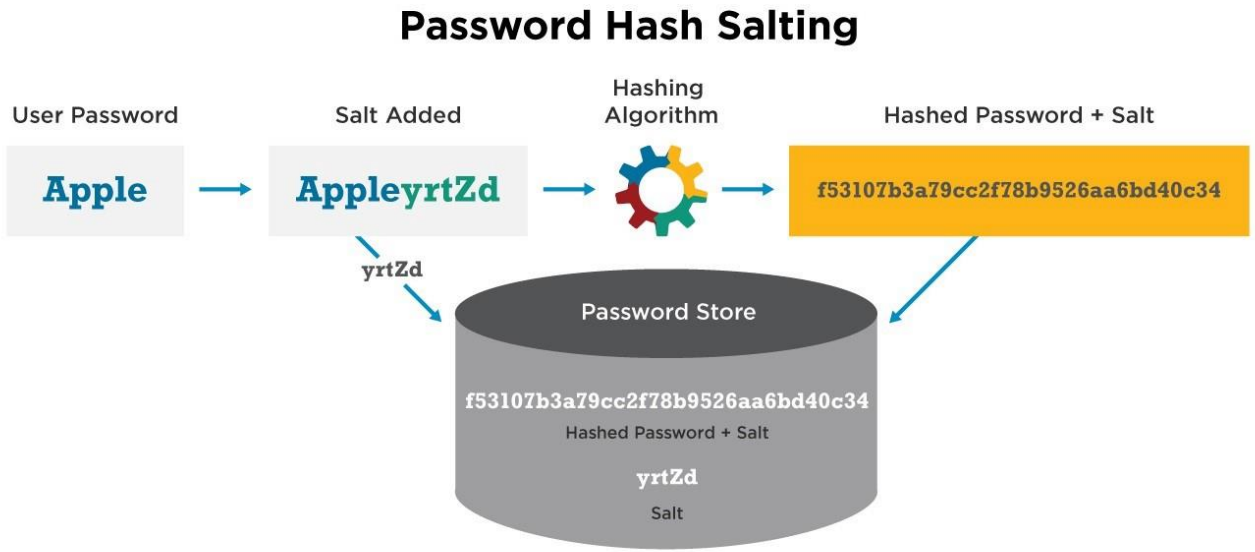
3. 해시값으로부터 R 함수를 통해 특정 단어들을 추출해냄.

4. 위 과정을 반복하면서 테이블을 채워 넣음.



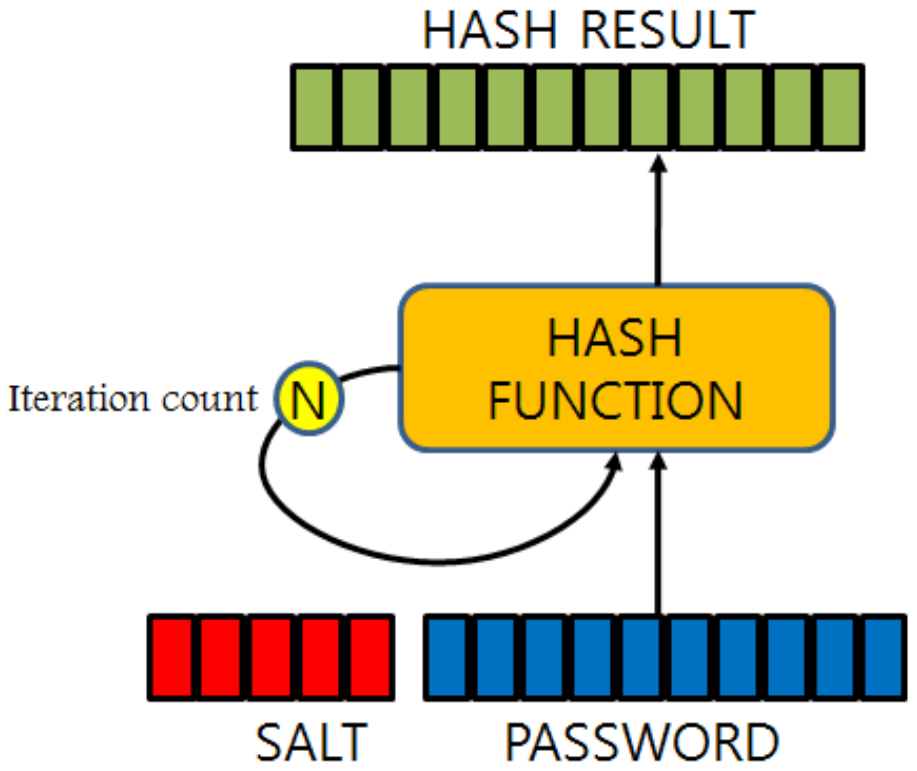
솔트 (Salt)

- 레인보우 테이블 등의 추측성 대입 공격을 막기 위한 방법.
- 사용자 비밀번호 앞/뒤에 랜덤 문자열을 붙인 뒤 해싱.
- 생성된 솔트값은 별도의 공간에 저장하여 인증 시에만 사용.



키 스트레칭

- 패스워드의 해시값을 또 해싱하고, 이 과정을 반복하여 결과값을 얻음.
- 억지 기법 공격(Brute Force Attack)을 방지



Functions

- PBKDF2
- bcrypt
- scrypt

bcrypt

```
const hash = async pw => {
  const salt = await bcrypt.genSalt();
  const hash = await bcrypt.hash(pw, salt);
  return hash;
};

const pw = 'hello';

for (let i=0; i<10; i++) {
  const digest = await hash(pw);
  console.log(`no.${i} digest: ${digest}`);
}
```

```
no.0 digest: $2a$10$dPgXK87BWwufVjv6XO.ej.IR0gP9RN6ovJ8wNua/VbCS/JBWnLRFS
no.1 digest: $2a$10$MHP5mvQOTC.J4UNJeBH2EOTMhinMjV1A9rmm4ZCwYvQNNc/elQfyC
no.2 digest: $2a$10$FRog4pC1711vu75slrZHROt9ljweNcEN0QVQrJI9l3rNOlo5jyJT2
no.3 digest: $2a$10$g3prYZf0VFBakNAlhcBvHuk8Ra1KluZJaiJDL2YroCL65V/y4L6jy
no.4 digest: $2a$10$6S2Ot7VvGAd8jtt2/DolneegI0qtbFdC0coBPRQO48Kmr9Ur2yBhW
no.5 digest: $2a$10$NrU6Bc2KmnfRB.xCMqn8hut1x.6cDyuGRyVNc92MU5LBz5m.1yhv6
no.6 digest: $2a$10$CshKo4ohUfQqAU/SuXo.P.C0.98wuwLVyhlyHXvZyioDHjs2H8Ou2
no.7 digest: $2a$10$2Ynla/HDUGuzG.vdqFlNEOKTU5XHuRcahqxs3QHVbQoSrAX8spVba
no.8 digest: $2a$10$TeKvEGX0JXrnzWN2rA.b/uW1m65t0bh8n5ab0Xe7nIXcDHCWUjJ/i
no.9 digest: $2a$10$IGvE9.gZ..1sQENTYgczPeV9dHM1uWKi9KJ.DxFyI2nW.NgJyyPW.
```

Q & A