Applications of One-Way Hash Functions

- Integrity Verification — Detecting when data has been altered
- Commitments — Committing a secret without telling it
- **Password Verification** — Verifying a password without storing the plaintext

User Device

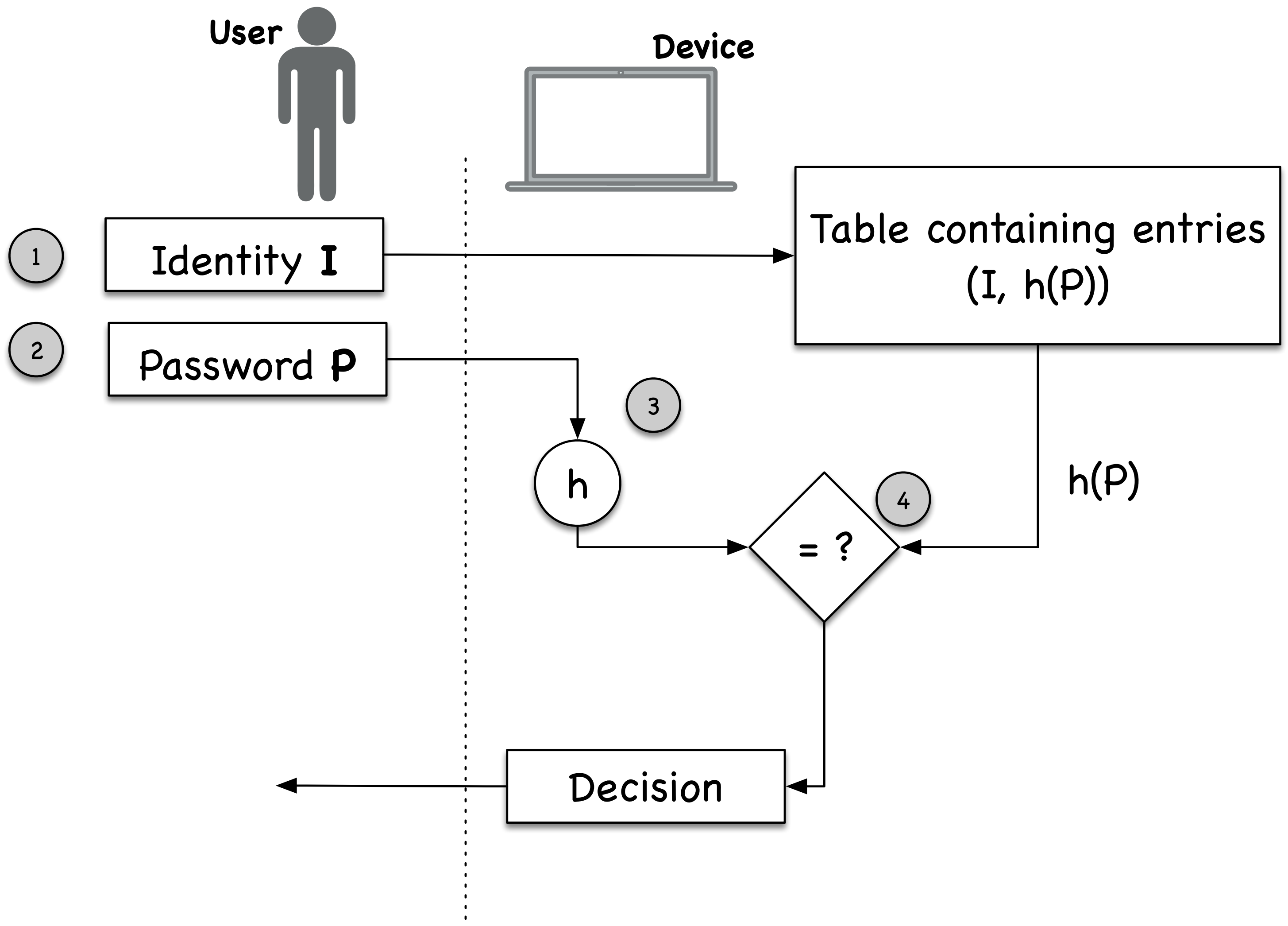
Password Verification

- To login into account, user needs to know the secret (password)
- Should <u>**never**</u> store the secrets in their plaintext form
- **Requirements:**
  - Password storage where nobody can know what the password is
  - If provided with a password, it verified against the stored password
- **Solution:** store hash of password using one-way hash function

```
$ sudo cat /etc/shadow
root:$6$NrF46O1p$.vDnKEtVFC2bXsl ...(omitted)... spr/kqzAqtcu.:17400:0:99999:7:::
...
seed:$6$wDRrWCQz$IsBXp9.9wz9SGrF ...(omitted)... J8sbCT7hkxXY/:17372:0:99999:7:::
john:$6$6MiP8itO$uFVUFX8qZnxcIUD ...(omitted)... Fz/biD8mR7an.:18290:0:99999:7:::
newseed:$6$ZPwHFy.m$tKETCWrzE6WL ...(omitted)... cDsSgSm4TNRrf:18290:0:99999:7:::
```

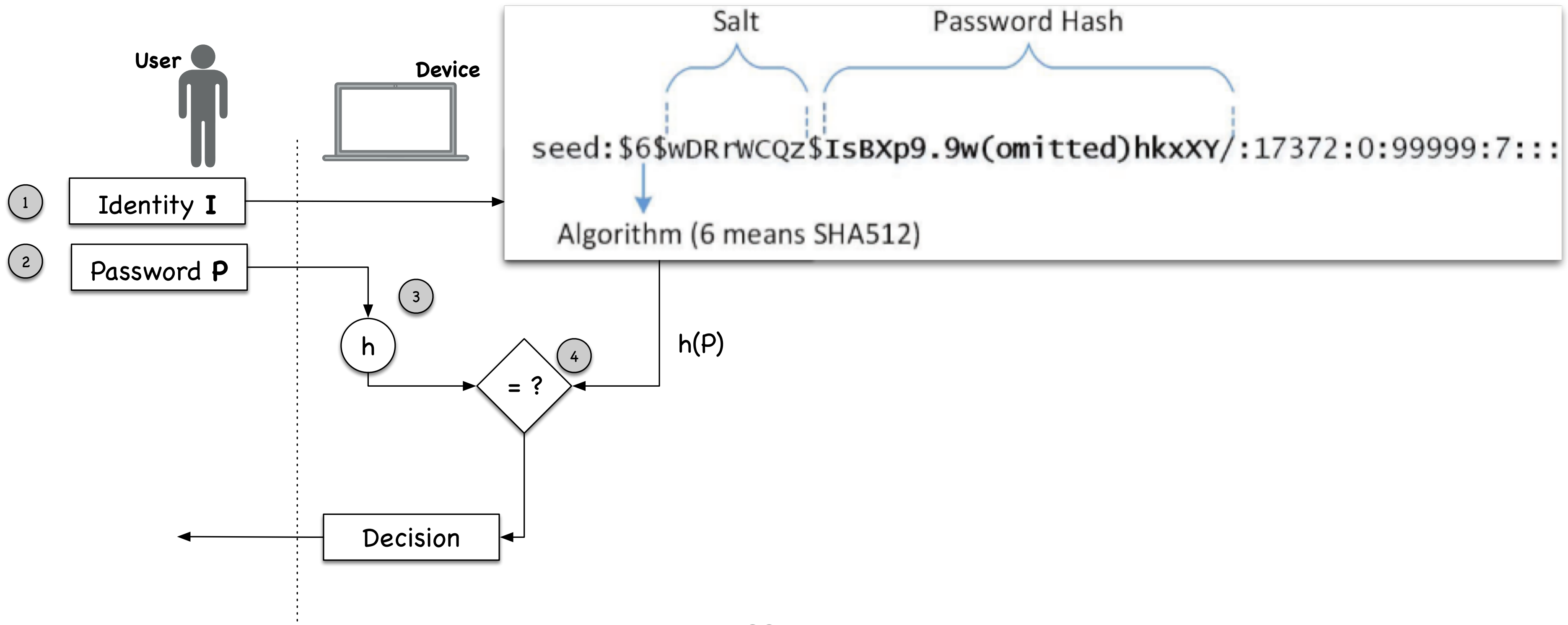
Example: Linux stores passwords in the /etc/shadow file

Password Verification



Password Verification

- Password field has 3 parts: the **algorithm** used, **salt**, **password hash**
- Salt and password hash are encoded into printable characters (e.g., base64)
- Multiple rounds of hash function -> slow down brute-force attack



```
seed:$6$wDRrWCQz$IsBXp9.9w(omitted)hkxXY/:17372:0:99999:7:::
```

Purpose of Salt

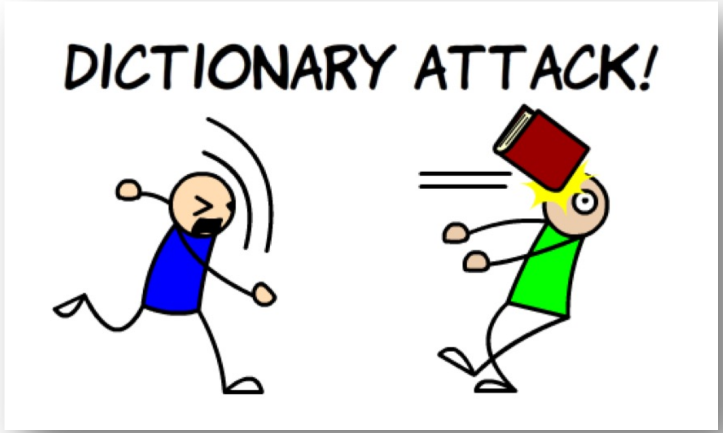
So what is the purpose of a “salt”?

- Salt is nothing more than a random value (string)
- Using salt, the same input can result in different hashes
- Password hash = one-way hash rounds (password || random string)

```
$ python
>>> import crypt
>>> print crypt.crypt('dees', '$6$wDRrWCQz')
$6$wDRrWCQz$IsBXp9.9wz9SGrF  ...(omitted)... J8sbCT7hkxXY/
```

```
$ sudo cat /etc/shadow
...
seed:$6$wDRrWCQz$IsBXp9.9wz9SGrF ...(omitted)... J8sbCT7hkxXY/:17372:0:99999:7:::
...
```

Attacks Prevented by Salt



Dictionary Attack

- Put candidate words in a dictionary
- Try each against the targeted password hash to find a match

Rainbow Table Attack

- Precomputed table for reversing cryptographic hash functions

How Does A Salt Prevent These Attacks?

- If target password is same as precomputed data, the hash will be the same
- If this property does not hold, all the precomputed data are useless
- Salt destroys that property