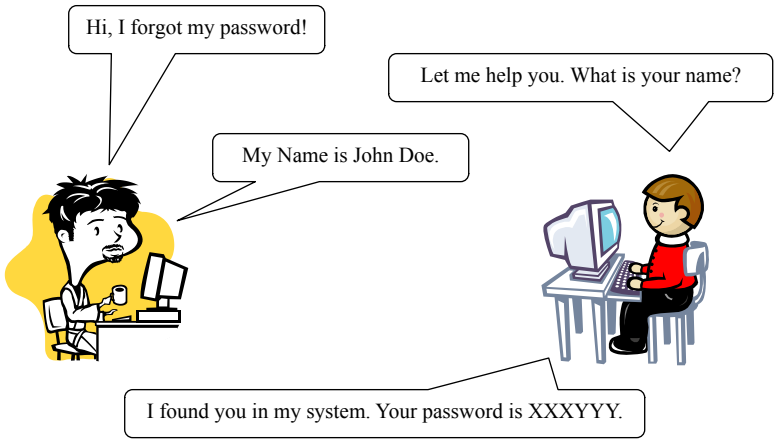
Security: Some Fun with Passwords

- How to handle Passwords: the Basics
- Rainbow Attacks

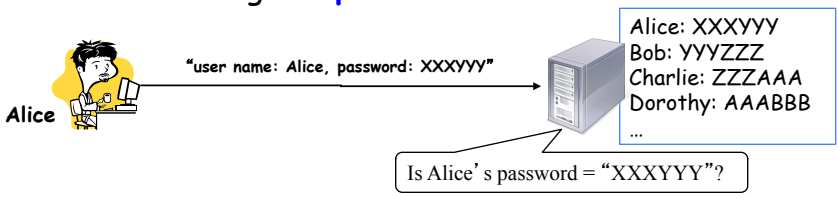
Not too long ago…



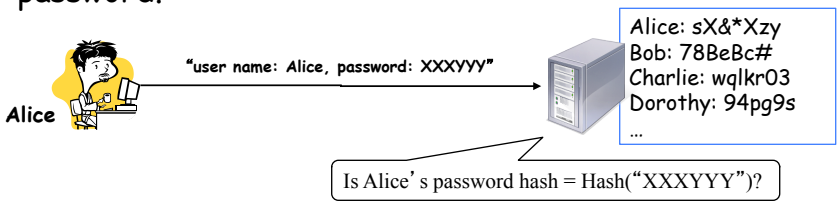
Q: What is wrong with this scenario?

Password Hashing

Instead of storing the **password** on the server …

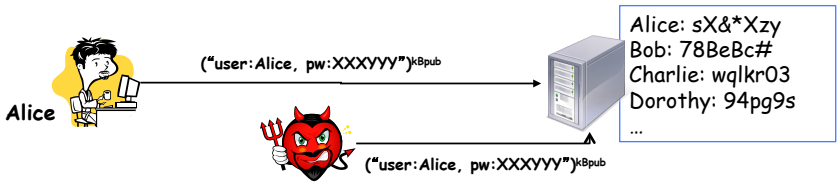


… we store an encrypted (**hashed**) version of the password.

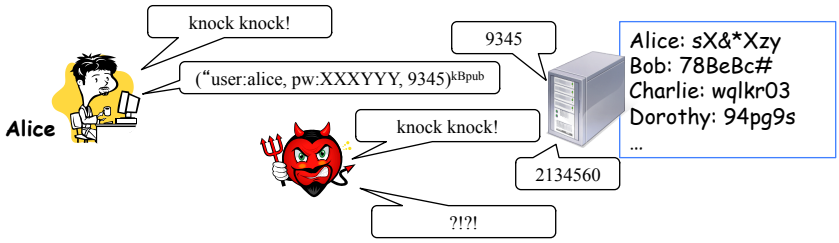


Replay Attacks and Challenge Response

Simply encrypting a request does not protect from replays.

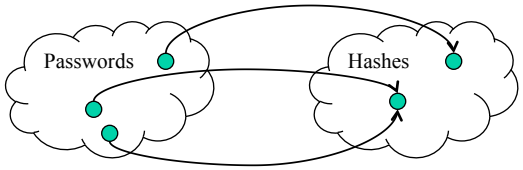


Solution: Challenge-response.



Is Password Hashing Overrated?
(or, hacking password files using rainbow tables)

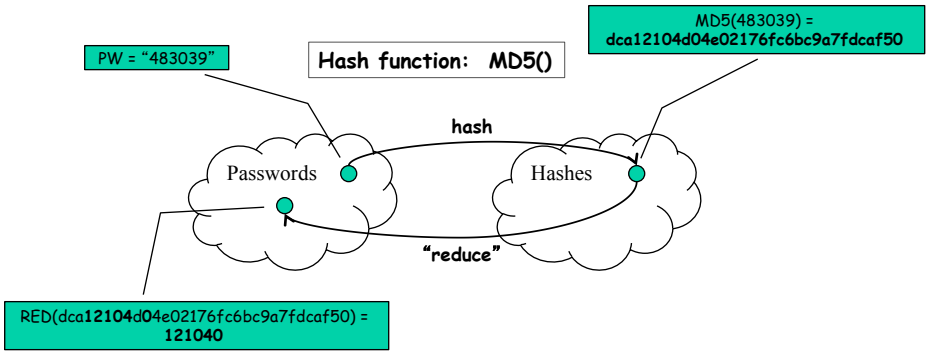
Password Hashing:



Two approaches to "Decrypt" passwords:

1. Exhaustively generate and hash passwords and check for match.
2. Generate table for all possible passwords and their hashes. Then just look up.

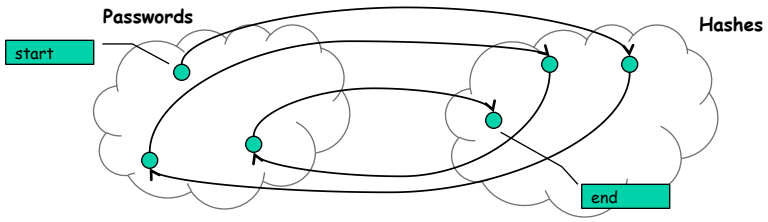
Rainbow Tables: Reduction Functions



Reduction function:

RED(arg) := pick first numerical 6 digits of arg.

"Chains" of Hashes



Simple Hash Chain Table:

start	end
iaisudhiu	4259cc34599c530b1e4a8f225d665802
oxcvioix	c744b1716cbf8d4dd0ff4ce31a177151
9da8dasf	3cd696a8571a843cda453a229d741843
[…]	
sodifo8sf	7ad7d6fa6bb4fd28ab98b3dd33261e8f

Problems with Hash Chains

- Chains can **collide**:
  - When hash function or reduction values collide, hash chains **merge**.
  - Hash function values are unlikely to collide
  - Reduction function values are **likely** to collide
- Reduction function should map back to **likely subset** of passwords.
  - If not, we are spending time scanning the entire space.

How to Counter such Attacks? Salting

Instead of storing the hash

```
hash(password)
```

we store the salted hash

```
hash(password + salt)
```

where salt is a very large number.