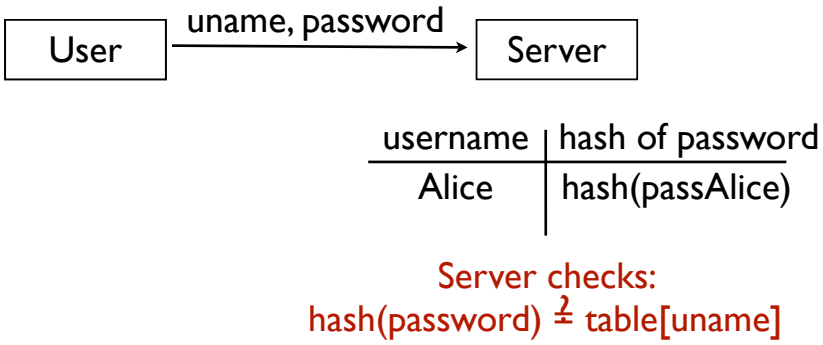
Recitation 2, Part 2 Notes: Passwords
Feb 22, 2013

- Passwords are used for authentication. A user Alice authenticates to a server, if the server has a way to check that it is indeed communicating with Alice.
- **Security goal:** adversary can authenticate only if he guesses a password.
  - The protocol used should not make it any easier for the adversary to authenticate than guessing the password. For example, a protocol that posts publicly user passwords is a bad protocol because it suffices for an adversary to look up the password of interest and authenticate and does not have to go through all possible guesses.



- What properties do we need from the hash function to achieve the security goal?
- It depends *on what information the adversary has* available
- Case 1: The adversary knows the username, but knows nothing about the password and *does not know the hash.*
  - One-wayness is not needed: Consider that the hash function is the identity function. It is clearly not one-way. In this situation, the adversary can only authenticate himself if he guesses the password.
  - Collision resistance is not needed: Consider the hash function that maps every two consecutive values to the same hash value. It is easy to exhibit a collision for any given value (just exhibit its neighboring value). However, in this case as well, the adversary has to guess the password (or its neighboring value).
  - Consider a hash function that maps all values to 1. This is clearly bad because the adversary can choose any password, and succesfully authenticate because the hash of the value chosen will be 1 which will match the hash of any other password, including the one of Alice.

  - We want "pseudorandomness". Namely, we want the values of the hash to be close to being uniformly distributed in the range of values. Now assuming that Alice chose her password randomly, the adversary effectively has to guess one of a small fraction of the passwords that map to the same hash.

- Case 2: The adversary was able to steal the table from the server, so he *knows* the usernames *and hashes*, but he knows nothing else about the hashes
  - One-wayness: given any hash, the adversary cannot come up with a password that matches that hash.
  - This is the more interesting case and a reason why hashes are employed for passwords.