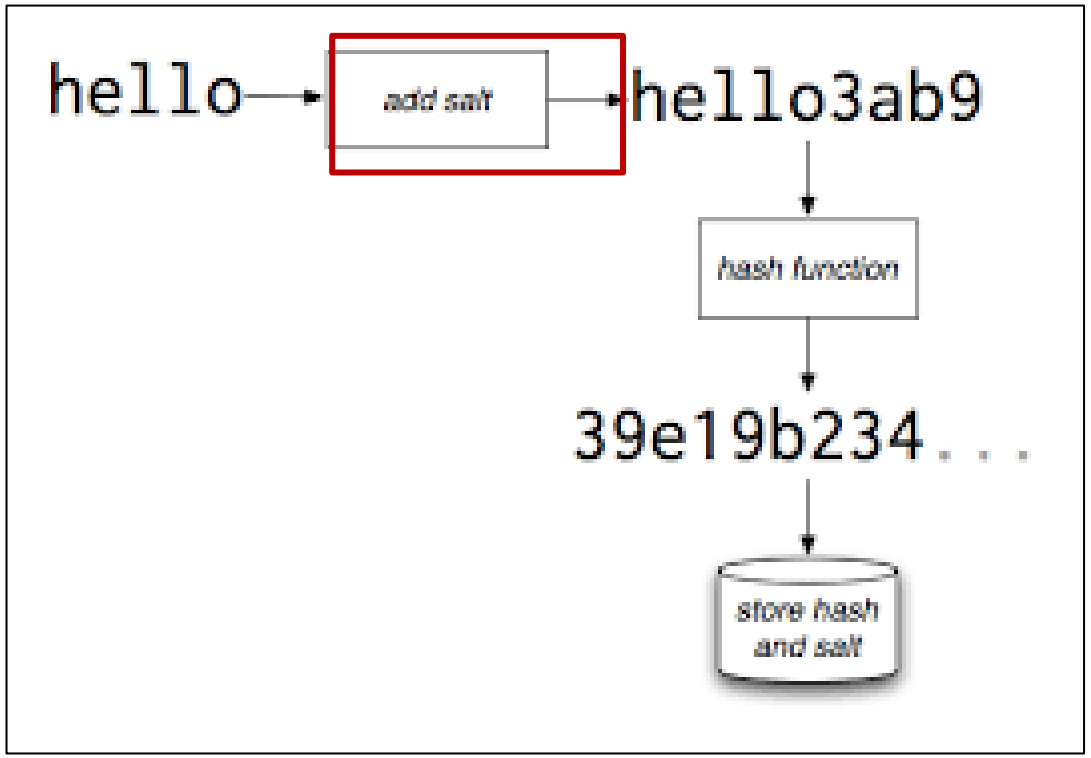
PASSWORD SALTING

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Salting

Illustration



- To reduce the effectiveness of offline attacks using pre-computed hashes, a *salt* is added to a password before applying the hash function.
- A salt is just a random string.
- Each password has its own salt.
- The salt value is stored along with the hash of password+salt.
- For a salt of n-bit, the attacker needs to pre-compute 2^n of hashes for *the same password*.

Password Storage Cheat Sheet

Introduction¶

- It is essential to store passwords in a way that prevents them from being obtained by an attacker even if the application or database is compromised.
- After an attacker has acquired stored password hashes, they are always able to brute force hashes offline.
- As a defender, it is only possible to slow down offline attacks by selecting hash algorithms that are as resource intensive as possible.

Hashing vs Encryption

- Hashing and encryption both provide ways to keep sensitive data safe.
- **Passwords should be hashed, NOT encrypted.**
- **Hashing is a one-way function** (i.e., it is impossible to "decrypt" a hash and obtain the original plaintext value).
- Hashing is appropriate for password validation.
- Even if an attacker obtains the hashed password, they cannot enter it into an application's password field and log in as the victim.
- **Encryption is a two-way function**, meaning that the original plaintext password can be retrieved (if we have the key)

How Attackers Crack (unsalted) Password Hashes

- Although it is not possible to "decrypt" password hashes to obtain the original passwords, it is possible to "crack" the hashes in some circumstances. The basic steps are:
- Select a password you think the victim has chosen (e.g.password1!)
- Calculate the hash
- Compare the hash you calculated to the hash of the victim.
- If they match, you have correctly "cracked" the hash and now know the plaintext value of their password. (stop)

How Attackers Crack Password Hashes

- This process is repeated for a large number of potential candidate passwords.
- Different methods can be used to select candidate passwords, including:
  - Lists of passwords obtained from other compromised sites
  - Brute force (trying every possible candidate)
  - Dictionaries or wordlists of common passwords

How Attackers Crack Password Hashes

- While the number of permutations can be enormous, with high speed hardware (such as GPUs) and cloud services with many servers for rent, the cost to an attacker is relatively small to do successful password cracking especially when best practices for hashing are not followed.
- **Strong passwords stored with modern hashing algorithms and using hashing best practices should be effectively impossible for an attacker to crack.**
- It is your responsibility as an administrator to select a modern hashing algorithm (later)

Password Storage Concepts

- Salting:
- A salt is a unique, randomly generated string that is added to each password as part of the hashing process.
- As the salt is unique for every user, an attacker has to crack hashes one at a time using the respective salt rather than calculating a hash once and comparing it against every stored hash.
- This makes cracking large numbers of hashes significantly harder, as the time required grows in direct proportion to the number of hashes.

Password Storage Concepts

- Salting also protects against an attacker pre-computing hashes using rainbow tables or database-based lookups.
- Finally, salting means that it is impossible to determine whether two users have the same password without cracking the hashes, as the different salts will result in different hashes even if the passwords are the same.
- Modern hashing algorithms such as Argon2id, bcrypt, and PBKDF2 automatically salt the passwords, so no additional steps are required when using them.