Guide to Understanding Hashing

1 What is Hashing?

Hashing is a process that transforms a piece of data (like a password) into a fixed-size string of characters, which looks like a random mix of letters and numbers. This transformation is done using a special function called a hash function.

2 Key Concepts

2.1 Hash Function

A hash function takes input data and produces a unique hash value. It's like a special recipe that turns your data into a unique dish.

2.2 Hash Value (Hash)

The output of the hash function. It's a fixed-length string that represents the original data but looks completely different.

2.3 One-Way Function

Hashing is one-way, meaning you can't reverse the hash to get back to the original data. It's like having a secret recipe: once you bake the cake (hash), you can't go back to the ingredients (original data).

3 Why Use Hashing?

3.1 Security

- **Protects Data**: Hashing keeps your original data hidden. Even if someone sees the hash, they can't easily figure out the original data.
- **Unique Hashes**: If two pieces of data are the same, their hashes will be the same, but different data will have different hashes.

3.2 Verification

- **Check Data**: Hashing is used to check if the data is correct. For example, when you log in, your password is hashed, and the system checks if it matches the stored hash.

4 How Hashing Works

4.1 Create a Hash

- **Input Data**: Start with your data (like a password).
- **Hash Function**: Use a hash function to convert this data into a hash.
- **Result**: The hash is stored instead of the original data.

4.2 Verify Data

- **Input Data Again**: When you want to verify the data, hash it again using the same function.
- **Compare Hashes**: Compare the new hash with the stored hash. If they match, the data is correct.

5 Example: Hashing a Password

- **Password Creation**:
  Original Password: `mySecretPassword`
  Hashing: Use a hash function to create a hash like `abc123xyz456`.
- **Storing the Hash**:
  Database: Save `abc123xyz456` in the database, not the actual password.
- **User Login**:
  Enter Password: User enters `mySecretPassword`.
  Hash the Input: Hash `mySecretPassword` again, resulting in `abc123xyz456`.
  Check: Compare this new hash with the stored hash. If they match, the password is correct.

6 Common Hashing Algorithms

- **MD5**: Older and less secure. Not recommended for passwords.
- **SHA-256**: More secure than MD5, but still basic.
- **BCrypt**: Modern and secure, includes a salt to add extra protection.
- **Argon2**: Highly secure, designed for password hashing.

7 Important Tips

- **Always Hash Passwords**: Never store plain-text passwords. Always use a secure hash function.
- **Use Salting**: Add a random value (salt) to make hashes unique, even for identical passwords.
- **Choose a Strong Algorithm**: Use a modern, secure hashing algorithm like BCrypt or Argon2.

8 Summary

Hashing is a way to securely transform and store data. It protects the original information by making it impossible to reverse the hash and reveals if the data matches by comparing hashes. Always use a secure hashing algorithm and remember to add a salt for extra security.