Hyoin Lee (20102711)

Step 1 – Knowledge Question (40-70 words) In your own words, describe what hashing is in general.

I would say hashing is process that transforms an input (or key) into a usually fixed size string of characters, which is typically a hash code. This transformation uses a hash function. Hashing is commonly used in data retrieval for quick access, data encryption, and ensuring data integrity.

Step 2 – Knowledge Question (60-100 words) Research hashing algorithms. Describe advantages and disadvantages for at least three different hashing algorithms. Please provide references for external resources.

1. MD5 (Message Digest Algorithm 5)

Advantages:

Speed: MD5 is fast and efficient for small data blocks.

Simplicity: Easy to implement and widely supported in various programming languages and libraries.

Disadvantages:

Security Vulnerabilities: MD5 is vulnerable to collision attacks where different inputs produce the same hash output. It is considered cryptographically broken and unsuitable for further use in security contexts.

Weaknesses in Integrity: The collisions make it unreliable for ensuring data integrity in secure applications.

2. SHA-256 (Secure Hash Algorithm 256-bit)

Advantages:

Security: SHA-256 is part of the SHA-2 family, which is considered highly secure and resistant to collisions and preimage attacks.

Widespread Use: Used in many security protocols and applications, such as SSL/TLS, Bitcoin, and blockchain technology.

Disadvantages:

Performance: SHA-256 is slower compared to MD5 and other less secure hash functions. This can be a disadvantage in performance-sensitive applications.

Computational Overhead: Requires more computational resources, making it less efficient for applications needing rapid hash calculations.

3. Blake3

Advantages:

High Performance: Blake2 is faster than MD5 and SHA-2 while providing a high level of security.

Security: Offers cryptographic strength comparable to SHA-3 and is resistant to various forms of attacks, including collision and length-extension attacks.

Flexibility: Provides features like configurable output length and keying for message authentication.

Disadvantages:

Relatively New: While highly regarded, Blake2 is newer and not as extensively vetted as some of the older algorithms like SHA-2, although it has gained significant adoption in recent years.

Less Widespread Support: Not as universally supported in libraries and protocols as older algorithms, although this is changing

Reference

<https://www.geeksforgeeks.org/what-is-the-md5-algorithm/>  
<https://nordvpn.com/blog/sha-256/>

<https://www.blake2.net/>

Step 3 – Knowledge Question (50-90 words) Provide a stepwise description (algorithmic) of a) how you can store a password safely using hashing techniques and b) how you can verify that some string is the right password?

1. 1. Obtain a plaintext password.

2. Generate a salt.

3. Combine password and salt.

4. Apply hash algorithm to the salted password.

5. Store the salt and hashed password in a database.

6. Catch specific exceptions rather than using a bare ‘except’ to avoid masking unexpected errors.

1. 1. Retrieve the hashed password from the database based on the username or id

2. Obtain the password entered by the user for verification.

3. Hash the entered password using the same hash algorithm used for storage.

4. Compare the result of hash user entered with already stored.

5. If the hashes match, the entered password is correct, otherwise not correct.

Step 4 – Knowledge Question (20-40 words) What is the purpose of a “salt” when hashing a password? What are the two most important properties of a “salt”?

Salt is a randomly generated string of characters that is added to a password before hashing. This is to enhance the security of the hashed password.

The two most important features of salt are ‘uniqueness’ and ‘randomness’. With these features, the security of hashed password is significantly strengthened.

Step 5 – How does the argon2-cffi package handle salt?

When we hash a password with ‘argon2-cffi’, it automatically creates a unique salt.

Then, the salt is combined with the password and included in the resulting hash string.

This salt is stored within the hash string in the database, so we don’t need to manage it separately. During password verification, the package extracts the salt from the stored hash, combines it with the provided password, and compares the result to the stored hash.