SHA-512 Hashing Code Documentation

Introduction

SHA-512, or Secure Hash Algorithm 512, is a hashing algorithm used to convert text of any length into a fixed-size string. Each output produces a SHA-512 length of 512 bits (64 bytes).

This algorithm is commonly used for email addresses hashing, password hashing, and digital record verification. SHA-512 is also used in blockchain technology, with the most notable example being the BitShares network.

SHA-512 Algorithm:

1. Convert the input message into binary format (sequence of bits).
2. Append '0' bits to the message until its length is 896 bits less than a multiple of 1024 (i.e., the message length is congruent to 896 mod 1024).
3. Initialize eight 64-bit words (H0 to H7) as the initial hash values.
4. Define a set of 64 64-bit constant words (K0 to K63) derived from the first 64 prime numbers' fractional parts.
5. Divide the padded message into 1024-bit (128-byte) chunks.
6. For each chunk, perform the following steps:
    a. Initialize eight 64-bit working variables (A, B, C, D, E, F, G, H) with the current hash values.
    b. Iterate through 80 rounds, performing a series of operations in each round.
    c. Update the working variables A, B, C, D, E, F, G, and H based on the round-specific operations and values from the message schedule W.
    d. Update the hash values H0 to H7 by adding the corresponding working variables (A to H).
7. Concatenate the final hash values H0 to H7 to obtain the 512-bit (64-byte) hash.
8. The 512-bit hash is the final output.

How It Works:

1. **Constants and Initial Hash Values**:
    - The code defines the SHA-512 constants K, initial hash values H, and other constants for padding. These constants are derived from the square roots of the first 80 prime numbers and are used in the SHA-512 algorithm.
2. **SHA-512 Functions**:
    - Several functions are defined, such as right_rotate, ch, maj, sigma0, sigma1, epsilon0, and epsilon1. These functions are used within the SHA-512 algorithm for various bitwise and logical operations.
3. **Padding the Message**:
    - The pad_message function is used to pad the input message as per SHA-512 requirements. It adds a single '1' bit to the message, followed by '0' bits to make the message length congruent to 896 (mod 1024), and finally appends the original message length as a 128-bit big-endian integer.
4. **Main Code for Hashing**:
    - The code takes user input as the message to be hashed.
    - The message is encoded to bytes and padded using the pad_message function.
    - The code then processes the padded message in 1024-bit (128-byte) chunks.
    - For each chunk, it initializes the message schedule W and extends it according to the SHA-512 algorithm.
    - It also initializes working variables a, b, c, d, e, f, g, and h with the initial hash values.
    - The main loop iterates through the message schedule and updates the working variables according to the SHA-512 algorithm.
    - The hash values are updated at the end of each chunk.
5. **Final Hash Value**:
    - After processing all the chunks, the final hash value is stored in the H array.
6. **Conversion to Hexadecimal Digest**:
    - The final hash values in H are converted to a hexadecimal digest, which is the SHA-512 hash of the input message.

The SHA-512 algorithm is designed with the aim of being highly secure and resistant to collision attacks, where two different inputs produce the same hash value, as well as preimage attacks, where an attacker tries to find an input message that produces a specific hash value.