SHA-512 Hashing Code Documentation

Introduction

This Python code is designed to calculate the SHA-512 hash of a user-provided string. SHA-512 is a cryptographic hash function that produces a fixed-size 512-bit (64-byte) hash value. The code utilizes the SHA-512 algorithm to compute this hash.

SHA-512 Algorithm Overview

Input: A message (arbitrary-length binary data) to be hashed.

Output: A fixed-length 512-bit (64-byte) hash value.

Constants and Initial Hash Values:

- The SHA-512 algorithm uses a set of predefined constants (K) and initial hash values (H).

Message Preprocessing:

1. **Padding**: The input message is padded to a length that is congruent to 896 (mod 1024) bits, following the rules defined by the SHA-512 standard. Padding ensures that the message can be processed in 1024-bit (128-byte) blocks.
2. **Length Representation**: The length of the original message (in bits) is appended to the end of the padded message as a 128-bit big-endian representation.

Hash Computation:

- The padded message is divided into 1024-bit blocks, and the following steps are performed for each block:

1. **Message Schedule (W):** The 1024-bit block is divided into 80 64-bit words, forming the message schedule (W). The first 16 words come directly from the block, and the remaining 64 words are calculated based on the previous words using a combination of bitwise rotations and XOR operations.

2. **Working Variables (a, b, c, d, e, f, g, h):** Eight 64-bit working variables are initialized with the current hash values (H).
3. **Main Loop (80 Rounds):** For each of the 80 rounds, the working variables are updated as follows:
    - Calculate temporary variables T1 and T2 based on bitwise operations and logical functions (ch, maj, sigma0, sigma1, epsilon0, epsilon1).
    - Update working variables:
        - h = g
        - g = f
        - f = e
        - e = d + T1
        - d = c
        - c = b
        - b = a
        - a = T1 + T2
4. **Update Hash Values:** After the 80 rounds, the current hash values (H) are updated as follows:
    - H[0] += a
    - H[1] += b
    - H[2] += c
    - H[3] += d
    - H[4] += e
    - H[5] += f
    - H[6] += g
    - H[7] += h

How It Works

1. **Initialization:** The code begins by initializing the SHA-512 constants, initial hash values (H), and various supporting functions. These constants and functions are essential for the SHA-512 algorithm.
2. **Padding:** The input string provided by the user is first encoded as bytes and then padded according to the SHA-512 rules. The padding ensures that the message length is congruent to 896 (mod 1024), with the addition of a single '1' bit followed by '0' bits and a 128-bit big-endian representation of the message length.
3. **Message Block Processing:** The padded message is then processed in blocks of 1024 bits (128 bytes). For each block:
    - A message schedule (W) is initialized, consisting of 80 64-bit words.
    - The first 16 words of the message schedule are derived from the 128-byte block.
    - The remaining words (from 16 to 79) are computed by applying various bitwise operations on previous words.

4. **Hash Computation:** The code initializes eight working variables (a, b, c, d, e, f, g, and h) with the initial hash values (H). It then enters the main loop where it processes each word in the message schedule using a series of calculations.
    - The working variables are updated in each iteration of the loop based on various bitwise and logical operations, involving the message schedule (W), constants (K), and the working variables themselves.
    - The result is a series of eight 64-bit values representing the intermediate hash values.
5. **Final Hash:** The final step is to combine the eight intermediate hash values into a single 512-bit hash value. These values are converted to a hexadecimal representation and presented as the SHA-512 hash.

Final Result:

- After processing all blocks, the final hash value is formed by concatenating the 64-bit hash values H0, H1, H2, H3, H4, H5, H6, and H7, resulting in a 512-bit (64-byte) hash.

Output:

- The output is the SHA-512 hash of the input message, typically represented as a 128-character hexadecimal string.

The SHA-512 algorithm is designed to produce a secure and unique hash value for any given input message, making it suitable for various security applications, including data integrity verification and password storage.