**SHA-512**

The hashing algorithm SHA-512 applies a hashing operation to some data that is supplied to it.

Many things, like internet security, digital certificates, and even blockchains, require hashing algorithms. This is a brief overview with some basic and simple arithmetic along with some graphics for a hashing technique called SHA-512 because hashing algorithms are so important to digital security and encryption. It belongs to a class of hashing algorithms known as SHA-2, which also includes SHA-256 and is used to hash the bitcoin blockchain.

So, SHA-512 does its work in a few stages. These stages go as follows:

* Input formatting
* Hash buffer initialization
* Message Processing
* Output

SHA-512 is used to produce a fixed-size (512-bit) hash value from an input message. The code consists of various functions and constants to perform the necessary operations for SHA-512. Here's an explanation of each function and constant:

**K (Additive Constants):**

K is an array of 64 constants, used in the compression function. These constants are derived from the fractional parts of the cube roots of the first 80 prime numbers. They play a critical role in the compression loop.

**HASH\_VALUE (Initial Hash Values):**

HASH\_VALUE is an array of eight 64-bit initial hash values. These values are used as the initial state of the hash function before processing the message. They are also updated during the compression function.

**Ch(e, f, g):**

The Ch function is used in the SHA-512 compression loop. It takes three 64-bit input values (e, f, and g) and computes the Ch operation, which is a bitwise operation defined as (e & f) ^ (~e & g).

**Maj(a, b, c):**

The Maj function is another bitwise operation used in the compression loop. It takes three 64-bit input values (a, b, and c) and computes the Maj operation, defined as (a & b) ^ (a & c) ^ (b & c).

**rotr(x, n):**

The rotr function performs a right circular shift operation on a 64-bit value x by n bits.

**summation\_a(a):**

The summation\_a function is used in the SHA-512 compression loop. It applies a combination of right circular shifts and XOR operations on a 64-bit input value a.

**summation\_e(e):**

Similar to summation\_a, the summation\_e function performs a combination of right circular shifts and XOR operations on a 64-bit input value e.

**sigma\_0(word):**

The sigma\_0 function is used to calculate a value based on a 64-bit word. It applies two different right circular shifts and a logical XOR operation.

**sigma\_1(word):**

The sigma\_1 function is similar to sigma\_0 but uses different shift and XOR operations.

**addition\_modulo\_2\_64(value):**

This function performs modulo 2^64 addition on a 64-bit value, ensuring that the result is kept within the 64-bit range.

**pad\_message(message):**

This function pads the input message to ensure that its length is a multiple of 128 bytes, as required by the SHA-512 algorithm. It adds the padding bits, including the message length in bits, to the end of the message.

**divide\_to\_blocks(message):**

This function divides the padded message into 128-byte blocks for processing.

**compression\_function(message):**

The compression\_function is the core of the SHA-512 algorithm. It processes a 128-byte block of the message using a series of operations, including bitwise operations, circular shifts, and addition modulo 2^64. It updates the hash values in the HASH\_VALUE array.