

SHA-512 Hashing Algorithm

Secure Hash Algorithms (SHA) are cryptographic functions that transform data into a fixed-length hash value, which can be seen as the digital “fingerprint” of the data. The SHA-512 algorithm is part of the SHA-2 (Secure Hash Algorithm 2) family, which was designed by the National Security Agency (NSA) and published in 2001 by the National Institute of Standards and Technology (NIST) as a U.S. Federal Information Processing Standard (FIPS PUB 180-4).

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 Bit Shares network.

Basics of SHA-512

SHA-512 produces a hash value of 512 bits, or 64 bytes, making it one of the longer hash functions in the SHA-2 family. Like all cryptographic hash functions, SHA-512 has a few essential properties:

Deterministic: The same input will always produce the same output.

Fast to compute: For any given data, it’s relatively quick to calculate the hash.

Irreversible: You can’t deduce the original input from its hash.

Collision-resistant: It’s computationally difficult to find two different inputs that produce the same hash.

Avalanche effect: A tiny change in input (even flipping a single bit) will produce a drastically different hash.

Real-world Applications

Digital signatures: To confirm the integrity of a message or document.

Certificate generation: Used by Certificate Authorities (CAs) to ensure the security of digital certificates.

Password hashing: Storing passwords in databases as hashes rather than plain text.

Blockchain and cryptocurrencies: For ensuring data integrity and security.

How it works?

Initialization: It begins with eight initial hash values derived from the square roots of the first eight prime numbers.

Pre-processing: The input message is padded to ensure its length is a multiple of the block size. A 128-bit length of the original message (before padding) is added at the end of the padded message.

Parsing: The message is then divided into 1024-bit blocks.

Main loop: Each 1024-bit block is processed in a series of 80 rounds that manipulate the data using logical operations, bitwise shifts, and modular arithmetic.

Output: After all the blocks are processed, the resulting 512-bit message digest is outputted as the hash.

Documentation for SHA-512 Hash Algorithm

This Python script implements the SHA-512 hash algorithm, which is commonly used for securing data integrity and authentication. It takes an input message as a string and returns its SHA-512 hash in hexadecimal format.

Usage:

1. Import the necessary libraries - ``binascii`` & ``struct``
2. Initialize the initial hash values and constants used in the SHA-512 algorithm.
3. Define a function to right-rotate bits of an integer.
4. Define the ``sha512`` function to calculate the SHA-512 hash of a given message.
5. Verify that the input message is a string and raise a `TypeError` if not.
6. Prepare the message by encoding it to bytes and adding necessary padding.
7. Break the message into 128-byte chunks and process each chunk.

8. Calculate intermediate hash values based on the message blocks and constants.

9. Finally, the SHA-512 hash is computed and returned as a hexadecimal string.

Functions:

right_rotate(n: int, bits: int) -> int:

Description: This function rotates an integer right by a specified number of bits. It simulates a bitwise right rotation operation.

Parameters:

n (int): The integer to be rotated.

bits (int): The number of bits to rotate the integer.

Returns: The result of the right rotation.

sha512(message: str) -> str:

Description: This function calculates the SHA-512 hash of the input message using the SHA-512 algorithm. It pads the message, breaks it into chunks, and applies the algorithm to each chunk to compute the final hash.

Parameters:

message (str): The input message for which the SHA-512 hash is to be calculated.

Returns: The SHA-512 hash of the input message as a hexadecimal string.

initial_hash:

Description: A tuple containing the initial hash values (H0 - H7) used in the SHA-512 algorithm. These values are specific to the SHA-512 algorithm.

const_hash:

Description: A tuple containing the additive constants used in the SHA-512 algorithm. These constants are used during the processing of message chunks

```
if __name__ == "__main__":
```

Description: The script's entry point for user interaction. It reads user input, calls the sha512 function to calculate the hash, and displays the result.

The code ensures proper message padding and processing of message chunks according to the SHA-512 specification.

Input:

user_input = input("Hey, Enter your message here: ") : This line prompts the user to enter a message and stores the input in the user_input variable.

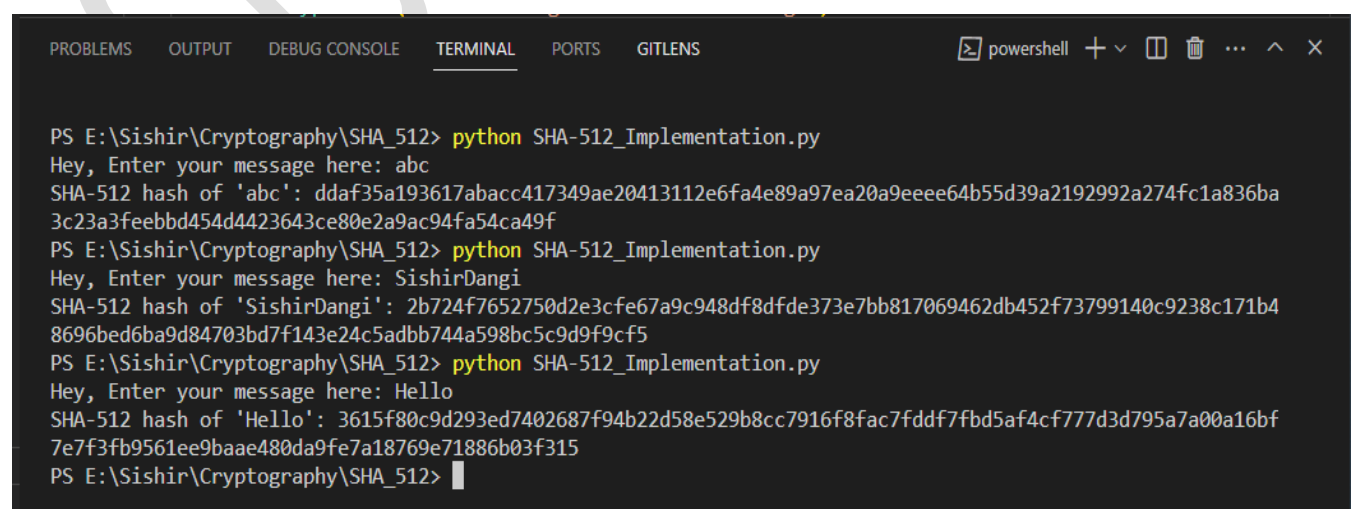
result = sha512(user_input) : This line attempts to compute the SHA-512 hash of the user's input.

Output:

Returns the SHA-512 hash of the input message as a hexadecimal string.

print(f"SHA-512 hash of '{user_input}': {result}") : This line prints the SHA-512 hash of the user's input, assuming that the sha512 function is correctly imported and the result variable contains the hash.

Example:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS powershell + - [ ] [X] ... ^ X
PS E:\Sishir\Cryptography\SHA_512> python SHA-512_Implementation.py
Hey, Enter your message here: abc
SHA-512 hash of 'abc': ddaf35a193617abacc417349ae20413112e6fa4e89a97ea20a9e64b55d39a2192992a274fc1a836ba
3c23a3feebbd454d4423643ce80e2a9ac94fa54ca49f
PS E:\Sishir\Cryptography\SHA_512> python SHA-512_Implementation.py
Hey, Enter your message here: SishirDangi
SHA-512 hash of 'SishirDangi': 2b724f7652750d2e3cfe67a9c948df8dfde373e7bb817069462db452f73799140c9238c171b4
8696bed6ba9d84703bd7f143e24c5adbb744a598bc5c9d9f9cf5
PS E:\Sishir\Cryptography\SHA_512> python SHA-512_Implementation.py
Hey, Enter your message here: Hello
SHA-512 hash of 'Hello': 3615f80c9d293ed7402687f94b22d58e529b8cc7916f8fac7fddf7fbd5af4cf777d3d795a7a00a16bf
7e7f3fb9561ee9baae480da9fe7a18769e71886b03f315
PS E:\Sishir\Cryptography\SHA_512> |
```