Reversing Eight Rounds of SHA-256

mcdair@protonmail.com

Abstract—This paper demonstrates a method to deterministically reverse the first eight rounds (iterations) of the SHA-256 compression function. By tracking the propagation of a single message schedule word through the internal state and expressing its effect algebraically across rounds, we isolate and recover its original value from the final state. The results indicate that for eight rounds of SHA-256, the function remains fully invertible, with each round preserving enough structural information to reconstruct the original input. When reduced to eight rounds, SHA-256 no longer satisfies the criteria of a one-way function, but instead exhibits behavior more akin to a reversible transformation. The reversibility framework introduced here may also extend to SHA-512.

INDEX TERMS—Compression function, cryptanalysis, hash function, reduced-round, reversibility, SHA-256.

I. Introduction

SHA-256, a member of the SHA-2 family of hash functions, is widely deployed in modern cryptographic systems. It relies on a Merkle–Damgård construction with a compression function that performs 64 rounds of mixing operations on a fixed-length input block. The perceived strength of SHA-256 as a cryptographic hash function lies in its resistance to preimage, second preimage, and collision attacks—properties derived from its apparent irreversibility and avalanche effect.

This paper investigates the reversibility of SHA-256 when the compression function is reduced to only eight rounds. It is shown that despite complex and nonlinear transformations within these rounds, the original variable data—the message schedule—persist and remain recoverable. Through algebraic tracing of input propagation across the compression function's internal variables, we explicitly reconstruct each 32-bit message schedule word from the final state.

The implications of this result suggest that SHA-256, truncated to eight rounds, does not qualify as a cryptographic hash function in the strictest sense. Instead, it exhibits behavior similar to an encryption mechanism, where outputs can be reversed given sufficient information. Although this finding does not directly compromise full SHA-256, it offers insights into the structural resilience and diffusion properties of its early rounds.

II. Preliminaries

A. Key Terms

Word	A 32-bit unsigned integer.
Round	A single iteration of the compression function.
$H^{(i)}$	The (intermediate) hash value after processing the *i*-th message block. $H^{(0)}$ denotes the initial hash value and $H^{(N)}$ represents the final hash value serving as the message digest.
$H_j^{(i)}$	A specific 32-bit word *j* (zero-indexed) from a total of eight words that make up 256-bit hash value $\mathrm{H}^{(i)}$.

$a, b, \ldots, h$	32-bit working variables used in the computation of the hash values, $H^{(i)}$.
$a_{t+1}, b_{t+1}, ..., h_{t+1}$	The working variables (values) after processing round t (zero-indexed), collectively referred to as the state. $a_0, b_0, ..., h_0$ represent the initial state of the compression function.
K^t	Round-specific 32-bit constant value (zero-indexed) used in the compression process.
M	The original input message.
$M^{(i)}$	The i-th fixed-length message block obtained from M.
$M_j^{(i)}$	A specific 32-bit word j (zero-indexed) from a total of sixteen words that make up 512-bit message block $M^{(i)}$.
W_t	A specific 32-bit word t (zero-indexed) of the message schedule.
$Ch, \Sigma_1, Maj, \Sigma_0$	Nonlinear bitwise functions used in the SHA-256 compression function.

B. Symbols

Addition (+) and subtraction (−) are defined modulo 2^{32} over 32-bit words.

C. The Truncated Message Schedule

For our analysis, we consider a single 512-bit (padded) message block $M^{(1)}$, allowing an original message M length up to 447 bits. In the standard SHA-256 specification, this block would be expanded into a 64-word message schedule. However, to facilitate the reversibility analysis of the compression function truncated to eight rounds, we introduce a simplified, or *truncated*, message schedule.

Specifically, for the purpose of this study, we define the message schedule W to consist solely of the first eight 32-bit words of $M^{(1)}$, directly mapped as follows:

$$W_t = M_t^{(1)}, \text{ for } t = 0 \text{ to } 7$$

As a result, only the initial 256 bits of the padded input message block (and at most 256 bits of original message M) are utilized and constitute the entire message schedule for the eight rounds of the compression function under investigation. This simplification allows us to directly trace and recover these initial input bits from the truncated function's output.

D. Round Representation

We represent the state update process of the SHA-256 compression function at round t (ranging from 0 to 7 in this implementation) slightly differently from the original specification as we omit the explicit use of temporary variables (T_1, T_2) by substituting their definitions directly into the update equations.

The state variables entering round t are a_t, b_t, …, h_t. These are updated to produce the state variables for the next round, a_{t+1}, b_{t+1}, …, h_{t+1}.

$$a_{t+1} = (h_t + \Sigma_1(e_t) + Ch(e_t, f_t, g_t) + K_t + W_t) \; + (\Sigma_0(a_t) + Maj(a_t, b_t, c_t))$$

$$e_{t+1} = d_t + (h_t + \Sigma_1(e_t) + Ch(e_t, f_t, g_t) + K_t + W_t)$$

$$h_{t+1} = g_t$$

$$g_{t+1} = f_t$$

$$f_{t+1} = e_t$$

$$d_{t+1} = c_t$$

$$c_{t+1} = b_t$$

$$b_{t+1} = a_t$$

This unrolled form emphasizes the algebraic structure and dependencies between rounds, which is essential for the reversibility analysis presented in later sections. Figure 1 specifically illustrates the direct inheritance of state variables that forms a crucial part of these dependencies.

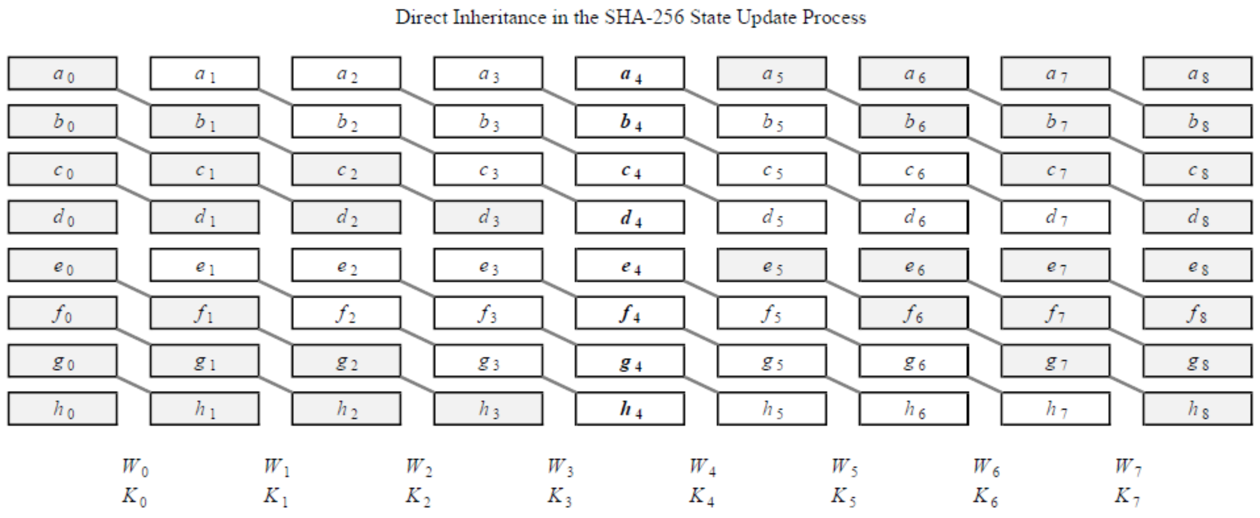
Direct Inheritance in the SHA-256 State Update Process
a_0 b_0 c_0 d_0 e_0 f_0 g_0 h_0
a_1 … h_8
W_0 K_0
W_1 K_1
W_2 K_2
W_3 K_3
W_4 K_4
W_5 K_5
W_6 K_6
W_7 K_7

Fig. 1. Variable Inheritance over Eight SHA-256 Compression Rounds.

E. Initial and Final State Values

The initial state of the compression function, consisting of eight working variables, is derived from initial hash value $H^{(0)}$, which comprises a predefined set of eight 32-bit constants.

$$\boldsymbol{a_0} = H_0^{(0)}, \boldsymbol{b_0} = H_1^{(0)}, ..., \boldsymbol{h_0} = H_7^{(0)}$$

Since this analysis is restricted to a single message block, the final state of the compression function can be directly deduced from hash value $H^{(1)}$, which serves as the resulting 256-bit message digest.

$$H_0^{(1)} = a_8 + H_0^{(0)}, H_1^{(1)} = b_8 + H_1^{(0)}, ..., \ H_7^{(1)} = h_8 + H_7^{(0)}$$

Therefore:

$$\boldsymbol{a_8} = H_0^{(1)} - H_0^{(0)}, \boldsymbol{b_8} = H_1^{(1)} - H_1^{(0)}, ..., \mathbf{h_8} = H_7^{(1)} - H_7^{(0)}$$

We find that the SHA-256 compression function initial as well as final state values are known a priori, for a single-block message.

III. Reversing SHA-256 Reduced to Eight Rounds

A. Calculating the First Word of the Message Schedule

We examine how the compression function incorporates the first word of the message schedule, W_0, and trace its propagation through the internal state across eight rounds.

1) First Round

W_0 is used in the computation of next state variable a (a_1) as follows:

$$a_1 = h_0 + \boldsymbol{W_0} + K_0 + Ch(e_0, f_0, g_0) + \Sigma_1(e_0) + Maj(a_0, b_0, c_0) + \Sigma_0(a_0)$$

W_0 also contributes to e_1:

$$e_1 = d_0 + h_0 + \boldsymbol{W_0} + K_0 + Ch(e_0, f_0, g_0) + \Sigma_1(e_0)$$

Since W_0 directly influences both a_1 and e_1, we proceed to examine the evolution of these values.

2) Second Round

Next state variable b (b_2) directly inherits the value of a_1, and f_2 takes over the value of e_1:

$$b_2 = a_1$$

$$f_2 = e_1$$

3) Third Round

c_3 and g_3 inherit b_2 and f_2 respectively:

$$c_3 = b_2$$

$$g_3 = f_2$$

4) Fourth Round

d_4 and h_4 inherit c_3 and g_3 respectively:

$$d_4 = c_3$$

$$h_4 = g_3$$

5) Fifth Round

a_5 depends on h_4, and e_5 depends on h_4 and d_4 (both containing value of interest W_0):

$$a_5 = \boldsymbol{h_4} + W_4 + K_4 + Ch(e_4, f_4, g_4) + \Sigma_1(e_4) + Maj(a_4, b_4, c_4) + \Sigma_0(a_4)$$

$$e_5 = \boldsymbol{d_4} + \boldsymbol{h_4} + W_4 + K_4 + Ch(e_4, f_4, g_4) + \Sigma_1(e_4)$$

6) Sixth Round

b_6 inherits a_5, and f_6 inherits e_5:

$$b_6 = a_5$$

$$f_6 = e_5$$

7) Seventh Round

c_7 inherits b_6, and g_7 inherits f_6:

$$c_7 = b_6$$

$$g_7 = f_6$$

8) Eighth Round

d_8 inherits c_7, and h_8 inherits g_7:

$$d_8 = c_7 = b_6 = a_5 = \boldsymbol{h_4} + W_4 + K_4 + Ch(e_4, f_4, g_4) + \Sigma_1(e_4) + Maj(a_4, b_4, c_4) + \Sigma_0(a_4)$$

$$h_8 = g_7 = f_6 = e_5 = \boldsymbol{d_4} + \boldsymbol{h_4} + W_4 + K_4 + Ch(e_4, f_4, g_4) + \Sigma_1(e_4)$$

After eight rounds we conclude that W_0 (contained by both h_4 and d_4) is part of final state variables d_8 and h_8. We also notice that the fifth word of the message schedule (W_4) has been mixed into these resulting values. Consequently, it may initially appear infeasible to deterministically reconstruct W_0. However, we can eliminate W_4 whilst preserving W_0 the following way.

$$d_8 = \boldsymbol{h_4 + W_4 + K_4 + Ch(e_4, f_4, g_4) + \Sigma_1(e_4)} + Maj(a_4, b_4, c_4) + \Sigma_0(a_4)$$

$$h_8 = d_4 + \boldsymbol{h_4 + W_4 + K_4 + Ch(e_4, f_4, g_4) + \Sigma_1(e_4)}$$

Subtract d_8 from h_8:

$$h_8 - d_8 = d_4 - Maj(a_4, b_4, c_4) - \Sigma_0(a_4)$$

From earlier, we recall that:

$$d_4 = c_3 = b_2 = a_1 = h_0 + \boldsymbol{W_0} + K_0 + Ch(e_0, f_0, g_0) + \Sigma_1(e_0) + Maj(a_0, b_0, c_0) + \Sigma_0(a_0)$$

Substituting:

$$h_8 - d_8 = h_0 + \boldsymbol{W_0} + K_0 + \mathrm{Ch}(e_0, f_0, g_0) + \Sigma_1(e_0) + Maj(a_0, b_0, c_0) + \Sigma_0(a_0) - Maj(a_4, b_4, c_4) - \Sigma_0(a_4)$$

Rearranging to solve for W_0:

$$\boldsymbol{W_0} = h_8 - d_8 - h_0 - K_0 - Ch(e_0, f_0, g_0) - \Sigma_1(e_0) - Maj(a_0, b_0, c_0) - \Sigma_0(a_0) + Maj(\boldsymbol{a_4}, \boldsymbol{b_4}, \boldsymbol{c_4}) + \Sigma_0(\boldsymbol{a_4})$$

This expression isolates W_0, with remaining unknowns being a_4, b_4 and c_4, which can be further constrained in subsequent analysis.

9) Determining Fifth State Variable a

Our analysis will once again focus on the evolution of variable a_4 as it traverses the compression function:

We observe that the value of a_4 propagates to b_5, c_6 and d_7 respectively.

$$d_7 = c_6 = b_5 = a_4$$

d_7 is being used to compose state variable e_8 in (final) round eight as follows:

$$e_8 = d_7 + h_7 + W_7 + K_7 + Ch(e_7, f_7, g_7) + \Sigma_1(e_7)$$

Using a similar technique as before, we can eliminate W_7 and isolate d_7:

$$e_8 = d_7 + \boldsymbol{h_7 + W_7 + K_7 + Ch(e_7, f_7, g_7) + \Sigma_1(e_7)}$$

$$a_8 = \boldsymbol{h_7 + W_7 + K_7 + Ch(e_7, f_7, g_7) + \Sigma_1(e_7)} + Maj(a_7, b_7, c_7) + \Sigma_0(a_7)$$

$$e_8 - a_8 = d_7 - Maj(a_7, b_7, c_7) - \Sigma_0(a_7)$$

$$d_7 = e_8 - a_8 + Maj(a_7, b_7, c_7) + \Sigma_0(a_7)$$

Substituting $\boldsymbol{a_7}$ for b_8, $\boldsymbol{b_7}$ for c_8 and $\boldsymbol{c_7}$ for d_8 we get:

$$\boldsymbol{a_4} = d_7 = e_8 - a_8 + Maj(b_8, c_8, d_8) + \Sigma_0(b_8)$$

10) Determining Fifth State Variable b

Similarly, we can calculate b_4. Note that the calculation of a_4 (equalling d_7), as just presented, is a prerequisite.

$$d_6 = c_5 = b_4$$

$$e_7 = d_6 + \boldsymbol{h_6 + W_6 + K_6 + Ch(e_6, f_6, g_6) + \Sigma_1(e_6)}$$

$$a_7 = \boldsymbol{h_6 + W_6 + K_6 + Ch(e_6, f_6, g_6) + \Sigma_1(e_6)} + Maj(a_6, b_6, c_6) + \Sigma_0(a_6)$$

$$e_7 - a_7 = d_6 - Maj(a_6, b_6, c_6) - \Sigma_0(a_6)$$

$$d_6 = e_7 - a_7 + Maj(a_6, b_6, c_6) + \Sigma_0(a_6)$$

Substituting $\boldsymbol{e_7}$ for f_8, $\boldsymbol{a_7}$ for b_8, $\boldsymbol{a_6}$ for c_8, $\boldsymbol{b_6}$ for d_8 and $\boldsymbol{c_6}$ for a_4 we get:

$$\boldsymbol{b_4} = d_6 = f_8 - b_8 + Maj(c_8, d_8, \boldsymbol{a_4}) + \Sigma_0(c_8)$$

11) Determining Fifth State Variable c

Finally, c_4 can be determined in an analogous way. Note that the calculation of a_4 and b_4 (equalling d_7 and d_6 respectively), as just presented, is a prerequisite.

$$d_5 = c_4$$

$$e_6 = d_5 + \boldsymbol{h_5 + W_5 + K_5 + Ch(e_5, f_5, g_5) + \Sigma_1(e_5)}$$

$$a_6 = \boldsymbol{h_5 + W_5 + K_5 + Ch(e_5, f_5, g_5) + \Sigma_1(e_5)} + Maj(a_5, b_5, c_5) + \Sigma_0(a_5)$$

$$e_6 - a_6 = d_5 - Maj(a_5, b_5, c_5) - \Sigma_0(a_5)$$

$$d_5 = e_6 - a_6 + Maj(a_5, b_5, c_5) + \Sigma_0(a_5)$$

Substituting $\boldsymbol{e_6}$ for g_8, $\boldsymbol{a_6}$ for c_8, $\boldsymbol{a_5}$ for d_8, $\boldsymbol{b_5}$ for a_4 and $\boldsymbol{c_5}$ for b_4 we get:

$$\boldsymbol{c_4} = d_5 = g_8 - c_8 + Maj(d_8, \boldsymbol{a_4}, \boldsymbol{b_4}) + \Sigma_0(d_8)$$

12) Summary

We have solved all unknowns and are now able to calculate W_0 deterministically.

$$\boldsymbol{a_4} = e_8 - a_8 + Maj(b_8, c_8, d_8) + \Sigma_0(b_8)$$

$$\boldsymbol{b_4} = f_8 - b_8 + Maj(c_8, d_8, \boldsymbol{a_4}) + \Sigma_0(c_8)$$

$$\boldsymbol{c_4} = g_8 - c_8 + Maj(d_8, \boldsymbol{a_4}, \boldsymbol{b_4}) + \Sigma_0(d_8)$$

$$\boldsymbol{W_0} = h_8 - d_8 - h_0 - K_0 - Ch(e_0, f_0, g_0) - \Sigma_1(e_0) - Maj(a_0, b_0, c_0) - \Sigma_0(a_0) + Maj(\boldsymbol{a_4}, \boldsymbol{b_4}, \boldsymbol{c_4}) + \Sigma_0(\boldsymbol{a_4})$$

B. Calculating the Subsequent Word(s) of the Message Schedule

Following the same process of determining W_0, we find following results with regard to W_1.

Note that W_1 is mixed into the state as of the second round, so we start there.

1) Second Round

$$a_2 = h_1 + \boldsymbol{W_1} + K_1 + Ch(e_1, f_1, g_1) + \Sigma_1(e_1) + Maj(a_1, b_1, c_1) + \Sigma_0(a_1)$$

$$e_2 = d_1 + h_1 + \boldsymbol{W_1} + K_1 + Ch(e_1, f_1, g_1) + \Sigma_1(e_1)$$

2) Third Round

$$b_3 = a_2$$

$$f_3 = e_2$$

3) Fourth Round

$$c_4 = b_3$$

$$g_4 = f_3$$

4) Fifth Round

$$d_5 = c_4$$

$$h_5 = g_4$$

5) Sixth Round

$$a_6 = \boldsymbol{h_5} + W_5 + K_5 + Ch(e_5, f_5, g_5) + \Sigma_1(e_5) + Maj(a_5, b_5, c_5) + \Sigma_0(a_5)$$

$$e_6 = \boldsymbol{d_5} + \boldsymbol{h_5} + W_5 + K_5 + Ch(e_5, f_5, g_5) + \Sigma_1(e_5)$$

6) Seventh Round

$$b_7 = a_6$$

$$f_7 = e_6$$

7) Eighth Round

$$c_8 = b_7 = a_6 = \boldsymbol{h_5 + W_5 + K_5 + Ch(e_5, f_5, g_5) + \Sigma_1(e_5)} + Maj(a_5, b_5, c_5) + \Sigma_0(a_5)$$

$$g_8 = f_7 = e_6 = d_5 + \boldsymbol{h_5 + W_5 + K_5 + Ch(e_5, f_5, g_5) + \Sigma_1(e_5)}$$

Eliminate W_5:

$$g_8 - c_8 = \boldsymbol{d_5} - Maj(a_5, b_5, c_5) - \Sigma_0(a_5)$$

Substitute $\boldsymbol{d_5} = c_4 = b_3 = a_2 = h_1 + \boldsymbol{W_1} + K_1 + Ch(e_1, f_1, g_1) + \Sigma_1(e_1) + Maj(a_1, b_1, c_1) + \Sigma_0(a_1)$:

$$g_8 - c_8 = h_1 + \boldsymbol{W_1} + K_1 + Ch(e_1, f_1, g_1) + \Sigma_1(e_1) + Maj(a_1, b_1, c_1) + \Sigma_0(a_1) - Maj(a_5, b_5, c_5) - \Sigma_0(a_5)$$

Rearranging to solve for W_1:

$$\boldsymbol{W_1} = g_8 - c_8 - h_1 - K_1 - Ch(e_1, f_1, g_1) - \Sigma_1(e_1) - Maj(a_1, b_1, c_1) - \Sigma_0(a_1) + Maj(a_5, b_5, c_5) + \Sigma_0(a_5)$$

Substitute $\boldsymbol{h_1}$ for g_0, $\boldsymbol{f_1}$ for e_0, $\boldsymbol{g_1}$ for f_0, $\boldsymbol{b_1}$ for a_0, $\boldsymbol{c_1}$ for b_0, $\boldsymbol{a_5}$ for d_8, $\boldsymbol{b_5}$ for a_4 and $\boldsymbol{c_5}$ for b_4 we get:

$$W_1 = \boldsymbol{g_8 - c_8} - g_0 - K_1 - Ch(e_1, e_0, f_0) - \Sigma_1(e_1) - Maj(a_1, a_0, b_0) - \Sigma_0(a_1) \boldsymbol{+ Maj(d_8, a_4, b_4) + \Sigma_0(d_8)}$$

Substitute $g_8 - c_8 + Maj(d_8, a_4, b_4) + \Sigma_0(d_8)$ for c_4 as previously computed:

$$\boldsymbol{W_1} = \boldsymbol{c_4} - g_0 - K_1 - Ch(\boldsymbol{e_1}, e_0, f_0) - \Sigma_1(\boldsymbol{e_1}) - Maj(\boldsymbol{a_1}, a_0, b_0) - \Sigma_0(\boldsymbol{a_1})$$

At this stage, unknowns $\boldsymbol{a_1}$ and $\boldsymbol{e_1}$ remain.

8) Determining Second State Variable a

Since we have previously calculated the value of W_0, we have all the information needed to determine the value of a_1 using the formula originating from the compression function itself.

$$\boldsymbol{a_1} = h_0 + \boldsymbol{W_0} + K_0 + Ch(e_0, f_0, g_0) + \Sigma_1(e_0) + Maj(a_0, b_0, c_0) + \Sigma_0(\mathrm{a}_0)$$

Alternatively, a_1 can be determined in the same manner as a_4, b_4 and c_4 (corresponding to d_7, d_6 and d_5 respectively).

$$d_4 = c_3 = b_2 = a_1$$

$$e_5 = d_4 + \boldsymbol{h_4 + W_4 + K_4 + Ch(e_4, f_4, g_4) + \Sigma_1(e_4)}$$

$$a_5 = \boldsymbol{h_4 + W_4 + K_4 + Ch(e_4, f_4, g_4) + \Sigma_1(e_4)} + Maj(a_4, b_4, c_4) + \Sigma_0(a_4)$$

$$e_5 - a_5 = d_4 - Maj(a_4, b_4, c_4) - \Sigma_0(a_4)$$

$$d_4 = e_5 - a_5 + Maj(a_4, b_4, c_4) + \Sigma_0(a_4)$$

Substitute $\boldsymbol{e_5}$ for h_8 and $\boldsymbol{a_5}$ for d_8:

$$\boldsymbol{a_1} = d_4 = h_8 - d_8 + Maj(\boldsymbol{a_4}, \boldsymbol{b_4}, \boldsymbol{c_4}) + \Sigma_0(\boldsymbol{a_4})$$

9) Determining Second State Variable e

Since we have already calculated the value of W_0, we have all the information needed to determine the value of e_1 using the formula originating from the compression function itself.

$$\boldsymbol{e_1} = d_0 + h_0 + \boldsymbol{W_0} + K_0 + Ch(e_0, f_0, g_0) + \Sigma_1(e_0)$$

Alternatively, e_1 can be computed as follows.

$$a_1 = \boldsymbol{h_0 + W_0 + K_0 + Ch(e_0, f_0, g_0) + \Sigma_1(e_0)} + Maj(a_0, b_0, c_0) + \Sigma_0(a_0)$$

$$e_1 = d_0 + \boldsymbol{h_0 + W_0 + K_0 + Ch(e_0, f_0, g_0) + \Sigma_1(e_0)}$$

$$e_1 - a_1 = d_0 - Maj(a_0, b_0, c_0) - \Sigma_0(a_0)$$

$$\boldsymbol{e_1} = d_0 + \boldsymbol{a_1} - Maj(a_0, b_0, c_0) - \Sigma_0(a_0)$$

Now we have all the information needed to calculate W_1 deterministically.

C. Summary

After extending the same logic to the remaining words, we find next normalized formulas.

$$\boldsymbol{a_4} = e_8 - a_8 + Maj(b_8, c_8, d_8) + \Sigma_0(b_8) \tag{1}$$

$$\boldsymbol{b_4} = f_8 - b_8 + Maj(c_8, d_8, \boldsymbol{a_4}) + \Sigma_0(c_8) \tag{2}$$

$$\boldsymbol{c_4} = g_8 - c_8 + Maj(d_8, \boldsymbol{a_4}, \boldsymbol{b_4}) + \Sigma_0(d_8) \tag{3}$$

$$\boldsymbol{d_4} = h_8 - d_8 + Maj(\boldsymbol{a_4}, \boldsymbol{b_4}, \boldsymbol{c_4}) + \Sigma_0(\boldsymbol{a_4}) \tag{4}$$

$$\boldsymbol{h_4} = d_0 + \boldsymbol{d_4} - Maj(a_0, b_0, c_0) - \Sigma_0(a_0) \tag{5}$$

$$\boldsymbol{g_4} = c_0 + \boldsymbol{c_4} - Maj(\boldsymbol{d_4}, a_0, b_0) - \Sigma_0(\boldsymbol{d_4}) \tag{6}$$

$$\boldsymbol{f_4} = b_0 + \boldsymbol{b_4} - Maj(\boldsymbol{c_4}, \boldsymbol{d_4}, a_0) - \Sigma_0(\boldsymbol{c_4}) \tag{7}$$

$$\boldsymbol{e_4} = a_0 + \boldsymbol{a_4} - Maj(\boldsymbol{b_4}, \boldsymbol{c_4}, \boldsymbol{d_4}) - \Sigma_0(\boldsymbol{b_4}) \tag{8}$$

$$\boldsymbol{W_0} = \boldsymbol{h_4} - d_0 - h_0 - K_0 - Ch(e_0, f_0, g_0) - \Sigma_1(e_0) \tag{9}$$

$$\boldsymbol{W_1} = \boldsymbol{g_4} - c_0 - g_0 - K_1 - Ch(\boldsymbol{h_4}, e_0, f_0) - \Sigma_1(\boldsymbol{h_4}) \tag{10}$$

$$\boldsymbol{W_2} = \boldsymbol{f_4} - b_0 - f_0 - K_2 - Ch(\boldsymbol{g_4}, \boldsymbol{h_4}, e_0) - \Sigma_1(\boldsymbol{g_4}) \tag{11}$$

$$\boldsymbol{W_3} = \boldsymbol{e_4} - a_0 - e_0 - K_3 - Ch(\boldsymbol{f_4}, \boldsymbol{g_4}, \boldsymbol{h_4}) - \Sigma_1(\boldsymbol{f_4}) \tag{12}$$

$$\boldsymbol{W_4} = h_8 - \boldsymbol{d_4} - \boldsymbol{h_4} - K_4 - Ch(\boldsymbol{e_4}, \boldsymbol{f_4}, \boldsymbol{g_4}) - \Sigma_1(\boldsymbol{e_4}) \tag{13}$$

$$\boldsymbol{W_5} = g_8 - \boldsymbol{c_4} - \boldsymbol{g_4} - K_5 - Ch(h_8, \boldsymbol{e_4}, \boldsymbol{f_4}) - \Sigma_1(h_8) \tag{14}$$

$$\boldsymbol{W_6} = f_8 - \boldsymbol{b_4} - \boldsymbol{f_4} - K_6 - Ch(g_8, h_8, \boldsymbol{e_4}) - \Sigma_1(g_8) \tag{15}$$

$$\boldsymbol{W_7} = e_8 - \boldsymbol{a_4} - \boldsymbol{e_4} - K_7 - Ch(f_8, g_8, h_8) - \Sigma_1(f_8) \tag{16}$$

IV. Computational Cost of Inversion

The computational complexity of reversing the first eight rounds of SHA-256 is comparable to that of the corresponding forward computation. The inversion procedure employs approximately the same number of bitwise functions (*Ch*, *Maj*, Σ_0, Σ_1) and modular arithmetic operations as those used in the original compression function.

This suggests that the inversion process does not introduce significant overhead. However, empirical benchmarks are required to rigorously validate this observation and to quantify performance differences under practical conditions.

V. Extensibility to SHA-512

SHA-512, as part of the SHA-2 family, extends the structural design of SHA-256 to operate on 64-bit words rather than 32-bit words. This architectural shift results in a hash output size of 512 bits, doubling that of SHA-256. While the message block size increases from 512 bits in SHA-256 to 1024 bits in SHA-512, the underlying Merkle–Damgård construction and Davies–Meyer compression model remain conceptually unchanged.

The SHA-512 message schedule expands the 1024-bit message block into 80 64-bit words, compared to the 64 32-bit words used in SHA-256. The SHA-512 compression function processes 80 rounds accordingly, utilizing round-specific constants tailored to the 64-bit domain.

In addition to the differences noted above, SHA-512 also employs a distinct set of eight initial hash value constants. These are derived from the fractional parts of the square roots of the first eight prime numbers and serve to initialize the internal state of the compression function—much like in SHA-256, but adapted to the 64-bit setting.

Although the specific bitwise operations—such as the Σ (uppercase sigma) and σ (lowercase sigma) functions—are redefined to accommodate 64-bit words, their functional behavior and purpose closely mirror those in SHA-256. As such, the core logic of the compression function remains invariant between the two variants.

Given this close structural correspondence, the deterministic inversion approach presented for the first eight rounds of SHA-256 can, in principle, be extended to SHA-512 with appropriate adaptations to account for the expanded word size, updated constants, and modified bitwise functions.

VI. Conclusion

We were able to efficiently compute and subtract (modulo 2^{32}) all necessary values from the SHA-256 message digest to deterministically recover the original message schedule for the first eight rounds, assuming a single message block.

Given two specific states in the compression process separated by eight rounds, a single solution exists. That is, there is only one unique set of message schedule words that could have produced that transition.

After eight rounds of SHA-256, despite extensive mixing, bitwise and nonlinear operations, the variable information used by its compression function remains uncompressed: the message schedule persists and is fully reversible.

Therefore, SHA-256 reduced to eight rounds does not meet the criteria of a one-way hash function and instead exhibits characteristics closer to a reversible transformation.

As SHA-512 functions at its core in the same manner, a similar inversion method can be applied.

References

[1] National Institute of Standards and Technology (NIST), "FIPS PUB 180-4: Secure Hash Standard (SHS)," 2015.