cryptography

Article

Strict Avalanche Criterion of SHA-256 and Sub-Function-Removed Variants

Riley Vaughn [1,*] and Mike Borowczak [2]

[1] Department of Electrical Engineering and Computer Science, University of Wyoming, Laramie, WY 82071, USA
[2] Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL 32816, USA; mike.borowczak@ucf.edu
* Correspondence: rvaughn4@uwyo.edu

Abstract: The measure of diffusion, the property of dissipating patterns and statistical structures in cryptographic transformations, serves as a valuable heuristic for assessing the obscurity of patterns that could lead to collisions. As with many cryptographic hash functions, SHA-256 is thought to exhibit the property of diffusion. While SHA-256's diffuse output is loosely documented, even less is known about how the diffusion rate changes across the 64 rounds in its compression function and how the algorithm's individual sub-functions contribute to the overall diffusion. The diffusion of the unmodified compression function is initially measured using the Strict Avalanche Criterion (SAC), with the aim of understanding the alteration in diffusion across the 64 rounds of compression. The level to which sub-functions affect diffusion is subsequently measured, enabling potential prioritization of these sub-functions in future collision attacks. To accomplish this, the compression function is modified by removing sub-functions, and the diffusion of these new variants is measured. While the SAC measurements of each function eventually plateau close to the 50% target, no function, including the unmodified compression function, strictly meets the SAC, and multiple variant functions diffuse at comparatively slower rates.

Keywords: SHA-256; Strict Avalanche Criterion; Avalanche Effect; hash function

Citation: Vaughn, R.; Borowczak, M. Strict Avalanche Criterion of SHA-256 and Sub-Function-Removed Variants. *Cryptography* **2024**, *8*, 40. https://doi.org/10.3390/cryptography8030040

Academic Editors: Josef Pieprzyk and Christoforos Ntantogian

Received: 28 July 2024
Revised: 23 August 2024
Accepted: 5 September 2024
Published: 8 September 2024

1. Introduction

The cryptographic hash function SHA-256 is a member of the SHA-2 family of functions, initially standardized by NIST in FIPS 180-2 [1] and since superseded in FIPS 180-4 [2]. Even with the standardization of the SHA-3 family, particularly SHA3-256, SHA-256 continues to be extensively employed: within the TLS protocol [3], for authenticating software packages, and as a significant component of the proof-of-work algorithm in the Bitcoin protocol [4].

A cryptographic hash function is an "easy to compute but hard to invert" function that maps arbitrary length input into fixed length output and satisfies the properties of collision resistance, preimage resistance, and second-preimage resistance [5]. In practice, collision resistance implies both preimage resistance and second-preimage resistance [5], so collision resistance is often the priority concern of the security analyst.

While not a rigorously defined property of a cryptographic hash function, the measure of the dissipation of patterns and statistical structure, referred to as as diffusion [6], is frequently correlated with the measure of collision resistance. Patterns that produce collisions are more obvious in transformations with low measures of diffusion than amongst those with high measures of diffusion, and thus predicting collisions becomes practical in low diffusion transformations. The Avalanche Effect [7] and the stronger Strict Avalanche Criterion [8] are proposed quantitative measures of diffusion.

1.1. The Strict Avalanche Criterion (SAC)

The Strict Avalanche Criterion (SAC) of a cryptographic transformation is a combination of measurements: completeness and the Avalanche Effect [8]. For a cryptographic transformation to be complete, each bit of output must depend on all bits of input. To exhibit the Avalanche Effect, a cryptographic transformation must, on average, flip half of it is output bits when a single input bit is flipped. Therefore, to satisfy the SAC, "each output bit should change with probability of one half whenever a single input bit is complemented [8]." Here, the Hamming distance is represented by H, for a cryptographic transformation f, and the Avalanche Effect is defined [9] as follows:

$$\forall x, y H(x,y) = 1, \text{avg}(H(f(x), f(y)) = \frac{n}{2}. \tag{1}$$

Similarly, where a binomial distribution is represented by B, the Strict Avalanche Criterion is defined as

$$\forall x, y H(x,y) = 1, H(f(x), f(y)) \approx B(\frac{1}{2}, n). \tag{2}$$

In practice, for modern cryptographic transformations, it is infeasible to compute the test for all (x, y) pairs. Rather, a large k value is chosen, and the test is computed k times, such that the result is statistically significant.

The SAC of cryptographic transformation f, which inputs an n length plaintext vector and outputs an h length ciphertext vector, produces an $n \times h$ dependency matrix $\mathbf{A}$ [8]. First, each element of $\mathbf{A}$ is initialized to 0. Then, a plaintext vector $\mathbf{x}$ of length n is chosen at random, and the corresponding ciphertext vector $\mathbf{y} = f(\mathbf{x})$ is computed. For each bit i of the plaintext, a modified plaintext $\mathbf{x}_i$ is generated, such that $\mathbf{x}_i$ is equivalent to $\mathbf{x}$, except for the complimented bit at index i. These modified plaintext vectors $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$ form a set of size n. The corresponding modified ciphertext vectors $(\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n)$ are computed where $\mathbf{y}_i = f(\mathbf{x}_i)$. Using $\mathbf{v}_i = \mathbf{y} \oplus \mathbf{y}_i$, where $\oplus$ is the XOR function, the set of avalanche vectors $(\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n)$ are computed. The set of avalanche vectors are added to $\mathbf{A}$ such that the j-th element of the avalanche vector $\mathbf{v}_i$ is added to the i-th row, j-th column of the $\mathbf{A}$. The choice of plaintext vector is repeated k times, and finally each element in the matrix is divided by k.

The n rows of $\mathbf{A}$ correspond to the n bits of input, with one complimented per row. The h columns correspond to the h bits of output. The values in the matrix range from $[0 - 1]$, where a 0 value represents an output bit that is independent of the change in input, and a 1 value represents a bit completely dependent on the change in input [8]. To meet the criteria of completeness, no element in $\mathbf{A}$ ought to be 0. To meet the criteria for the Avalanche Effect, each row in $\mathbf{A}$ must average to $1/2$. Therefore, a cryptographic transformation meets the Strict Avalanche Criterion if every element in $\mathbf{A}$ is close to $1/2$, for a sufficiently large k.

1.2. Related Works

The diffuse output of the entirety of SHA-256 is loosely measured in previous works [10,11]. Gilbert and Handschuh do not use a specific algorithm to measure diffusion. Rather, they describe how specific sub-functions introduce diffusion: integer addition and Sigma functions. Upadhyay et al. measured a variety of hash functions using the Bit Independence Criterion and SAC. They found that in an k = 8000 bit-string test, when measuring strictly the base function, SHA-256 does not pass the SAC. This result suggests that SHA-256 might contain obscure patterns, which, if discovered, could be used as the basis for future collision attacks.

Yoshida and Biryukov [12] analyze a variant of SHA-256 in which integer addition is replaced with the XOR operation. This analysis was carried out, in part, as a potential stepping stone. Although there is no straightforward way to convert the SHA-256-XOR attacks into true SHA-256 attacks by studying the variant function, the strengths of the carry operation of integer addition became obvious. Therefore, if an attack would be able to focus on this specific carry operation, the attacks on this variant function might translate to the original function.

Previous works in this area of cryptanalysis [13–15] use differential cryptanalysis to find collisions and pseudo-collisions on round-reduced SHA-256 variants. Differential cryptanalysis is currently the main tool in the cryptanalysis of hash functions. The most recent of these works [15] finds collisions for a 31-round SHA-256 and pseudo-collisions for 39-round SHA-256.

1.3. Our Contribution

Unlike previous work [10,11] in SHA-256 diffusion measurement, this work investigates the SHA-256 compression function and its sub-functions, as well as the message scheduler. The SAC of the base compression function is easily measured, and by removing individual sub-functions and specifically measuring the progress of the SAC throughout the iterative rounds of compression, the strength of diffusion of sub-functions is determined.

Inspired by the XOR variant introduced by Yoshida and Biryukov [12], a message scheduler-removed variant is introduced. Following the finding that the message scheduler introduces limited non-redundant diffusion, a sub-function-removed variant is introduced for each identified sub-function: message scheduler, MAJORITY, CHOOSE, Σ_0, Σ_1, integer addition (+), and the K constant.

This work does not implement differential cryptanalysis, yet there is considerable reason to believe that SAC measurements might provide a justification for future ad hoc differential cryptanalysis. Similarly, sub-function-removed variants might provide useful targets for differential cryptanalysis as stepping stones for full SHA-256 collision attacks.

Damgard [16] suggests that with a proper padding procedure, if a compression function is collision- and pseudo-collision-resistant, then the containing hash function is also collision-resistant. Since measures of diffusion are correlated with measures of collision resistance, a measure of the SAC through the 64 rounds of compression should loosely inform us about the progression of collision resistance throughout the iterative process.

Therefore, the SAC of the 64 rounds of compression with the involvement of the message scheduler is measured. From these measurements, 64 dependency matrices are obtained, which are then parsed and reduced using simple data analytical tools. The SAC is then applied, alongside similar data analytics, to the 64 rounds of compression without the message scheduler. Lastly, the compression function is stripped of individual component sub-functions, and the SAC is measured on these new variant compression functions for the purpose of identifying the change in diffusion of individual sub-functions throughout the 64 rounds. The resultant measurements are used to identify sub-functions which contribute most to the SAC of each compression round and therefore the SAC of the overall function.

2. SHA256

The SHA-256 hash function receives a message m of length n, where n is represented as an unsigned 64-bit integer. In order to assure the total message length l is a multiple of 512 bits, a single '1' bit is first appended to the message followed by variable '0' bits, such that $l \equiv 448 \pmod{512}$. The original message length n is then appended to the message, such that $l \equiv 0 \pmod{512}$. This padding process results in non-ambiguous padding, which when concatenated with the original message will always form a unique combined message [10].

The padded message is then split into 512-bit blocks, where each block is passed through a message scheduler. The scheduler lengthens and jumbles the 512-bit block into 64, 32-bit blocks. For each 32-bit block, the compression function is run, where the output of one round is used as the input for the next. The output of the 64th round is used either as the initialization vector for the next block or the final output of the function.

2.1. Message Scheduler

The SHA-256 message scheduler inputs a 512-bit vector, split into 16, 32-bit words. The scheduler expands the message into a 2048-bit vector with 64, 32-bit words using the recurrence formula $W_t = \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16}$. The use of integer addition (+), instead of only XOR ($\oplus$), complicates the recurrence relation with the carry operation.

Sigma Functions (σ_0 and σ_1)

Defined as $\sigma_0(X) = ROTR^7(X) \oplus ROTR^{18}(X) \oplus SHR^3(X)$ and $\sigma_1(X) = ROTR^{17}(X) \oplus ROTR^{19}(X) \oplus SHR^{10}(X)$, respectively. Handschuh and Gilbert [5] show that both σ functions are linear and injective in GF(2), a property which prevents internal collisions.

2.2. Compression Function

During each of its 64 rounds of compression, the SHA-256 compression function employs six sub-functions which contribute to some level of diffusion: MAJORITY, CHOOSE, Σ_0, Σ_1, integer addition (+), and the K constant.

After the message scheduler has processed the message block, the compression function inputs that block of 64, 32-bit words and an IV. The IV is split into 8 variables (a–h), and for each of the 64 words, a round occurs (see Figure 1). The output of the 64th round is added with the IV, which in turn becomes either the IV for the next block or the final output.

$$\begin{aligned}
&T_1 = h + \Sigma_1(e) + Ch(e,f,g) + K_t + W_t\\
&T_2 = \Sigma_0(a) + Maj(a,b,c)\\
&h = g\\
&g = f\\
&f = e\\
&e = d + T_1\\
&d = c\\
&c = b\\
&b = a\\
&a = T_1 + T_2
\end{aligned}$$

Figure 1. One round of the SHA-256 compression function.

2.2.1. MAJORITY and CHOOSE

Defined as $Maj(X,Y,Z) = (X \wedge Y) \oplus (X \wedge Z) \oplus (Y \wedge Z)$ and $Ch(X,Y,Z) = (X \wedge Y) \oplus (\neg X \wedge Z)$, respectively. The non-linear functions use three bits of input to decide one bit of output, which when repeated many times, has the effect of diffusing the effects of the input across the eventual output. The non-linearity also complicates the analysis of the recurrence relation of SHA-256.

2.2.2. Sigma functions (Σ_0 and Σ_1)

Defined as $\Sigma_0(X) = ROTR^2(X) \oplus ROTR^{13}(X) \oplus ROTR^{22}(X)$ and $\Sigma_1(X) = ROTR^6(X) \oplus ROTR^{11}(X) \oplus ROTR^{25}(X)$. The Σ functions are linear and injective mappings in $GF(2)$. This fact is elegantly shown [5] by representing 32-bit words as elements of $GF(2)[X]/X^{32} + 1$, and Σ_0 and Σ_1 as polynomials $X^2 + X^{13} + X^{22}, X^6 + X^{11} + X^{25} \in GF(2)[X]$, respectively. These polynomials are co-prime with $X^{32} + 1 \in GF(2)[X]$; therefore, the Σ functions are linear and injective mappings in $GF(2)$.

Although not necessary in the permutation layer of a cipher [17], a consensus of design leads to the permutation layer of the cipher often consisting of linear boolean functions. The linear and injective Σ functions are such examples. This property also assists in preventing internal collisions [5].

2.2.3. Integer Addition (+)

Standard integer addition is not linear in $GF(2)$ because of the carry operation, which makes the SHA-256 recurrence relation more difficult to analyze. The carry operation of integer addition introduces diffusion through the a simple permutation layer of occasionally shifting bits.

2.2.4. *K* constants

The values of K “represent the first thirty-two bits of the fractional parts of the cube roots of the first sixty-four prime numbers” [2]. These constants are added in each round as seemingly minor but useful way of complicating the SHA-256 recurrence relation and therefore further diffusing the input.

3. Methods

First, the SAC of the base SHA-256 compression function with the use of the message scheduler is measured, as well as the SAC for each round of the function. These data, although novel, act as the control to which other data are compared. The compression function variants are then stripped of the message scheduler and individual component sub-functions. The SAC is measured on these new variant compression functions, and the results are then compared to the original full compression function.

The plaintext data are generated via the GO Standard Library math/rand package [18] as 1,000,000-length 16 arrays of 32-bit unsigned integers. Each 512-bit plaintext is input into the compression function, and a final ciphertext as well as the state of each of the 64 rounds are output. Then, each bit of each plaintext is complimented, and the new, bit-modified plaintext is input into the compression function. The state of each of the 64 rounds is recorded for each run, and the XOR operation combines each round of this set with each unmodified round state. This process results in a final dependency matrix of size 512×256, as well as $64{,}512 \times 256$ round state dependency matrices.

3.1. SAC Concerns

For the purpose of presenting readable data, when plotting the SAC of the full compression function matrix, only the minimum and maximum change percentages for each bit complimented were plotted: the minimum and maximum element in each row of the dependency matrix. When plotting the entire 64 rounds, the dependency matrices, each 512×256, were reduced to their maximum, mean, and minimum values. These simplifications preserve the most substantial information from each set and allow for readable data. For simplicity, we refer to the percent change in output bits when a single bit of the input is flipped as the SAC data.

When Webster and Tavares [8] first introduced the Strict Avalanche Criterion, they described no concrete specification for what it means to be “close to one half”. A reasonable choice for this measurement is binomial testing using a 99% confidence level. A binomial test checks if a frequency distribution corresponds to an expected distribution. In this case, the test checks if the minimum and maximum cells of each SAC dependency matrix correspond to the target of 50%. If the confidence intervals’ output from these tests contains the target 50%, then the functions exhibits the SAC with 99% confidence. We refer to a function as passing the binomial test if the confidence interval for the test contains 50%.

The per-round SAC minimum and maximum measurements plateau in every case. These plateau values are reported as the values at the initial point of plateau $\pm$, which is the smallest possible value, such that the range of both resultant sums contains all consecutive values. The data are presented as such because values slightly vary at the plateau, as if random noise was present. In all per-round cases, the mean appears to plateau but continues to increase in precision. Although not useful in verifying if a function exhibits the SAC, a binomial test on the mean SAC data can verify the weaker property of exhibiting the Avalanche Effect. Therefore, the per-round mean data are tested, and the first round where the confidence interval contains 50% is reported.

3.2. SHA256 Implementation

The level of algorithm dissection required for measuring the SAC of the SHA-256 compression function was not possible using standard SHA-256 packages, such as the GO crypto/sha256 implementation [19]. Therefore, a modular version of the algorithm was developed [20] and tested with vectors from the NIST Secure Algorithm Validation System [21].

4. Results

4.1. SAC of Compression Rounds with Message Scheduler

For each random plaintext input into the combined message scheduler and compression function, when each bit of the plaintext is individually complimented then ran through the SAC process, on average, over many plaintexts, the minimum and maximum change in output bit per single input bit complimented fall close to the desired 50% confidence interval (Figure 2). The maximum element in the resultant dependency matrix is 50.2119%, and the minimum 49.7688%. Yet, a binomial test with a 99% confidence level on the minimum and maximum elements of the matrix shows that the function does not strictly meet the SAC. The respective confidence intervals range from 49.6400% to 49.8976% and from 50.0831% to 50.3407%, neither of which contains 50%. Given the previously documented [11] SAC measurements on the entire SHA-256 function, it is expected that the measurements of the combined compression function and message scheduler will issue similar results.

The per-round measurement of the SAC of the compression function (Figure 3) shows that the minimum and maximum change in output bit per single input bit complimented remain at 0.0% and 100.0%, respectively, until round 21. These values quickly plateau by round 23, at 49.7845% ± 0.0289% and 50.2024% ± 0.0411% (Table 1), respectively. Meanwhile, the mean of the elements throughout the rounds appears as a steep s-curve, reaching 49.9682% by round 21, where the confidence interval of the binomial test ranges from 49.8394% to 50.0971%. Unlike the minimum and maximum values, which vary slightly in precision at the plateau as if through random noise, the mean becomes increasingly precise, reaching 49.99997% by round 64 (Table 2).

The analysis of these SAC measurements reveals little in terms of novel cryptanalysis of SHA-256. Known collision attacks against step-reduced variants of the SHA-2 family, dating back as far as 2008 [13], exceed the 22 rounds where knowledge of a lack of diffusion might assist in finding collisions.

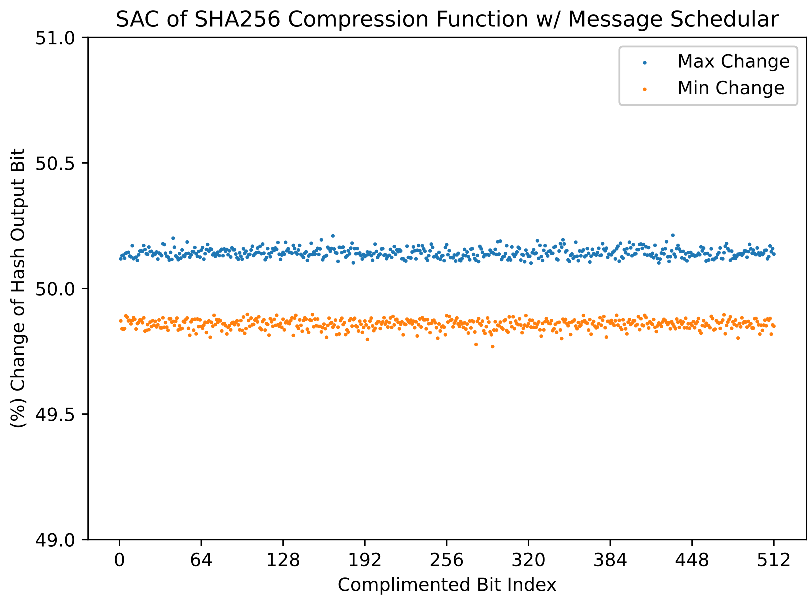

Figure 2. The minimum and maximum values from each row of the SAC dependency matrix of the full compression function with the message scheduler.

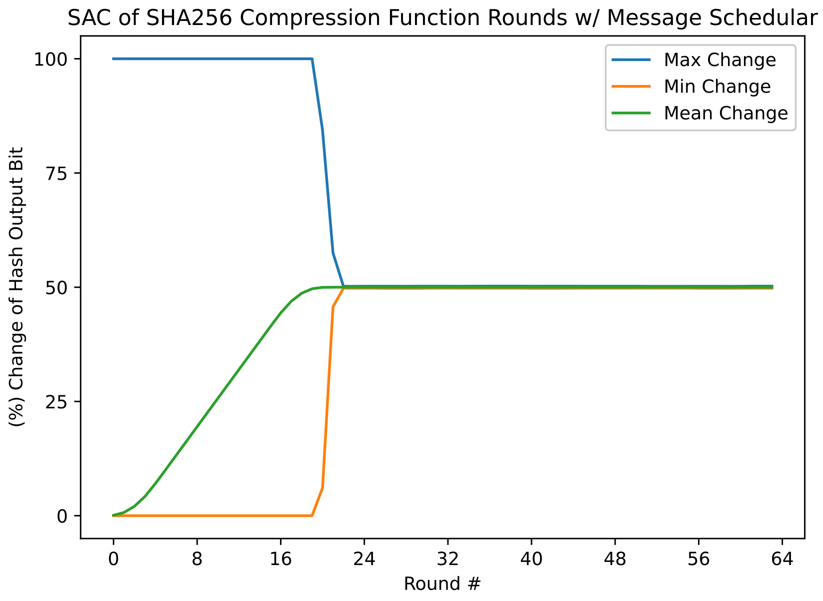

Figure 3. The minimum, maximum, and mean values from each round's SAC dependency matrix with the message scheduler.

4.2. SAC of Compression Rounds without Message Scheduler

In order to measure the SAC of the compression function without the use of the message scheduler, using the same plaintext vectors, the message scheduler must be replaced with a function which extends a 512-bit plaintext vector into a 2048-bit plaintext vector. The most sensible replacement is to simply define a message scheduler with recurrence relation $W_t = W_{t-16}$. This leads to a simple, four repetitions of the original plaintext.

Measurement of the SAC of the full compression round without the use of the message scheduler results in a plot similar to that of the previous measurement (Figure 4). The minimum and maximum changes in output bit per single input bit complimented are 49.7912% and 50.208%, respectively. Similar to the compression function with the use of the message scheduler, a binomial test with a 99% confidence level on the minimum and maximum elements of the matrix shows that the function does not strictly meet the SAC. The respective confidence intervals range from 49.5834% to 49.8411% and 50.0973% to 50.3549%, neither of which contains 50%.

The per-round measurements of the SAC of the compression function without the message scheduler (Figure 5) are also similar to the SAC measurements with the message scheduler. The minimum and maximum changes in output bit per single input bit complimented remain at 0.0% and 100.0%, respectively, until round 21, plateauing by round 23 at 49.7122% $\pm$ 0.0874% and 50.2261% $\pm$ 0.0307% (Table 1), respectively. The mean of the elements throughout the rounds similarly appears as a steep s-curve, reaching 49.9638% by round 21 and increasing in precision to 49.999996% by round 64 (Table 2). At round 21, the confidence interval, 49.8350% to 50.0926%, of the binomial test on the mean of this variant has a 50% confidence level.

There being no significant difference in diffusion amongst this sub-function-removed variant and the original function is not obvious, as the message schedule is often speculated to be a considerable source of diffusion [10]. It seems that SHA-256 contains significant redundancy in diffusion, such that the removal of the message scheduler has very little effect on this property.

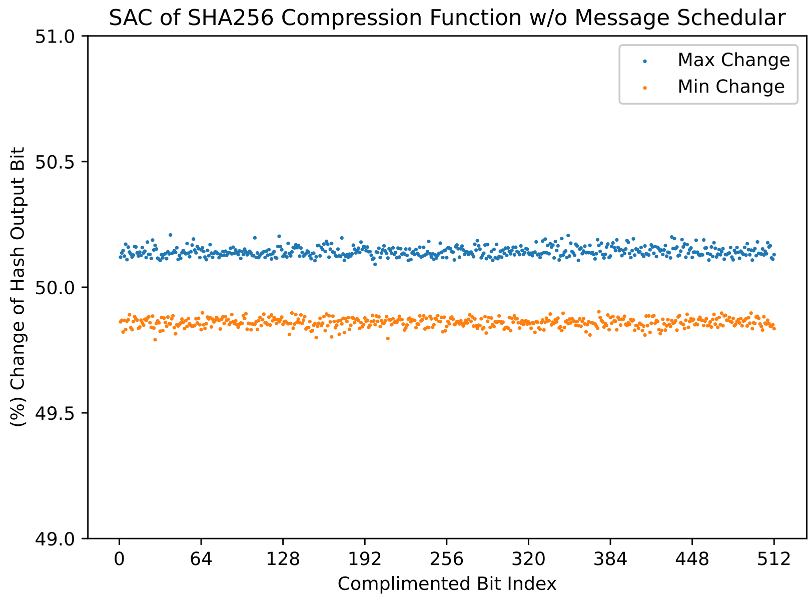

Figure 4. The minimum and maximum values from each row of the SAC dependency matrix of the full compression function without the message scheduler.

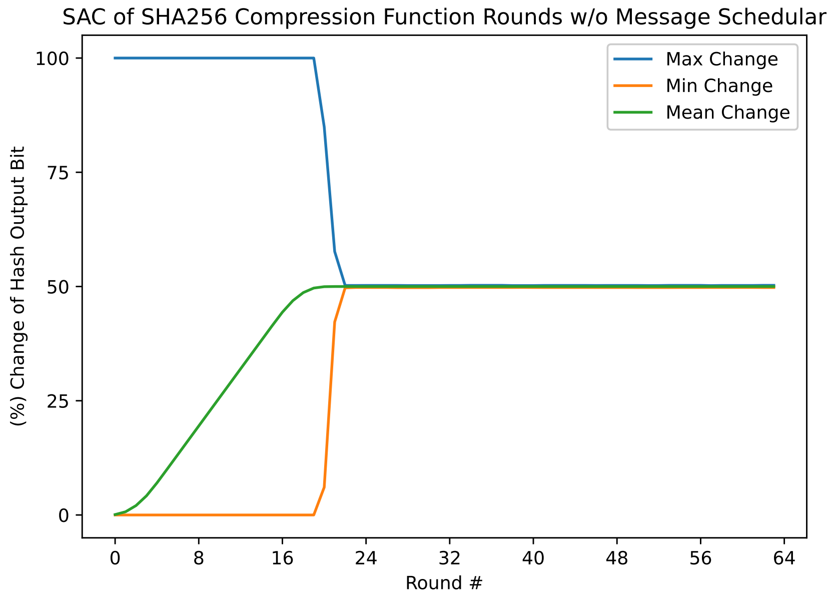

Figure 5. The minimum, maximum, and mean values from each round's SAC dependency matrix without the message scheduler.

4.3. SAC of Compression Function with Sub-Function-Removed Variants

The result of the previous measurement compels further investigation: what is the effect of removing other sub-functions on the measure of diffusion? The SAC is measured for 6 other sub-function-removed variants: CHOOSE-removed (CR), MAJORITY-removed (MR), Σ_0-removed (S0R), Σ_1-removed (S1R), K constant-removed (KR), and integer addition-removed (AR) variants. In all cases except integer addition, the sub-function is outright removed. Integer addition is replaced with XOR, as XOR comprises integer addition without a carry operation.

As is evident in Figure 6, most sub-function-removed variants, including MR, KR, S0R, and message schedule-removed variants, provide similar SAC metrics as to the unmodified compression function. The CR, AR, and S1R variants differ significantly, as shown in Table 1 and Table 2.

While the minimum and maximum changes in output bit per single input bit complimented remain at 0.0% and 100.0%, respectively, until round 21 for the CR variant, the minimum for the next round is 1.3664% and the maximum is 98.6421%. Compared to the normal, unmodified compression function, which at round 21 has a minimum at 6.0623% and maximum at 84.3895%, the CR variant approaches a plateau at a steeper curve. The minimum and maximum values of the CR variant plateau at 49.7740% ± 0.0232% and 50.2066% ± 0.0290%, respectively, while the mean of the CR variant passes the binomial test at round 21 with a confidence interval of 49.7918% to 50.0494%. Like the unmodified compression function, the variant continues in precision, reaching 49.9998% by round 64.

The minimum and maximum elements of the AR variant's per-round dependency matrices remain at 0.0% and 100.0%, respectively, until round 23, not approaching the target percentage until round 25. The minimum and maximum SAC values plateau at 49.7740% ± 0.0418% and 50.2103% ± 0.0409%, respectively. The mean, resembling a slightly less steep s-curve, passes the binomial test at round 22 with a confidence interval of 49.7431% to 50.0007%. At round 64, the mean reaches 50.0001%. Comparing these values to the unmodified function shows a general two-round difference in diffusion: the AR variant is two rounds slower at diffusing the input in the output.

Slowest to diffuse, the minimum and maximum values of the measurement of the SAC on the S1R variant remain at 0.0% and 100.0%, respectively, until round 24, at which they change slowly, reaching the plateau only by round 27. The minimum and maximum SAC values plateau at 49.7680% ± 0.0309% and 50.2057% ± 0.0368%, respectively. The mean passes the binomial test at round 24, with a confidence interval of 49.8180% to 50.0756%. At round 64, the mean reaches 50.00009%. The mean only reaches plateau by round 25. Comparing S1R to the unmodified function, S1R diffuses three to four rounds more slowly.

These data show that the CHOOSE sub-function, the carry operation of integer addition, and Σ_1 contribute significantly to diffusion; thus, it might be worthwhile to focus on these candidates when it comes to collision and pseudo-collision attacks on SHA-256.

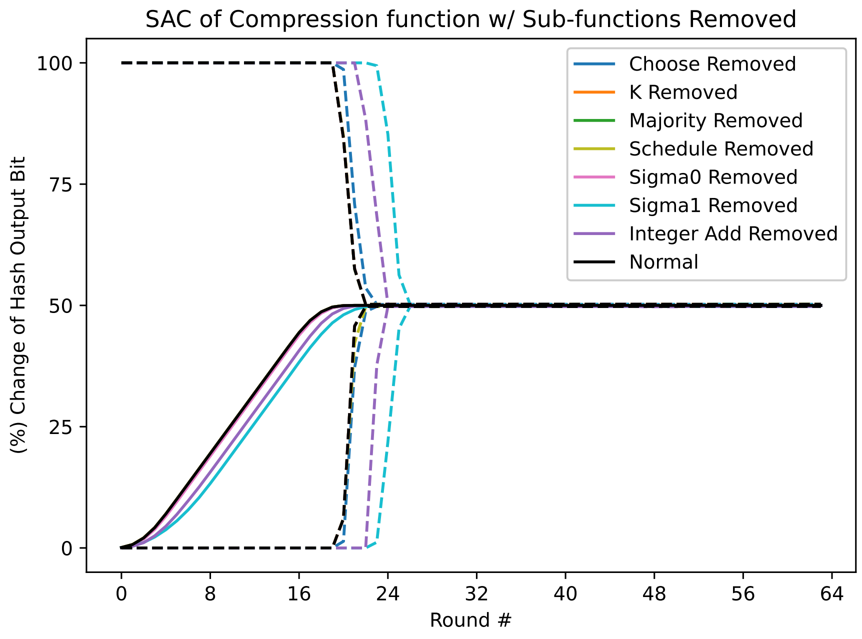

Figure 6. The minimum (Dashed), maximum (Dashed), and mean (Solid) values from each of the 64 SAC dependency matrices of compression function round variants with sub-functions-removed variants.

Table 1. Round and values of min/max plateau for each variant (CR: CHOOSE function-removed, KR: K constant-removed, MR: MAJORITY function-removed, MSR: message scheduler-removed, S0R: Σ_0-removed, S1R: Σ_1-removed, AR: integer addition-removed).

Variant	Round	Min Value (%)	Min ± (%)	Max Value (%)	Max ± (%)
Unmodified	23	49.7845	0.0289	50.2024	0.0411
CR	24	49.7740	0.0232	50.2066	0.0290
KR	23	49.7792	0.0153	50.2073	0.0428
MR	23	49.7685	0.0309	50.2247	0.0252
MSR	23	49.7122	0.0874	50.2261	0.0307
S0R	23	49.7931	0.0427	50.2228	0.0270
S1R	27	49.768	0.0309	50.2057	0.0368
AR	25	49.774	0.0418	50.2103	0.0409

Table 2. Round and values of mean data binomial test pass for each variant (CR: CHOOSE function-removed, KR: K constant-removed, MR: MAJORITY function-removed, MSR: message scheduler-removed, S0R: Σ_0-removed, S1R: Σ_1-removed, AR: integer addition-removed). CI: confidence interval, Mean R64: mean at round 64.

Variant	Round	Mean Value (%)	CI Low (%)	CI High (%)	Mean R64 (%)
Unmodified	21	49.9682	49.8394	50.0970	49.999970
CR	21	49.9206	49.7918	50.0494	49.999842
KR	21	49.9680	49.8392	50.0968	49.999975
MR	21	49.9674	49.8386	50.0962	50.000073
MSR	21	49.9638	49.8350	50.0926	49.999996
S0R	21	49.9407	49.8119	50.0695	49.999956
S1R	24	49.9468	49.8180	50.0756	49.000091
AR	22	49.8719	49.7431	50.0007	49.000123

5. Discussion and Future Works

Our first set of measurements, encompassing the SAC of the compression function with the use of the message scheduler, reveals that the unmodified function does not exhibit the Strict Avalanche Criterion. This is unsurprising as Upadhyay et al. [11] suggest that base SHA-256 does not exhibit the SAC. The second measurement is of the change in SAC values in each of 64 rounds of the compression function with the use of the message scheduler. These measurements once again show that the confidence intervals of minimum and maximum values of the SAC data never contain 50% and thus the function never passes the SAC. The function must lack completeness as the Avalanche Effect is certainly exhibited. This is shown when the confidence interval of the binomial test, applied to the mean of the SAC data, contains 50% at round 21 and continues to do so for all consecutive rounds. The 64 rounds of the unmodified compression function are then used as control data used for comparing variant compression functions.

When compared with the results of the per-round SAC measurements on the unmodified compression function, the compression function with the message scheduler-removed variants issues a surprising result. Since the message scheduler is often speculated to be a considerable source of diffusion [10] for SHA-256, our results show that both sets of data are extremely similar is interesting. Both sets of minimums and maximums plateau at round 23 at similar values. Also, the range of confidence intervals on both sets of means is shown to contain 50% at round 21, with similar mean and confidence interval values. These results suggests that SHA-256 contains significant redundancy in diffusion, such that the removal of the message scheduler does not noticeably reduce diffusion.

The SAC measurements of six additional sub-function-removed variants of the SHA-256 compression function show our most significant results. When compared to the unmodified function, the CHOOSE, integer addition, and Σ_1 sub-function-removed variants differ most significantly. The minimum and maximum SAC measurements of these functions plateau at rounds 24, 25, and 27, respectively, while the mean SAC data of the CHOOSE-removed variant passes the binomial test at round 21 (the same as the unmodified function), integer addition passes it at round 22, and Σ_1 passes it at round 24. These variations suggests that CHOOSE, integer addition, and Σ_1 are the sub-functions that have the greatest effect on SAC measurements and therefore diffusion.

An analysis of the causative elements of diffusion in the CHOOSE, Σ_1, and integer addition sub-functions will be an important and interesting future undertaking. It is non-obvious why these specific sub-functions contribute more to the diffusion of SHA-256 than other sub-functions. Specifically when in an early paper [10], Gilbert and Handschuh suggest the message scheduler a significant source of diffusion.

Previous works [13,14] show strong results on attacking round-reduced variants of SHA-256 using differential cryptanalysis techniques; whoever, these results do not work to convey the weaknesses of individual sub-functions. The relative diffusive strength of the CHOOSE, integer addition, and Σ_1 functions indicates that they are important targets for future attacks. This strength of diffusion provides a justification for future ad hoc differential cryptanalysis with the aim of targeting said sub-functions. If, through these means, the diffusion of one or more of these sub-functions is negated, the overall function would be proportionately weaker. Alternatively, targeting sub-function-removed variants with differential cryptanalysis could potentially provide an important stepping stone for finding collisions in SHA-256.

6. Conclusions

The measure of diffusion is often correlated with the measure of collision resistance of a hash function. When a hash function measures low in diffusion, it becomes easier to predict collisions, as the patterns that produce collisions become more obvious. One commonly used quantitative measure of diffusion is the Strict Avalanche Criterion, and the diffusion of a hash function is directly tied to the diffusion of its compression function and message scheduler.

The SHA-256 compression function does not exhibit the Strict Avalanche Criterion, though it does exhibit the Avalanche Effect. Measures of the SAC data from the 64 rounds of SHA-256's compression function show that the mean reaches 49.6670% by round 23 and continues to become more precise thereafter. Similar findings are true when the SAC is measured on the message scheduler-removed variant. Though disregarding the message scheduler did not significantly reduce diffusion, removing other sub-functions did have some effect. The CHOOSE, integer addition, and Σ_1 sub-functions of SHA-256 provide relatively stronger diffusion when compared to similar sub-functions.

Author Contributions: Conceptualization, R.V. and M.B.; methodology, R.V. and M.B.; software, R.V.; validation, R.V.; formal analysis, R.V.; investigation, R.V.; resources, M.B.; data curation, R.V.; writing—original draft preparation, R.V.; writing—review and editing, R.V. and M.B.; visualization, R.V.; supervision, M.B.; project administration, R.V.; funding acquisition, M.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by IO Global.

Data Availability Statement: A subset of the data is available at [20]. The data totals 500GB, and thus cannot be conveniently uploaded. Nonetheless, the data is available from the authors upon request.

Conflicts of Interest: The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. *NIST:180-2*; FIPS pub 180-2 Secure Hash Standard. National Institute of Standards and Technology: Gaithersburg, MD, USA, 2002.
2. *NIST:180-4*; FIPS pub 180-4 Secure Hash Standard. National Institute of Standards and Technology: Gaithersburg, MD, USA, 2015.
3. Rescorla, E. *The Transport Layer Security (TLS) Protocol Version 1.3*; RFC Editor: Marina del Rey, CA, USA, 2018. [CrossRef]
4. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 7 September 2024).
5. Handschuh, H.; Gilbert, H. The Evaluation Report of SHA-256 Crypt Analysis Hash Function. In Proceedings of the 2009 International Conference on Communication Software and Networks, Chengdu, China, 27–28 February 2009; pp. 588–592. [CrossRef]
6. Shannon, C.E. Communication theory of secrecy systems. *Bell Syst. Tech. J.* **1949**, *28*, 656–715. [CrossRef]
7. Feistel, H. Cryptography and Computer Privacy. *Sci. Am.* **1973**, *228*, 15–23. [CrossRef]
8. Webster, A.F.; Tavares, S.E. On the design of S-Boxes. *LNCS* **1985**, *218*, 523–534. [CrossRef]
9. Castro, J.C.H.; Sierra, J.M.; Seznec, A.; Izquierdo, A.; Ribagorda, A. The strict avalanche criterion randomness test. *Math. Comput. Simul.* **2005**, *68*, 1–7. [CrossRef]
10. Gilbert, H.; Handschuh, H. Security Analysis of SHA-256 and Sisters*. *LNCS* **2004**, *3006*, 175–193. [CrossRef]
11. Upadhyay, D.; Gaikwad, N.; Zaman, M.; Sampalli, S. Investigating the Avalanche Effect of Various Cryptographically Secure Hash Functions and Hash-Based Applications. *EEE Access* **2022**, *10*, 112472–112486. [CrossRef]
12. Yoshida, H.; Biryukov, A. Analysis of a SHA-256 Variant. In *Selected Areas in Cryptography. SAC 2005. Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3897. [CrossRef]
13. Sanadhya, S.K.; Sarkar, P. New Collision attacks Against Up To 24-step SHA-2. *LNSC* **2008**, *5365*, 91–103. [CrossRef]
14. Lamberger, M.; Mendel, F. Higher-Order Differential Attack on Reduced SHA-256. 2011. Available online: https://eprint.iacr.org/2011/037 (accessed on 7 September 2024).
15. Li, Y.; Liu, F.; Wang, G. New Records in Collision Attacks on SHA-2. *Cryptol. ePrint Arch.* **2024**, Paper 2024/349. Available online: https://eprint.iacr.org/2024/349 (accessed on 7 September 2024).
16. Damgard, I.B. A Design Principle for Hash Functions. In *Advances in Cryptology — CRYPTO' 89 Proceedings*; Springer: New York, NY, USA, 1989. [CrossRef]
17. Liu, Y.; Rijmen, V.; Leander, G. Nonlinear Diffusion Layers. *Des. Codes Cryptogr.* **2018**, *86*, 2469–2484. [CrossRef]
18. Random GO Standard Library. 2024. Available online: https://pkg.go.dev/math/rand (accessed on 7 September 2024).
19. SHA-256 GO Standard Library. 2024. Available online: https://pkg.go.dev/crypto/sha256 (accessed on 7 September 2024).
20. Vaughn, R. 2024. Available online: https://github.com/RileyVaughn/Sha256-SAC (accessed on 7 September 2024).
21. National Institute of Standards and Technology. *The Secure Hash Algorithm Validation System (SHAVS)*. 2014. Available online: https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Algorithm-Validation-Program/documents/shs/SHAVS.pdf (accessed on 7 September 2024).