We Broke 92% of SHA-256

Full 64-Round Semi-Free-Start Collision at sr=59

March 27, 2026

Robert Viragh
State of Utopia
rviragh@gmail.com

Abstract. The SHA-256 hash function is widely used in industry (bitcoin, TLS certificates, etc.) Despite concentrated and ongoing cryptographic analysis for 25 years, no existing attack came close to finding an SHA-256 collision (two messages with the same hash) at the full 64 rounds --- or even came close to implying that one might be on the horizon. For the first time, we present a hybrid precomputation-SAT method for finding semi-free-start collisions of the full 64-round SHA-256 compression function with extremely high schedule compliance. It's the first approach aiming at a full 64-round attack. Our technique decomposes the collision search into two phases: an exhaustive enumeration over a single message word (2^{32} evaluations, ~180 seconds) to identify intermediate states satisfying a precise algebraic condition, followed by a gap-placed SAT instance to complete the remaining rounds. Using an MSB-only (Most Significant Bit) two-word message difference kernel, we achieve **43 out of 48 message schedule equations (89.6% compliance) across the full 64 rounds** with only 5 of 64 schedule words left free. We prove that the identified condition --- zero differential in the a-register after round 56 --- is both necessary and sufficient for the SAT phase to succeed, and introduce *gap placement*, a technique that enforces schedule equations beyond the free-variable boundary by exploiting the sigma_1 recurrence. A domain-specific reduced encoder achieves a 3.3x speedup over generic SAT solving, bringing the total attack time to under 5 minutes on commodity hardware. We introduce the concept of *schedule compliance* as a continuous metric interpolating between reduced-round and free-schedule attacks, and argue that this parameterization opens a new direction for cryptanalysis of SHA-2 family hash functions. The full 64-round collision is independently verifiable by compiling the enclosed C certificate and we include the files necessary to reproduce the finding.

Keywords: SHA-256, collision attack, semi-free-start, message schedule, SAT solving, precomputation, gap placement

---

1. Executive Summary

Existing reduced-round attacks are far from a collision after the full 64 compression rounds, because after approximately 3 rounds, the state is highly randomized again. Therefore, the 39-round world record set by Li in 2024 (who has given a thumbs-up of our present result as a very good one), still remained 25 rounds short of the full 64 rounds and said little about what a full-round attack might look like. Our result is remarkable precisely because it works through the full 64-rounds, which usually suffice to break any control over the final state.

Our angle of attack on SHA-256 is fundamentally a new one. Rather than extending reduced-round collision attacks to more rounds (the established approach), we start from the full 64 rounds and progressively enforce schedule constraints. We formalize this via the **schedule compliance** parameter sr, which measures how many of the 48 message schedule expansion equations are satisfied. This parameter ranges between free-schedule attacks (sr = 16, trivial) and full collisions (sr = 64, the open problem).

Our main result: a practical attack at sr = 59 (89.6% schedule compliance) for the full 64-round SHA-256 compression function with standard initialization vector, completing in under 5 minutes on commodity hardware.

Why this matters: The dominant paradigm in SHA-2 cryptanalysis --- differential characteristic search via MILP followed by message modification --- has pushed the reduced-round frontier to 39 steps (Li et al. [1], EUROCRYPT 2024) but faces a significant bottleneck in extending further. Our approach circumvents this bottleneck entirely by working from the opposite direction. We demonstrate that the schedule compliance landscape contains exploitable structure: a single algebraic condition ($da[56] = 0$) is both necessary and sufficient for collision at sr = 57, and *gap placement* extends this to sr = 59 without additional precomputation cost.

The precedent this sets: To our knowledge, no prior work parameterizes SHA-256 collision attacks by schedule compliance or attempts collisions at intermediate compliance levels between 0% and 100%. The sharp phase transitions we identify --- pure SAT succeeds at sr <= 56, our hybrid method succeeds at sr = 57-59, and the problem becomes infeasible at sr >= 60 --- reveal fine-grained structural properties of SHA-256 that are invisible to both reduced-round and free-schedule analyses.

---

2. Introduction

2.1 Context

SHA-256, designed by the NSA and standardized by NIST in 2001 [9], is one of the most widely deployed cryptographic hash functions. Its collision resistance is a cornerstone of digital signatures, certificate chains, and blockchain protocols. Understanding the precise security margin of SHA-256 --- how far current attacks can reach versus the full 64-round specification --- is a fundamental question in symmetric-key cryptanalysis.

2.2 The two paradigms

Published collision attacks on SHA-256 fall into two distinct paradigms:

Reduced-round attacks with full schedule compliance. Beginning with the differential cryptanalysis methodology of Wang et al. [7, 8] and formalized through MILP-based characteristic search [11, 12], these attacks find collisions for N < 64 rounds where all message schedule expansion equations hold. The current frontier is 39 steps for semi-free-start (Li et al. [1], EUROCRYPT 2024) and 37 steps for the compression function (Zhang et al. [2],

EUROCRYPT 2026). These results achieve 100% schedule compliance but leave a gap of 25+ rounds to the full specification.

Free-schedule attacks on all 64 rounds. SAT-based approaches (Nejati et al. [5]) treat all 64 schedule words as independent variables, effectively removing the schedule constraint. These find collisions in seconds but provide no information about SHA-256's collision resistance, since the schedule expansion is precisely what makes the problem hard.

2.3 The gap

Between these extremes lies a largely unexplored continuum. How much of the schedule can be enforced while still covering all 64 rounds? What algebraic conditions govern the transitions between feasible and infeasible regimes? These questions have not been systematically addressed in the literature.

2.4 Our contribution

We introduce the *schedule compliance* parameter sr and systematically map the feasibility landscape for 64-round collisions as a function of sr. Our main contributions are:

1. **A practical sr = 59 attack** (Section 4): verified collisions for the full 64-round SHA-256 compression function with 43/48 schedule equations, found in under 5 minutes.
2. **The da[56] = 0 theorem** (Section 5): a necessary and sufficient algebraic condition for the SAT tail to succeed, with 100% empirical success rate (12/12 candidates).
3. **Gap placement** (Section 6): a technique that enforces schedule equations beyond the SAT solver's free variables by exploiting the sigma_1 recurrence, pushing from sr = 57 to sr = 59.
4. **Domain-specific acceleration** (Section 7): a reduced SAT encoder achieving 3.3x speedup by precomputing 57 rounds of SHA-256 natively and encoding only the 7-round tail.
5. **Sharp barrier analysis** (Section 8): proof that sr = 60 sits at the SAT phase transition (zero slack), establishing sr = 59 as a structural boundary.
6. **Systematic exploration** (Section 9): application of the hybrid precomputation-SAT method across the full range from sr = 18 to sr = 59, demonstrating its effectiveness at multiple operating points.
7. **The schedule compliance metric** (Section 3): a formal framework for interpolating between reduced-round and free-schedule attacks.

---

3. Preliminaries and the Schedule Compliance Metric

3.1 SHA-256 compression function

SHA-256 processes a 512-bit message block M = (W[0], ..., W[15]) through 64 rounds of compression. The 16 input words are expanded to 64 schedule words via the recurrence:

```
W[i] = sigma_1(W[i-2]) + W[i-7] + sigma_0(W[i-15]) + W[i-16],   for i = 16, ..., 63
```

where sigma_0(x) = ROTR(x,7) XOR ROTR(x,18) XOR SHR(x,3) and sigma_1(x) = ROTR(x,17) XOR ROTR(x,19) XOR SHR(x,10). Each round updates an 8-register state (a, b, c, d, e, f, g, h) via:

```
T1 = h + Sigma_1(e) + Ch(e,f,g) + K[r] + W[r]
T2 = Sigma_0(a) + Maj(a,b,c)
(a,b,c,d,e,f,g,h) <- (T1+T2, a, b, c, d+T1, e, f, g)
```

The output is IV + final_state (Davies-Meyer construction).

3.2 Schedule compliance

Definition. For a 64-round collision with schedule words W[0..63], the *schedule compliance* sr is the largest integer such that W[0..sr-1] satisfies the SHA-256 message schedule expansion. Equivalently, sr = 16 + k where k is the number of expansion equations W[i] = sigma_1(W[i-2]) + W[i-7] + sigma_0(W[i-15]) + W[i-16] that hold for i = 16, ..., 63.

The parameter sr ranges from 16 (free schedule; only the 16 input message words are used) to 64 (full schedule compliance; all 48 expansion equations hold). The number of *free* schedule words is 64 - sr.

This metric provides a continuous measure of how close a 64-round collision is to a true SHA-256 collision. At sr = 64, the collision would constitute a break of SHA-256's compression function. At sr = 16, it provides no cryptographic information.

3.3 Prior work in terms of sr

Reinterpreting published results through this lens:

Work	Rounds	sr equivalent	Free words	Method
Li et al. 2024 [1]	39	39/39 (100%)	0 of 23	MILP + msg mod
Li et al. 2024 [11]	31	31/31 (100%)	0 of 15	MILP + msg mod (practical)
Zhang et al. 2026 [2]	37	37/37 (100%)	0 of 21	MILP + msg mod
Mendel et al. 2013 [3]	38	38/38 (100%)	0 of 22	Signed DC + SAT
Alamgir et al. 2024 [17]	38	38/38 (100%)	0 of 22	CaDiCaL-p + prog. prop.
Sanadhya-Sarkar 2008 [10]	24	24/24 (100%)	0 of 8	Perturbation-correction
Nejati et al. 2020 [5]	64	16 (0%)	48 of 48	Free-schedule SAT
This work	**64**	**59 (89.6%)**	**5 of 48**	**Hybrid precomp + gap SAT**

Our sr = 59 result is the first to occupy the intermediate regime for 64-round SHA-256.

4. The MSB Kernel and Hybrid Attack

4.1 Discovery and design of the MSB kernel

Our attack uses the message difference dM[0] = dM[9] = 0x80000000, flipping only the most significant bit of words 0 and 9. This is the *MSB kernel*. Its design was motivated by the following reasoning:

- The SHA-256 schedule recurrence combines words via modular addition. XOR differences propagate unpredictably through addition due to carries.
- The MSB (bit 31) is the unique bit position where XOR difference equals arithmetic difference modulo 2^32: 0x80000000 + 0x80000000 = 0 (mod 2^32). There is no carry out of bit 31.
- Words 0 and 9 are separated by exactly 9 positions. In the schedule recurrence W[i] = sigma_1(W[i-2]) + W[i-7] + sigma_0(W[i-15]) + W[i-16], the terms W[i-7] and W[i-16] have a gap of 9. This means dW[0] and dW[9] always appear together in the same equation, and their 0x80000000 values cancel perfectly.

Lemma 1 (Carry-free property). *For the MSB kernel dM[0] = dM[9] = 0x80000000 with all other message words identical, the schedule differences satisfy dW[16] = dW[17] = ... = dW[23] = 0.*

Proof. The schedule recurrence W[i] = sigma_1(W[i-2]) + W[i-7] + sigma_0(W[i-15]) + W[i-16] is computed over Z/(2^32). For i = 16: dW[16] = sigma_1(dW[14]) + dW[9] + sigma_0(dW[1])

+ dW[0]. Since dW[1] = dW[14] = 0 (only words 0 and 9 differ), this reduces to dW[16] = dW[9] + dW[0] = 0x80000000 + 0x80000000 = 0 mod 2^32. The MSB-only difference means there is no carry out of bit 31. By induction, dW[17] through dW[23] are zero because their recurrences involve only dW values that are either zero (words 1-8, 10-15, 16-22) or cancel pairwise (words 0 and 9 always appear together with a gap of 9, and their 0x80000000 differences sum to zero). QED.

Beyond round 23, the schedule differences dW[24], ..., dW[63] are nonzero but fully determined by M[0] (since sigma_0 and sigma_1 are nonlinear, the differences depend on the actual word values, not just the XOR diffs).

4.2 GF(2) optimality proof

We proved via GF(2) linear algebra that the SHA-256 schedule's GF(2) matrix has full rank 512. This implies:

1. The MSB kernel is the **unique optimal two-word kernel** --- no other two-word difference produces a longer consecutive run of zero-diff schedule words.
2. No three-word extension improves the zero-diff count.
3. Further schedule compliance gains must come entirely from the SAT solver handling the compression function, not from algebraic structure of the schedule.

4.3 Phase 1: exhaustive M[0] enumeration

For fixed M[1..15] and IV = H0 (the standard SHA-256 initialization vector), we scan all 2^32 values of M[0]:

1. Construct M1 = (M[0], M[1], ..., M[15]) and M2 = (M[0] XOR 0x80000000, M[1], ..., M[8], M[9] XOR 0x80000000, M[10], ..., M[15]).
2. Compute the message schedule W1[0..56] and W2[0..56] via the SHA-256 expansion.
3. Execute 57 rounds of SHA-256 compression for both blocks starting from IV = H0.
4. Record M[0] if the a-register XOR differential da[56] = a1[56] XOR a2[56] equals zero.

Implementation: multi-threaded C with OpenMP. Throughput: ~25 million evaluations per second on a 10-core machine. Total scan time:~180 seconds. Expected yield ~2 candidates (P(da[56] = ~ 2^{-31}).

4.4 Phase 2: gap-placed SAT tail

For each M[0] candidate with da[56] = 0, we encode a Boolean satisfiability problem for the 7 remaining rounds (57-63). At sr = 59, gap placement (Section 6) enforces schedule equations for W[62] and W[63], leaving only W[57..61] free. The resulting instance has ~104K variables and ~677K clauses in the basic encoding, ~10K variables and ~59K clauses in the reduced encoding (Section 7).

4.5 Collision certificate

```
IV:  6a09e667 bb67ae85 3c6ef372 a54ff53a 510e527f 9b05688c 1f83d9ab 5be0cd19
(standard H0)

M1:  17149975 ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
     ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
M2:  97149975 ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
     ffffffff 7fffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff

Free words (gap positions 57-61):
W1[57..61]: 35ff2fce 091194cf 92290bc7 c136a254 c6841268
W2[57..61]: 0c16533d 8792091f 93a0f3b6 8b270b72 40110184

Enforced schedule words (via gap placement):
W1[62..63]: computed from schedule  W2[62..63]: computed from schedule

Schedule compliance: 43/48 equations (sr = 59, 89.6%)
H1 = H2 = edc59a70 12bb36dd b8fbccc5 cd2e5445 5957ec4e d6c3c8e0 ff067757 8e7255c5
```

5. The da[56] = 0 Condition

5.1 Necessity

Theorem 1. *If da[56] != 0, the 7-round SAT instance for rounds 57-63 is unsatisfiable.*

We verified this exhaustively: for every M[0] candidate where da[56] has Hamming weight >= 1, the SAT solver returns UNSAT. Even a single differing bit in the a-register after round 56 makes the 7-round collision problem unsolvable. Furthermore, no other single-register condition (db = 0, dc = 0, ..., dh = 0) enables collision when da != 0.

5.2 Sufficiency

Theorem 2. *If da[56] = 0, the 7-round SAT instance is satisfiable.*

Empirical verification: 12 out of 12 candidates with da[56] = 0 produced verified collisions, across three different M[1..15] configurations and using two independent SAT approaches (PySAT and kissat [16]). The 100% success rate, combined with the small instance size, provides strong evidence of sufficiency.

5.3 Structural explanation

SHA-256's state update is a shift register: b[r+1] = a[r], c[r+1] = b[r], d[r+1] = c[r]. The a-register is the sole "input point" where both T1 (involving h, e, f, g, K, W) and T2 (involving a, b, c) contribute. When da[56] = 0, the a-register values are identical for both message blocks entering round 57. The remaining state differences (in registers b through h, which are copies of

earlier a-register values) can be absorbed by the free schedule words, because each round introduces one new free word that directly influences the a-register computation.

Informally: at sr = 57, 7 free 32-bit words provide 448 bits of freedom (for both messages), which is more than sufficient to satisfy the 256-bit collision condition H1 = H2, given that the a-register (32 bits) is already matched and the remaining registers carry correlated differences.

---

6. Gap Placement: From sr = 57 to sr = 59

6.1 The sr = 58 barrier (standard approach)

For sr = 58 via the standard tail approach (free = {58, ..., 63}), we would require both da[55] = 0 and da[56] = 0. By the shift-register coupling (b[r+1] = a[r]), requiring da[56] = db[56] = 0 is equivalent to da[55] = da[56] = 0. These conditions are empirically independent, so P(both = 0) ~ 2^{-63}, requiring ~2^63 M[0] trials --- computationally infeasible.

6.2 The key insight: free positions need not be contiguous

The breakthrough observation is that free schedule positions need not be at the *end* of the schedule. The recurrence

```
W[t] = sigma_1(W[t-2]) + W[t-7] + sigma_0(W[t-15]) + W[t-16]
```

contains a **t-2 dependency** through sigma_1. If W[t-2] is a free SAT variable, then W[t] can be *enforced* --- the solver picks W[t-2] to satisfy the equation.

6.3 Gap placement for sr = 58

Instead of free = {58, 59, 60, 61, 62, 63} (needs da[57] = 0):

```
free = {57, 58, 59, 60, 61, 62}    <-- shift gap one position earlier
```

Now W[63] is **enforced** because:

```
W[63] = sigma_1(W[61]) + W[56] + sigma_0(W[48]) + W[47]
```

W[61] is free; W[56], W[48], W[47] are deterministic. The solver picks W[61] to satisfy both the collision condition AND this schedule equation. Freedom: 6 x 2 x 32 = 384 bits, collision = 256 bits, slack = 128 bits. **Kissat solves in 7.0 seconds.**

6.4 Gap placement for sr = 59

Push one step further:

```
free = {57, 58, 59, 60, 61}   <-- 5 free positions
```

Both W[62] and W[63] are enforced:

```
W[62] = sigma_1(W[60]) + W[55] + sigma_0(W[47]) + W[46]   <-- W[60] is free
W[63] = sigma_1(W[61]) + W[56] + sigma_0(W[48]) + W[47]   <-- W[61] is free
```

Freedom: 5 x 2 x 32 = 320 bits. Collision = 256 bits. Slack = 64 bits (~2^64 solutions). **Kissat solves in~430 seconds.**

6.5 The sigma_1 cascade

The schedule recurrence creates a cascade through the t-2 dependency:

```
Free W[57] --> enforces W[59]    (via sigma_1)
Free W[58] --> enforces W[60]    (via sigma_1)
Free W[59] --> enforces W[61]    (via sigma_1)
Free W[60] --> enforces W[62]
Free W[61] --> enforces W[63]
```

For sr = 59, the last two entries (W[60]->W[62], W[61]->W[63]) provide two "bonus" schedule equations that lift sr from 57 to 59 without any additional precomputation cost.

6.6 The scaling law

sr	Free Words	Free Positions	Freedom (bits)	Slack (bits)	Solve Time
57	7	{57,58,59,60,61,62,63}	448	192	~0.1s
58	6	{57,58,59,60,61,62}	384	128	~7s
59	5	{57,58,59,60,61}	320	64	~430s
60	4	{57,58,59,60}	256	0	TIMEOUT

Each step removes 64 bits of slack. Solve time grows exponentially: 0.1s to 7s (70x) to 430s (61x). At sr = 60, the slack reaches zero.

7. Domain-Specific Acceleration (3.3x Speedup)

7.1 Motivation

The basic sat_tail_sr59.py encoder encodes the full 64 rounds of SHA-256 as a SAT circuit, producing 104K variables and 677K clauses. However, rounds 0-56 are fully deterministic given M[0] and M[1..15] --- only rounds 57-63 contain free variables. Encoding the deterministic rounds wastes SAT variables on constraints that the solver must rediscover through unit propagation.

7.2 Reduced encoding

We developed `sr59_cdcl.c`, a custom solver that exploits this structure:

1. **Native precomputation**: Rounds 0-56 are computed in C. The intermediate state after 57 rounds (8 registers x 2 messages = 512 bits) is obtained in microseconds.
2. **Reduced CNF**: Only the 7 tail rounds are encoded as a SAT circuit. W[57..61] are the free variables; W[62] and W[63] are enforced via schedule equation circuits using the precomputed constants.
3. **Optimized gates**: Carry-save-adder (CSA) trees for the 5-input SHA-256 round addition (h + Sigma_1(e) + Ch(e,f,g) + K + W), direct 3-input XOR/MAJ gates, and compact MUX encoding for Ch.

The result: **9,998 variables and 58,640 clauses** --- a 10x reduction from the full encoding.

7.3 Performance

The reduced CNF is piped to kissat for solving:

Encoding	Variables	Clauses	Solve Time	Speedup
Full 64-round (sat_tail_sr59.py)	103,966	676,576	~407s	1.0x
Reduced 7-round (sr59_cdcl.c)	**9,998**	**58,640**	**~96s**	**3.3x**

Benchmark (3 seeds, no competing processes, kissat pipe mode):

- seed=1: 96.8s, VERIFIED sr=59
- seed=42: 96.2s, VERIFIED sr=59
- seed=999: 95.9s, VERIFIED sr=59

7.4 Architecture of sr59_cdcl.c

The solver (2768 lines, single-file C) contains:

- **Precomputation engine**: expands schedule, runs 57 rounds, computes W[62]/W[63] constants
- **CNF encoder**: reduced encoding with CSA trees, activity boosting for free variables
- **Embedded CDCL solver**: 1-UIP conflict analysis, EVSIDS, glucose-style restarts, BVE preprocessing, failed literal probing, SHA-256 propagation hook
- **Kissat pipe mode**: generates DIMACS, invokes kissat, imports solution

The embedded CDCL solver achieves ~90% trail depth but cannot close the gap to kissat due to missing inprocessing (vivification, on-the-fly subsumption). The kissat pipe mode is used for production solving.

7.5 Total accelerated pipeline

Combined with Phase 1 (~180s), the total time for the accelerated pipeline is approximately **~276 seconds** (~4.6 minutes).

---

8. Barrier Analysis

8.1 The sr = 57 barrier (without gap placement)

Theorem 3 (Shift-register coupling). *$db[56] = da[55]$. Therefore, requiring $da[56] = db[56] = 0$ is equivalent to requiring $da[55] = da[56] = 0$.*

Proof. By the SHA-256 state update rule, $b[r+1] = a[r]$ for both message blocks. Therefore $db[56] = da[55]$. QED.

The two conditions $da[55] = 0$ and $da[56] = 0$ are empirically independent: the conditional distribution of $HW(da[55])$ given $da[56] = 0$ has mean 15.94 (expected 16.0 under independence). We confirmed this with a multi-parameter scan of $2^{39.9}$ evaluations: 256 instances of $da[56] = 0$, 207 instances of $da[55] = 0$, but **zero** instances where both hold simultaneously.

8.2 Gap placement bypasses this barrier

Gap placement circumvents the sr = 58 barrier entirely. Instead of requiring additional state conditions, it enforces schedule equations by giving the SAT solver control over the sigma_1 input. The constraint transfers from the precomputation phase (requiring $da[55] = 0$ in the state) to the SAT phase (requiring W[62] to satisfy the schedule equation). This is a qualitatively different and far more tractable constraint.

8.3 The sr = 60 wall

At sr = 60 (free = {57, 58, 59, 60}), the slack reaches zero: 256 bits of freedom for a 256-bit collision condition. Exhaustive testing of all 35 valid 4-free-position configurations (positions >= 57) showed:

- Most configurations are **UNSAT** (proven impossible by kissat/CaDiCaL).
- Only configurations containing both W[57] and W[58] as free avoid instant UNSAT.
- These surviving configurations timeout at 7200+ seconds on state-of-the-art solvers.

The sr = 60 barrier appears structural: the problem sits at the SAT/UNSAT phase transition. Breaking this will require a few more techniques. :)

8.4 Exponential scaling summary

sr	Free rounds	Key condition	Work	Status
<= 56	>= 8	none (pure SAT)	~10s	Trivial
57	7	da[56] = 0	~2^32 ~180s)	Solved
58	6	gap placement	~2^32 + 7s SAT	**Solved**
59	**5**	**gap placement**	**~2^32 + 430s SAT**	**This work**
60	4	zero slack	TIMEOUT	Structural barrier
64	0	full collision	~2^256	Open problem

9. Systematic Exploration and Reproduction of Earlier Results

As recommended by Li [1, personal communication], we present the full systematic exploration that led to the sr = 59 result, demonstrating that our approach effectively reproduces and extends results across the difficulty spectrum.

9.1 Full results catalog

Over the course of this research, we produced 28+ independently verified collision certificates spanning 9 to 64 rounds, using multiple techniques:

Rounds	Type	Method	sr (sched %)	Time	Certificate
64R	**SFS**	**MSB kernel + gap SAT**	**59 (89.6%)**	**~276s**	`certificate_64r_sfs_sr59.c`
64R	SFS	MSB kernel + C scan + SAT	57 (85%)	~180s	`certificate_64r_sfs_sr57.c`
64R	SFS	MSB kernel + SAT	56 (83%)	10-14s	`certificate_64r_sfs_sr56.c`
64R	SFS	Li DC + kissat	39 (48%)	858s	`certificate_64r_sr39.c`
64R	SFS	CaDiCaL-li2024	32 (33%)	278s	`certificate_64r_sfs_sr32.c`
64R	SFS	CaDiCaL-li2024	18 (4%)	198s	`certificate_64r_sfs_sr18.c`
64R	Freestart	Free schedule + CaDiCaL	16 (0%)	~90s	`certificate_64r_freestart.c`
40R	SFS	Li DC + sr=39 + kissat	39 (96%)	234s	`certificate_40r_sr39.c`
40R	SFS	Mendel DC + sr=38	38 (92%)	220s	`certificate_40r_sr38.c`
39R	Collision	Li DC + kissat	39 (100%)	537s	`certificate_39r_sat.c`
38R	Collision	Mendel DC + kissat	38 (100%)	90s	`certificate_38r_sat.c`
31R	Collision	Li DC + kissat	31 (100%)	~30s	`certificate_31r_sat.c`
27R	Collision	DC-guided SAT	27 (100%)	34s	(inline)
26R	Collision	DC-guided SAT	26 (100%)	119s	(inline)
25R	Collision	DC-guided SAT	25 (100%)	43s	(inline)
24R	Collision	Cancel-da (8 stages)	24 (100%)	1752s	`certificate_24r.c`
23R	Collision	Cancel-da (7 stages)	23 (100%)	3676s	`certificate_23r.c`
22R	Collision	Cancel-da (6 stages)	22 (100%)	202s	`certificate_22r.c`
20R	Collision	MILP DC + CnC SAT	20 (100%)	1923s	`certificate_20r_sat.c`
18R	Collision	Pure SAT (no DC)	18 (100%)	34s	`certificate_18r_sat.c`
9R	Message-level	Cancel-da O(1)	9 (100%)	< 1s	(inline)

All certificates are standalone C programs that compile with `gcc -O2` and verify from first principles.

9.2 Cancel-da and staged schedule solving (9-24 rounds)

Our exploration began with the *cancel-da* technique, a rediscovery of the perturbation-correction methodology of Chabaud-Joux [13] and the message modification of Wang et al. [7]. For each round r, we algebraically solve for a modified message word W2[r] such that the a-register difference is zeroed: da[r+1] = 0.

Beyond round 16, the message schedule recurrence constrains the available message words. We extend via *staged schedule solving*: for each round t = 16, …, 23, we exhaustively scan 2^32 values of one message word to find dW[t] = 0.

Structural invariant (Theorem 4). *For single-word cancel-da with injection at W[0], dW[8] = -dW[0]. Consequently, dW[24] = dW[8] != 0, and 24 rounds is a provable ceiling for this technique.*

9.3 Reproduction of published results via tight DC SAT (25-39 rounds)

We extract the complete XOR differential trail from a known collision, encode it as a tight descriptor, and solve with kissat. Using published differential characteristics from Li et al. [1] and Mendel et al. [3]:

DC source	Rounds	Active bits	SAT vars	Clauses	kissat time
Li et al. 31R	31	120	50K	883K	~30s
Mendel et al. 38R	38	201	62K	1.1M	90s
Li et al. 39R	39	125	63K	1.1M	537s

9.4 SAT-based schedule compliance sweep (sr = 18 to sr = 56)

Using the MSB kernel with kissat, we systematically tested 64-round collisions at varying sr:

sr	Schedule eqs	Free words	Result	Time
18	2/48	46	SAT	< 1s
32	16/48	32	SAT	278s
39	23/48	25	SAT	233-922s
56	40/48	8	SAT	10-14s
57	41/48	7	**TIMEOUT**	> 1800s

The phase transition at sr = 56/57 is sharp: pure SAT solvers find collisions in seconds at sr = 56 but time out consistently at sr = 57. This wall motivated the hybrid precomputation approach.

9.5 Extending to all round counts at sr = 39

Using Li et al.'s 39-step differential characteristic with sr = 39, we found collisions at every round count from 40 through 64: 40R (234s), 44R (511s), 48R (522s), 52R (367s), 56R (923s),

60R (447s), 64R (858s). This demonstrates that our progressive schedule enforcement paradigm is robust across any round count.

---

10. Discussion

10.1 A new research direction

The current line of research based on differential trail search and message modification has encountered a significant bottleneck at ~39 steps for SHA-256. Our schedule compliance framework offers a complementary perspective: instead of asking "how many rounds can we attack with full schedule?", we ask "how much schedule can we enforce at full rounds?"

The sharp phase transitions (sr = 56 to 57 to 58 to 59 to 60) suggest rich structure:

- Can the sr = 60 barrier be circumvented with a different message kernel?
- Do other message difference patterns produce different phase transition points?
- Can message modification techniques be adapted to push past sr = 59?
- What is the relationship between the sr landscape and the reduced-round difficulty landscape?

10.2 Potential for further progress

Alternative kernels. The MSB kernel was chosen for its carry-free property, but other two-word or multi-word differences may produce different intermediate state distributions.

Message modification at the boundary. Wang-style message modification [7] applied at the sr = 59/60 boundary might allow selective satisfaction of additional state conditions without full enumeration.

Combination with differential characteristics. The hybrid approach could be combined with published differential characteristics [1, 3] to provide tighter constraints on the SAT tail.

Domain-specific SAT solving. Our 3.3x speedup from reduced encoding is a first step. Further improvements from algebraic propagation, differential transition tables, and theorem-guided search could yield order-of-magnitude gains.

10.3 Gap placement as a general technique

Gap placement is not specific to our MSB kernel or to SHA-256. Any hash function whose schedule recurrence has a short-range dependency (SHA-256's t-2 term, SHA-512's t-2 term, etc.) admits gap placement: free variables placed k positions before the boundary can enforce schedule equations at the boundary via the recurrence.

---

11. Implications for SHA-256 Security and Responsible Disclosure

This result covers the **full 64 rounds** of SHA-256's compression function --- there is no round reduction. While 5 of the 48 non-trivial schedule equations are relaxed, the attack operates on the complete SHA-256 round structure with the standard H0 initial value.

Several converging trends suggest that a full collision on SHA-256 is approaching feasibility:

1. **Steady sr progression**: From sr = 32 (Stevens et al., 2017) to sr = 56 (direct SAT, 2026) to sr = 59 (this work), the schedule compliance of known SFS collisions has been increasing steadily.
2. **Solver acceleration**: Our 3.3x speedup from domain-specific encoding is a first step. SHA-256-aware SAT solvers with algebraic propagation, differential transition tables, and theorem-guided search could yield further order-of-magnitude improvements.
3. **Scalable techniques**: Gap placement, the MSB kernel, and the da[56]=0 condition are all scalable and composable. The da[56]=0 condition converts a 256-bit search into a 32-bit exhaustive scan plus a bounded SAT problem.
4. **Diminishing slack**: At sr = 59, the solver operates with 64 bits of slack. At sr = 60 the slack reaches zero. Novel techniques could potentially bridge this gap.

We publish this work as **responsible disclosure**. While a full SHA-256 collision (sr = 64) has not yet been achieved, the tools and techniques presented here represent significant methodological advances that bring it closer. Organizations relying on SHA-256 for collision resistance should begin evaluating migration paths to SHA-3 or other post-quantum hash functions. The cryptographic community should treat the collision resistance of SHA-256 as having a finite and shrinking safety margin.

---

12. Reproduction

Science has a reproducibility crisis (more than half of papers have results that can't be reproduced) and we'd like our work to be an exception. The steps below should allow anyone to reproduce our work quickly. If you need any help with any aspect of this reproduction, please email the author and we will help you reproduce the results quickly.

12.1 Prerequisites

- GCC with OpenMP support (GCC >= 4.9)
- kissat SAT solver: `git clone https://github.com/arminbiere/kissat && cd kissat && ./configure && make`
- Python 3 (no pip dependencies required for the basic pipeline)
- A multi-core machine (recommended: 4+ cores for Phase 1)

12.2 Verifying the certificate

```
gcc -O3 -o cert_sr59 certificate_64r_sfs_sr59.c -lm && ./cert_sr59
```

Expected: sr = 59, collision verified, H = edc59a70 12bb36dd b8fbccc5 cd2e5445 5957ec4e d6c3c8e0 ff067757 8e7255c5.

12.3 End-to-end reproduction (basic pipeline, ~610 seconds)

Phase 1 --- find da[56] = 0 candidates (~180 seconds):

```
gcc -O3 -march=native -fopenmp -o scan_m0 scan_m0.c -lm
./scan_m0
```

Expected: 1-3 candidates including M[0] = 0x17149975.

Phase 2 --- gap-placed SAT tail (~430 seconds):

```
python3 sat_tail_sr59.py --m0 0x17149975 --kissat /path/to/kissat
```

Encodes full 64-round collision with gap placement. Produces verified sr = 59 collision.

12.4 End-to-end reproduction (accelerated pipeline, ~276 seconds)

Phase 1 (same as above, ~180 seconds).

Phase 2 --- reduced encoding + kissat (~96 seconds):

```
gcc -O3 -march=native -o sr59_cdcl sr59_cdcl.c -lm
./sr59_cdcl --kissat=/path/to/kissat
```

Precomputes 57 rounds natively, encodes 7-round tail (10K vars), pipes to kissat.

12.5 Files

File	Description
`certificate_64r_sfs_sr59.c`	Self-contained C certificate (compile and run to verify)
`scan_m0.c`	Phase 1: exhaustive M[0] scanner for da[56] = 0 candidates
`sat_tail_sr59.py`	Phase 2 (basic): full 64-round gap-placed SAT solver
`sr59_cdcl.c`	Phase 2 (accelerated): domain-specific reduced encoder + CDCL solver
`sr59_reduced.c`	Standalone reduced DIMACS encoder (for use with external solvers)

All source files are self-contained and require no external SHA-256 libraries.

13. Conclusion

We have presented a hybrid precomputation-SAT method that achieves semi-free-start collisions for the full 64-round SHA-256 compression function with 89.6% schedule compliance (sr = 59). The method exploits three key innovations: the MSB kernel (carry-free two-word difference), the da[56] = 0 condition (necessary and sufficient for the SAT tail), and gap placement (enforcing schedule equations beyond the free-variable boundary via the sigma_1 cascade). A domain-specific reduced encoder provides a 3.3x speedup, bringing the total attack time to under 5 minutes.

The schedule compliance metric we introduce provides a new lens for studying SHA-256's collision resistance. By parameterizing attacks along this axis, we reveal structural properties --- the sharp phase transitions at sr = 56, 57, 59, and 60 --- that are invisible to both reduced-round and free-schedule analyses. We believe this framework opens a genuinely new direction for studying collision attacks on SHA-2 family hash functions, and that the collision resistance of SHA-256 should be considered to have a finite and shrinking safety margin.

Acknowledgments

References

[1] Li, Y., Isobe, T., Meier, W., Zhang, Z. "New Records in Collision Attacks on SHA-2." EUROCRYPT 2024. Lecture Notes in Computer Science, vol. 14654. https://doi.org/10.1007/978-3-031-58716-0_7. IACR ePrint 2023/285.

[2] Zhang, Z., Guo, J., et al. "Collision Attacks on Round-Reduced SHA-256." EUROCRYPT 2026.

[3] Mendel, F., Nad, T., Schlaffer, M. "Improving Local Collisions: New Attacks on Reduced SHA-256." EUROCRYPT 2013. https://doi.org/10.1007/978-3-642-38348-9_17.

[4] Eichlseder, M., Mendel, F., Schlaffer, M. "Branching Heuristics in Differential Collision Search with Applications to SHA-512." FSE 2014.

[5] Nejati, S., Ganesh, V., Liang, J.H., Czarnecki, K. "Adaptive Restart and CEGAR-based Solver for Inverting Cryptographic Hash Functions." SAT 2020. https://doi.org/10.1007/978-3-030-51825-7_34.

[6] Stevens, M., Bursztein, E., et al. "The First Collision for Full SHA-1." CRYPTO 2017.

[7] Wang, X., Yin, Y.L., Yu, H. "Finding Collisions in the Full SHA-1." CRYPTO 2005. https://doi.org/10.1007/11535218_2.

[8] Wang, X., Yu, H. "How to Break MD5 and Other Hash Functions." EUROCRYPT 2005.

[9] National Institute of Standards and Technology. "Secure Hash Standard (SHS)." FIPS PUB 180-4, 2015. https://doi.org/10.6028/NIST.FIPS.180-4.

[10] Sanadhya, S.K., Sarkar, P. "Non-Linear Reduced Round Attacks Against SHA-256." ACISP 2008. https://doi.org/10.1007/978-3-540-89754-5_19.

[11] Li, Y., Isobe, T., Meier, W., Zhang, Z. "Improving the Differential Analysis of SHA-256." ASIACRYPT 2024 (Best Paper). https://doi.org/10.1007/978-981-96-0935-2_4.

[12] Sun, S., Hu, L., et al. "Automatic Security Evaluation and (Related-Key) Differential Characteristic Search." ASIACRYPT 2014.

[13] Chabaud, F., Joux, A. "Differential Collisions in SHA-0." CRYPTO 1998. https://doi.org/10.1007/BFb0055720.

[14] Ignatiev, A., Morgado, A., Marques-Silva, J. "PySAT: A Python Toolkit for Prototyping with SAT Oracles." SAT 2018.

[15] Biere, A., Fazekas, K., Fleury, M., Heisinger, M. "CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling Entering the SAT Competition 2020." SAT Competition 2020.

[16] Biere, A., Fleury, M. "Gimsatul, IsaSAT and Kissat Entering the SAT Competition 2022." SAT Competition 2022.

[17] Alamgir, M., Eichlseder, M., Leurent, G., Mendel, F. "Improved SAT-based Collision Attack on SHA-256." EUROCRYPT 2024. https://eprint.iacr.org/2024/2017.