Conditional-Affine Redundant Clauses for SHA-256 Differential SAT

Jiqiang Feng[1] and Kun Gao[2]

[1] Innora AI Security Research Lab; Innora.ai
[2] IoT Security Products Department; China Mobile IoT Company Limited; Chongqing, China

Abstract

Standard Tseitin encodings of the SHA-256 nonlinear functions Ch and Maj can hide conditioned differential projections from Boolean Constraint Propagation (BCP). We materialize them as short, semantically redundant CNF clauses. A cofactor theorem characterizes all controlled differential linear forms; its implemented unit-vector specialization returns exactly all minimum-control projections, yielding four Ch and twelve Maj clauses per bit. The clauses preserve models, introduce no variables, and strictly strengthen BCP on an explicit gate fragment. We claim neither propagation completeness nor a general affine compiler.

We evaluate mechanism separately from performance and distinguish the production bundle from the proposed layer. Frozen studies show a fixed-formula bundle benefit, but the matched clause isolation fails its effect gate and an unrestricted control is inconclusive. We therefore show neither a solver-independent speedup nor a new cryptanalytic attack. The restricted weight-15 C15 census passes the CaDiCaL rule but not the Kissat rule. In the stratified C16 extension, the frozen decisions are *too censored* for CaDiCaL and *too censored* for Kissat; these solver-stratified labels are not pooled. A pinned three-solver replay validates larger-bundle execution but cannot attribute performance to the clauses. A variable-preserving probe exposes all 32 tested implications only after augmentation.

Keywords: SHA-256, differential cryptanalysis, SAT solving, BCP, CNF encoding, CDCL, redundant clauses

1 Introduction

Despite two decades of SAT-based hash cryptanalysis, local implications in a differential CNF can remain invisible to unit propagation. Recent SHA-256 collision results improve differential-trail search and large-scale solving [LLW24, ZLGW26], including a practical 35-step collision [LLWS26] and, in a still newer preprint, the first 38-step collision attacks on SHA-256 and SHA-512 and the first effective 39-step attack on SHA-512 [LLQZ26]. Our goal is different: we study one representation-level bottleneck in subsequent SAT instances under explicitly bounded message-difference restrictions, including MILP-derived support upper bounds and combinatorially fixed supports. The reduced 18–20 step setting is an instrumented testbed for that encoding question, not a claim to improve the cryptanalytic step record.

The bottleneck occurs in standard encodings of Ch and Maj. We introduce **conditional-affine differential clauses**: static redundant constraints that state the output-difference projection already implied when control values and their differences are fixed. An earlier

Emails: feng@innora.ai (Jiqiang Feng), gaokun@cmiot.chinamobile.com (Kun Gao)

Date of this document: 2026-07-23.

version of this construction was termed degenerate conditional clauses. They shorten a local XOR-equivalence chain for BCP. The construction is not a substitute for Gaussian elimination. Our artifact supplies ordinary DIMACS after expanding XOR constraints, and its CryptoMiniSat replay records neither recovered-XOR nor matrix-activation statistics. That run therefore does not test complementarity with active global XOR reasoning.

Unlike programmatic SAT approaches [ANB24, FNP+23], the proposed encoding requires only standard CNF input. This makes the format portable across off-the-shelf solvers, although the magnitude and even the direction of a performance change remain solver- and instance-dependent.

Our central question is therefore narrow and falsifiable: can explicitly materializing these local implications reduce search on the measured differential instances beyond what the solver's normal preprocessing and inprocessing already expose?

Contributions

- We formalize a local BCP blind spot in the differential Tseitin encodings of Ch and Maj, prove semantic redundancy of the added clauses, and exhibit a gate-fragment partial assignment for which they strictly strengthen unit propagation. In established SAT terminology, the witnessed bridge is an empowering implicate; our contribution is its SHA-256 differential specialization, not the general notion.

- We characterize conditional differential linear forms exactly: a controlled output difference equals a fixed linear form if and only if the corresponding Boolean cofactor has that linear part. Projection identities are the unit-vector special case. We then give and implement an arity-parameterized truth-table procedure for minimum-control projections and prove that it returns exactly all valid identities at the smallest control size. Each emitted bridge is also an auxiliary-free two-clause minimum whose clauses are prime implicates of that bridge implication. Exhaustive verification recovers exactly the former four Ch and twelve Maj clauses per bit, including clause order.

- We separate base, minimal-rule, and full conditional-affine interventions, and report a prospectively gated evidence ladder. The fixed-formula bundle result passes; the corresponding clause isolation fails; an unrestricted control is inconclusive; C15 supports a CaDiCaL-specific result on its restricted weight-15 census; and the prospective weight-12–14 C16 extension is too censored for a directional decision under either tested solver. Seeds are nested within formulas, solver strata are not pooled, and every timeout remains in the capped endpoint.

- We provide a pinned, no-network artifact replay with explicit support, CNF, source, solver, container, and analysis identities. The fresh three-solver replay closes the former pairing-provenance defect while remaining explicitly non-isolating and therefore not evidence for clause-layer performance. Negative and failed-gate evidence is retained rather than replaced by completed-only or favorable historical summaries.

Propositions 2 and 3 formalize the cofactor characterization and minimum-control completeness claims; Theorems 2 and 3 separately establish model preservation for the proposed clause layer and projection equivalence for the evaluated production bundle.

2 Preliminaries

2.1 SHA-256 and Differential Cryptanalysis

SHA-256 operates on 32-bit words and uses the step function specified in FIPS 180-4 [Nat15]. The core nonlinear functions are:

$$\begin{aligned} \mathrm{Ch}(e,f,g) &= (e \wedge f) \oplus (\neg e \wedge g) \\ \mathrm{Maj}(a,b,c) &= (a \wedge b) \oplus (a \wedge c) \oplus (b \wedge c) \end{aligned}$$

The SHA-256 update at step t modifies the 8-word state (a,b,c,d,e,f,g,h) as follows:

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

where $\Sigma_0(x) = \mathrm{ROTR}^2(x) \oplus \mathrm{ROTR}^{13}(x) \oplus \mathrm{ROTR}^{22}(x)$ and $\Sigma_1(x) = \mathrm{ROTR}^6(x) \oplus \mathrm{ROTR}^{11}(x) \oplus \mathrm{ROTR}^{25}(x)$, with additions performed modulo 2^{32}.

We use *step* for one application of this update (often called a round). Every timed instance below is a reduced-step *semi-free-start* collision instance [ANB24]: the two computations share one unconstrained chaining value ($\Delta\mathrm{IV} = 0$), at least one message bit differs, and all eight final working-state differences are zero. The implementation's legacy configuration field `free_start=true` denotes this common free chaining value; it does not permit independently chosen starting values and therefore is not a general free-start or pseudo-collision instance. Because the omitted feed-forward adds the same chaining word to both equal final working words, equality immediately before feed-forward is equivalent to equality after feed-forward in this setting.

In differential cryptanalysis, we trace the propagation of bitwise XOR differences between a pair of messages (M, M'). Let $\Delta x = x \oplus x'$. The goal is to find valid values for x and x' such that a specific differential characteristic holds.

2.2 SAT Solving and BCP

Modern CDCL SAT solvers search for variable assignments that satisfy a CNF formula. BCP, or unit propagation, is the core deductive engine: if all but one literal in a clause evaluate to false, the remaining literal must be true. An encoding's efficiency heavily depends on its ability to trigger BCP early in the search tree.

2.3 Gaussian Elimination in SAT

Some SAT solvers, including CryptoMiniSat, can accept explicit XOR constraints or attempt to recover XOR structure from CNF and selectively maintain linear systems for Gauss–Jordan reasoning [SNC09, SGM20]. When enabled on a recovered or supplied system, this engine can propagate global linear implications alongside CDCL. Capability does not establish activation on a particular run; our artifact replay supplies ordinary DIMACS and retains no matrix-activation log.

3 The BCP Blind Spot and Conditional-Affine Clauses

3.1 BCP Blind Spot Analysis

Consider the Ch function in a differential setting with two instances:

$$y = \mathrm{Ch}(e,f,g), \qquad y' = \mathrm{Ch}(e',f',g')$$

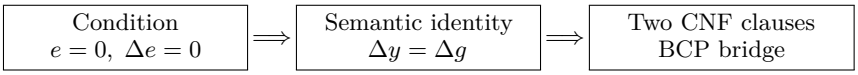


Figure 1: Mechanism for one Ch restriction. The base Tseitin encoding entails the middle identity but cannot propagate it under the condition alone (Proposition 1); the two redundant clauses expose the identity to BCP. The same synthesis is applied to the second Ch restriction and the six symmetric Maj restrictions.

The differential output is $\Delta y = y \oplus y'$. Suppose the solver decides $e = 0$ and $\Delta e = 0$ (which implies $e' = 0$). By Boolean algebra: $y = \mathrm{Ch}(0, f, g) = g$ and $y' = \mathrm{Ch}(0, f', g') = g'$. Consequently, $\Delta y = g \oplus g' = \Delta g$.

In a standard CNF encoding, y and y' are encoded separately using Tseitin transformations. The relation $\Delta y = y \oplus y'$ is encoded with 4 clauses. When $e = 0, e' = 0$, the conditioned Ch gate clauses reduce to binary equivalences $y \leftrightarrow g$ and $y' \leftrightarrow g'$, but they do not assign either side. To expose $\Delta y = \Delta g$, the solver must also compose the clauses encoding $y \oplus y' = \Delta y$ and $g \oplus g' = \Delta g$. Because BCP only triggers on *unit* clauses, it cannot infer any literal over $\{\Delta y, \Delta g\}$ until further variables are assigned. This constitutes the **BCP blind spot**: the logical equivalence $\Delta y = \Delta g$ is true, but the CDCL engine cannot "see" it without further branching. Figure 1 summarizes the condition, semantic identity, and added propagation bridge.

Proposition 1. *Let Φ be the artifact's gate-level Tseitin CNF encoding (in the sense of [Tse83]) of the joint constraint $y = \mathrm{Ch}(e, f, g) \wedge y' = \mathrm{Ch}(e', f', g')$ together with the differential definitions $\Delta x \leftrightarrow x \oplus x'$ for $x \in \{e, f, g, y\}$. Each* Ch *gate uses the standard four-clause ITE/MUX encoding and each binary XOR uses the standard four-clause encoding. Let $\rho = \{e \mapsto 0, \Delta e \mapsto 0\}$ be a partial assignment. Then the unit-propagation closure $\mathrm{UP}(\Phi \mid \rho)$ contains no literal over the variable set $\{\Delta y, \Delta g\}$.*

Proof. Under ρ, the differential clause for Δe (i.e., $\Delta e \leftrightarrow e \oplus e'$) becomes unit and forces $e' \mapsto 0$. The four ITE/MUX clauses for $y = \mathrm{Ch}(e, f, g)$ at $e = 0$ reduce to two binary clauses asserting $y \leftrightarrow g$; symmetrically the encoding for $y' = \mathrm{Ch}(e', f', g')$ reduces to $y' \leftrightarrow g'$. No further unit literals fire: the variables $\{g, g', y, y', \Delta y, \Delta g\}$ remain unassigned. In particular, the four Tseitin clauses encoding $\Delta y \leftrightarrow y \oplus y'$ each retain three unassigned literals, and the analogous clauses for Δg retain three unassigned literals. Since neither Δy nor Δg nor their negations appear as unit consequences, $\mathrm{UP}(\Phi \mid \rho)$ contains no literal over $\{\Delta y, \Delta g\}$. □

3.2 Conditional-Affine Clauses for Ch

To bypass this blind spot, we add redundant clauses enforcing conditional differential equivalence. For $\mathrm{Ch}(e, f, g)$, two control restrictions yield projection identities:

1. When $e = 0$ and $\Delta e = 0$: $\Delta y = \Delta g$.
2. When $e = 1$ and $\Delta e = 0$: $\Delta y = \Delta f$.

Case 1 translates to $(\neg e \wedge \neg \Delta e) \implies (\Delta y \leftrightarrow \Delta g)$:

$$e \vee \Delta e \vee \neg \Delta y \vee \Delta g, \qquad e \vee \Delta e \vee \Delta y \vee \neg \Delta g$$

Case 2: $(e \wedge \neg \Delta e) \implies (\Delta y \leftrightarrow \Delta f)$:

$$\neg e \vee \Delta e \vee \neg \Delta y \vee \Delta f, \qquad \neg e \vee \Delta e \vee \Delta y \vee \neg \Delta f$$

This yields exactly **4 clauses per bit** for Ch.

3.3 Conditional-Affine Clauses for Maj

For $\mathrm{Maj}(a, b, c)$, the logic collapses when two inputs are unequal but have no difference. If $a \neq b$ and $\Delta a = 0, \Delta b = 0$, then a, b take opposite values in both instances. The tie-breaker is c, yielding $\Delta\mathrm{Maj} = \Delta c$. The condition $(a \neq b) \wedge \neg\Delta a \wedge \neg\Delta b$ expands to:

$$a \vee \neg b \vee \Delta a \vee \Delta b \vee \neg\Delta\mathrm{Maj} \vee \Delta c$$
$$a \vee \neg b \vee \Delta a \vee \Delta b \vee \Delta\mathrm{Maj} \vee \neg\Delta c$$
$$\neg a \vee b \vee \Delta a \vee \Delta b \vee \neg\Delta\mathrm{Maj} \vee \Delta c$$
$$\neg a \vee b \vee \Delta a \vee \Delta b \vee \Delta\mathrm{Maj} \vee \neg\Delta c$$

By symmetry over all three pairs, this generates $4 \times 3 =$ **12 clauses per bit** for Maj.

3.4 Formal Correctness

The artifact's base encoding represents each Ch output with the standard four-clause ITE/MUX encoding, each Maj output with the standard six-clause majority encoding, and every difference bit with a Tseitin XOR relation. The results below refer to this concrete encoding; equivalent gate decompositions need not have the same BCP closure.

Theorem 1. ***(Strict local BCP strengthening).*** *Let Ψ_{Ch} be the local two-copy* Ch *and XOR fragment of Proposition 1, and let $\mathcal{C}_{\mathrm{Ch}}$ contain the two bridge clauses for $e = 0, \Delta e = 0$. For the partial assignment $\rho^* = \{e \mapsto 0, \Delta e \mapsto 0, \Delta g \mapsto 0\}$, the unit-propagation closure $\mathrm{UP}(\Psi_{\mathrm{Ch}} \cup \mathcal{C}_{\mathrm{Ch}} \mid \rho^*)$ strictly contains $\mathrm{UP}(\Psi_{\mathrm{Ch}} \mid \rho^*)$.*

Proof. Under ρ^*, the XOR definition for Δg reduces to the two binary clauses $g \leftrightarrow g'$. As in Proposition 1, the conditioned Ch clauses reduce to $y \leftrightarrow g$ and $y' \leftrightarrow g'$, while the clauses for $\Delta y \leftrightarrow y \oplus y'$ retain three unassigned literals. These binary equivalences assign none of g, g', y, y', so no clause in Ψ_{Ch} becomes unit on Δy; hence $\mathrm{UP}(\Psi_{\mathrm{Ch}} \mid \rho^*)$ contains neither polarity of Δy. In the augmented fragment, the clause $e \vee \Delta e \vee \neg\Delta y \vee \Delta g$ reduces under ρ^* to the unit clause $\neg\Delta y$. Therefore $\mathrm{UP}(\Psi_{\mathrm{Ch}} \cup \mathcal{C}_{\mathrm{Ch}} \mid \rho^*)$ contains $\neg\Delta y$ and the closure is strictly larger. This is a local witness; additional constraints in a complete instance may already derive the same literal, so the theorem does not claim strictness under every global context. □

The artifact verifier exhaustively enumerates all 2^{12} assignments of this local fragment: the base and augmented formulas have the same 64 models, while explicit unit-closure simulation confirms the theorem's strict witness.

3.5 Minimum-Control Projection Synthesis and Semantic Redundancy

Algorithm 1 states the procedure implemented in the artifact. It searches for conditional *projection identities*: under fixed control values whose differences are zero, the output difference equals one residual input-difference bit. It does not enumerate arbitrary multi-variable affine consequents.

Proposition 2. ***(Cofactor characterization of differential linear forms).*** *Let $F : \{0,1\}^n \to \{0,1\}$, let $S \subseteq \{1, \ldots, n\}$ be a set of control indices fixed to $\sigma \in \{0,1\}^S$, and let $\mathbf{a}$ be a binary coefficient vector on the residual indices $R = \{1, \ldots, n\} \setminus S$. Write g for the cofactor of F obtained by substituting $x_S = \sigma$, viewed as a Boolean function of the residual vector $\mathbf{z}$. Then*

$$F(\mathbf{x}) \oplus F(\mathbf{x} \oplus \Delta\mathbf{x}) = \langle \mathbf{a}, \Delta\mathbf{x}_R \rangle$$

for every $\mathbf{x}, \Delta\mathbf{x}$ satisfying $x_S = \sigma$ and $\Delta x_S = 0$ if and only if there is a constant $b \in \{0,1\}$ such that $g(\mathbf{z}) = \langle \mathbf{a}, \mathbf{z} \rangle \oplus b$ for every residual assignment $\mathbf{z}$. A projection onto Δx_t is the special case $\mathbf{a} = \mathbf{e}_t$.

Proof. If $g(\mathbf{z}) = \langle \mathbf{a}, \mathbf{z} \rangle \oplus b$, linearity gives $g(\mathbf{z}) \oplus g(\mathbf{z} \oplus \boldsymbol{\delta}) = \langle \mathbf{a}, \boldsymbol{\delta} \rangle$, proving the forward differential identity. Conversely, fix one residual assignment $\mathbf{z}_0$. For any $\mathbf{z}$, apply the assumed identity with $\boldsymbol{\delta} = \mathbf{z} \oplus \mathbf{z}_0$ to obtain

$$g(\mathbf{z}) \oplus g(\mathbf{z}_0) = \langle \mathbf{a}, \mathbf{z} \rangle \oplus \langle \mathbf{a}, \mathbf{z}_0 \rangle.$$

Thus $g(\mathbf{z}) \oplus \langle \mathbf{a}, \mathbf{z} \rangle$ is constant over all $\mathbf{z}$. Taking that constant as b gives the claimed cofactor form. □

Proposition 2 characterizes the complete family of controlled linear difference forms. The implemented grammar deliberately instantiates only unit vectors, so every evaluated clause remains a projection bridge rather than evidence for a general affine-consequence compiler. The theorem nevertheless supplies an independent certificate for every identity returned by the pairwise differential enumeration below.

Algorithm 1 Minimum-Control Differential Projection Synthesis

Require: Truth table of $F : \{0,1\}^n \to \{0,1\}$.
Ensure: Every valid identity at the smallest control size and its clause templates.

for $k = 0, 1, \ldots, n-1$ **do**
  $I_k \leftarrow \emptyset$
  for each target $t \in \{1, \ldots, n\}$ **do**
    for each $S \subseteq \{1, \ldots, n\} \setminus \{t\}$ with $|S| = k$ **do**
      for each $\sigma : S \to \{0,1\}$ **do**
        if $F(\mathbf{x}) \oplus F(\mathbf{x} \oplus \Delta\mathbf{x}) = \Delta x_t$ for all $\mathbf{x}, \Delta\mathbf{x}$ satisfying $x_i = \sigma(i)$ and $\Delta x_i = 0$ for $i \in S$ **then**
          Add (t, S, σ) to I_k
        end if
      end for
    end for
  end for
  if $I_k \neq \emptyset$ **then**
    Encode each antecedent $\Rightarrow (\Delta y \leftrightarrow \Delta x_t)$ as two CNF clauses
    return I_k and the resulting clause templates
  end if
end for
return $\emptyset$

Proposition 3. ***(Minimum-control completeness of Algorithm 1).*** *Suppose the algorithm first finds a nonempty identity set at control size k^*. Its returned identities are exactly all valid projection identities with k^* controls, and no valid projection identity exists with fewer than k^* controls.*

Proof. For each fixed k, the nested loops enumerate every target, every k-element control set disjoint from that target, and every Boolean assignment to the control set. The acceptance test ranges over every base input and input-difference assignment satisfying the antecedent, so a candidate is accepted if and only if it is a valid projection identity. Thus I_k is exactly the valid identity set at size k. The outer loop examines sizes in increasing order and returns at the first nonempty I_k, proving both assertions. □

Proposition 4. ***(Auxiliary-free clause minimality of one bridge).*** *Let $A = \bigwedge_{i=1}^{m} a_i$, where the a_i are literals on distinct variables and u, v are distinct variables absent from A. The auxiliary-free CNF*

$$\left(\bigvee_{i=1}^{m} \neg a_i \vee \neg u \vee v \right) \wedge \left(\bigvee_{i=1}^{m} \neg a_i \vee u \vee \neg v \right)$$

is equivalent to $A \Rightarrow (u \leftrightarrow v)$. Both clauses are prime implicates of this bridge implication, and no equivalent auxiliary-free CNF over these variables contains fewer than two clauses. Each clause has width $m + 2$.

Proof. The displayed clauses are the direct CNF of the implication. To see primality, first remove an antecedent literal $\neg a_j$ from either clause: set a_j false, every other a_i true, and choose the unequal values of u, v that falsify the remaining target literals. The implication is then true because A is false, while the shortened clause is false. If instead one of the two target literals is removed, set all a_i true and choose $u = v$ so that the remaining target literal is false; again the implication is true and the shortened clause is false.

Zero clauses cannot represent the non-tautological implication. If one clause represented it, that clause would have to be false under both forbidden assignments $A = 1, u = 1, v = 0$ and $A = 1, u = 0, v = 1$. No literal on u or v is false in both assignments, and every antecedent literal a_i is true in both, so such a clause can contain only their complements $\neg a_i$. It would then also be false when $A = 1$ and $u = v$, excluding a satisfying assignment. Hence at least two clauses are necessary, and the displayed pair is clause-minimal. □

For $F = \mathrm{Ch}(e, f, g)$, the minimum control size is one and the procedure returns exactly $e = 0 \Rightarrow \Delta y = \Delta g$ and $e = 1 \Rightarrow \Delta y = \Delta f$, with $\Delta e = 0$ in both antecedents. For $F = \mathrm{Maj}(a, b, c)$, the minimum is two: every pair of fixed, opposite inputs projects the output difference onto the remaining input, giving six identities. Two implication clauses per identity yield four Ch and twelve Maj clauses per bit.

Each fixed control contributes two antecedent literals—its base value and its zero-difference condition—so Proposition 4 gives width $2k + 2$: four for the one-control Ch identities and six for the two-control Maj identities. This is clause minimality and prime-implicate status relative to each individual auxiliary-free bridge implication. It is not a global minimum-size encoding claim, nor a claim that the clauses are prime implicates of the complete SHA-256 CNF, where surrounding constraints may entail shorter clauses.

The search is exponential in n: the implementation enumerates control sets, control assignments, and the remaining pairs $(\mathbf{x}, \Delta\mathbf{x})$. If all control sizes are exhausted, the number of tested input/difference pairs is at most $n \sum_{k=0}^{n-1} \binom{n-1}{k} 2^k 4^{n-k} = 4n6^{n-1}$, or $O(n6^n)$ truth-table evaluations up to the cost of evaluating F. Proposition 2 gives a cheaper equivalent certificate: testing the 2^{n-k} values of each controlled cofactor reduces the corresponding all-context projection bound to $n2^{2n-1} = O(n4^n)$. We retain the direct pairwise implementation as an auditable oracle for the constant-arity kernels rather than claim scalability to larger S-boxes. The implementation in `conditional_affine.py` is arity-parameterized; the shipped verifier exhaustively validates only the $n = 3$ Ch/Maj truth tables. The independent script `experiments/verify_projection_cofactor_characterization.py` checks the general linear-form equivalence for all Boolean functions of arity one through three over 14,536 function/control/form tuples, separately checks 7,012 projection contexts, confirms that no smaller-control projection exists, and recovers exactly the runtime identity sets. The generator also preserves the former templates' clause and literal order, so the evaluated CNF definition is unchanged.

Theorem 2. ***(Soundness of Algorithm 1).*** *$\Phi_{base} \models \mathcal{C}_{degen}$: every model of the base Tseitin CNF encoding satisfies all generated conditional-affine clauses. Because this clause-generation layer introduces no variables, Φ_{base} and $\Phi_{base} \wedge \mathcal{C}_{degen}$ have exactly the same models. This statement concerns the proposed layer, not the implementation's larger production bundle, whose carry-difference infrastructure introduces auxiliary variables.*

Proof. For each accepted identity, exhaustive testing establishes that the antecedent implies $\Delta y = \Delta x_t$ for every remaining assignment. The two emitted clauses are exactly the CNF of that implication. Since Φ_{base} encodes the two evaluations of F and their differential definitions, every model of Φ_{base} satisfies the clauses. □

The performance experiments use a larger production bundle, so its auxiliary-variable semantics require a separate statement. The implementation allocates numeric DIMACS identifiers incrementally; inserting carry-difference variables therefore shifts later identifiers and also changes pool-derived temporary names. We compare formulas by the canonical bijection α that preserves construction-order wire and gate roles after deleting those interleaved carry-difference variables, rather than by raw identifiers or temporary names. Let V_{base} denote the variables of the unaugmented encoding, let $V_b = \alpha(V_{base})$ be their corresponding variables in the bundle, and let V_c contain all carry-difference auxiliaries introduced by the minimal-rule infrastructure.

Theorem 3. ***(Projection soundness of the production bundle).*** *If Φ_{bundle} is the full* `degenerate` *encoding, then*

$$\{N|_{V_b} : N \models \Phi_{bundle}\} = \{\alpha(M) : M \models \Phi_{base}\}.$$

Moreover, after applying α, every base model has a unique extension to V_c.

Proof. For every modular-addition bit, the bundle defines the auxiliary carry difference by $\Delta c_i = c_i \oplus c'_i$ using the exact four-clause XOR encoding, so its value is unique. If $\Delta a_i = \Delta b_i = \Delta c_{i+1} = 0$, the two full adders receive identical input triples and therefore have identical sum and carry outputs; hence the added implications $\Delta s_i = 0$ and $\Delta c_i = 0$ are entailed. Likewise, if all inputs to a deterministic Ch or Maj gate have zero difference, its output difference is zero, so the generic difference-only lemma is entailed. The remaining added clauses are $\mathcal{C}_{\text{degen}}$, which are entailed by Theorem 2. Thus every base model has exactly one satisfying assignment to the definitional auxiliaries and satisfies every added implication, while restriction of any bundle model satisfies the unchanged base clauses. □

The artifact verifier `experiments/verify_bundle_projection_soundness.py` independently exercises Theorem 3: it exhaustively checks all 64 pairs of one-bit gate inputs, all 64 pairs of one-bit full-adder inputs, and uniqueness of the carry-XOR extension. Its two-step base-to-minimal structural check deletes exactly 462 carry-difference variables, restores the same 5,532 non-carry allocation roles and 2,624 XOR constraints, and confirms that all 7,389 base clauses survive under the construction-order renaming. The 128 remaining non-carry additions are precisely four-literal generic gate lemmas; carry-containing additions have the expected 14 unit, 1,848 ternary, and 896 quaternary clause shape. A separate minimal-to-full check confirms identical variable names and XOR constraints, no removed minimal-rule clause, and exactly 1,024 added conditional-affine clauses (256 of length four and 768 of length six).

An orthogonal semantic gate, `experiments/verify_diff_path_encoder_semantics.py`, solves a nontrivial two-step collision fixture in the base, minimal-rule, and full modes, evaluates every emitted CNF clause and XOR constraint on each returned model, and checks the final working states after the configured steps against an executable SHA-256 oracle. It separately checks the first expanded message word W_{16} for unequal message pairs in all three modes; the oracle itself matches four `hashlib` vectors. These deterministic fixtures strengthen implementation assurance but are not an exhaustive proof of the complete 20-step encoding.

Semantic redundancy is the correctness condition needed for model preservation. It is weaker than claiming that a production solver will discover the same clause through vivification, probing, or another inprocessing rule. SAT preprocessors and proof systems provide stronger checkable redundancy notions [EB05, JHB12, HJW14]; our argument does not depend on a particular schedule deriving these clauses.

Propagation boundary. In the terminology of Bordeaux and Marques-Silva [BMS12], the first bridge clause used in Theorem 1 is an *empowering implicate* of the local fragment: it

is entailed, while under ρ^* it unit-propagates $\neg\Delta y$ and the base fragment does not. This places the mechanism inside the established knowledge-compilation framework; it is not a new general class of SAT consequence. Deciding and compiling empowering implicates in arbitrary CNF is hard in general [BBČ+13]. Algorithm 1 avoids that general problem by exhaustive enumeration of constant-arity Ch/Maj kernels under a restricted projection grammar.

We do not claim propagation completeness of $\Phi_{base} \cup \mathcal{C}_{\text{degen}}$. Theorem 1 is local and existential—it exhibits one gate-fragment partial assignment for which BCP is strictly stronger—not a universal statement over full-instance contexts, all partial assignments, or all implied literals.

Scope beyond SHA-256. SHA-1 reuses Ch and Maj, so the same verified templates can be instantiated there. Extending the synthesizer to multi-variable affine consequents, and evaluating conditional restrictions in SHA-1, Keccak-χ, or SM3, remain future work rather than claims of this paper [ZHL25, LLL+25].

Empirical attribution boundary. Theorem 2 establishes semantic redundancy but does not imply a runtime improvement. We therefore never pool base-versus-production-bundle results with minimal-versus-full clause-isolation results. C12 and the artifact replay evaluate the larger bundle. C13–C16 hold the minimal-rule infrastructure fixed and add only $\mathcal{C}_{\text{degen}}$; these studies determine the clause-specific performance conclusion.

3.6 Design Exclusion: Σ_0/Σ_1

Σ_0/Σ_1 are purely linear combinations of rotations and XORs. Their output differences are deterministic functions of the input differences, so they do not exhibit the control-conditioned collapse targeted by Algorithm 1. We therefore exclude them from the proposed clause family rather than infer a performance effect without a separately controlled experiment.

4 Related Work

CNF encoding and synthesis. Prestwich [Pre09] surveys how semantically correct CNF encodings can differ substantially in size, propagation, and interaction with search. Bordeaux and Marques-Silva [BMS12] compile formulas toward propagation completeness by adding empowering implicates; Babka et al. [BBČ+13] characterize the complexity of detecting such clauses and of propagation completeness. Gwynne and Kullmann [GK14] formalize arc-consistent and propagation-complete CNF representations for XOR systems, making explicit that semantic correctness alone does not determine unit-propagation strength. We do not construct such a globally propagation-complete representation: we materialize selected domain-specific consequences and prove only the local strict witness of Theorem 1. Inala et al. [ISSL16] use program synthesis and autotuning to generate domain-specific bit-vector-to-CNF encoders. Algorithm 1 solves a narrower problem: it neither replaces the base Tseitin encoding nor optimizes a whole bit-vector encoder, but exhaustively derives entailed conditional projection bridges for a fixed Boolean kernel and minimizes the number of conditioned controls rather than the number of clauses or auxiliary variables.

Equi-propagation and redundant-clause learning. Li [Li03] identified unit propagation's weakness on equivalency clauses and added solver-side equivalent-literal propagation; Heule and van Maaren [HvM05] subsequently aligned equivalence reasoning with CNF reasoning on parity-rich formulas. Metodi et al. [MCS13] model small constraint fragments as Boolean functions, detect equalities between literals by equi-propagation,

and use those equalities to partially evaluate and simplify a generated encoding. Reeves et al. [RHB23] instead learn short propagation-redundant clauses from a formula-specific positive reduct before invoking an off-the-shelf solver. Thus both the generic equivalence-propagation blind spot and the idea of adding local equivalences or redundant clauses predate this work. Our narrower contribution is the minimum-control SHA-256 differential specialization: we leave the base encoding intact, enumerate all smallest-control projection identities of the repeated Ch/Maj truth tables, emit logically entailed auxiliary-free bridges, and prove two-clause minimality and prime-implicate status relative to each bridge implication. We do not claim a general-purpose equivalence-reasoning solver, equi-propagation compiler, or propagation-redundant-clause learner.

SAT-based cryptanalysis. Mironov and Zhang [MZ06] used a minimally modified off-the-shelf SAT solver to generate full MD4 and MD5 collisions from a supplied differential path, and reported this as the first SAT-solver-aided cryptanalysis of a nontrivial primitive. Nossum [Nos12] studied SAT preimage search on reduced-difficulty SHA-1 and introduced a 32-bit modular-addition encoding that significantly reduced solving time relative to the previously known encodings tested in that thesis. Stevens [Ste13] developed optimal joint local-collision analysis and reported an implemented open-source SHA-1 near-collision attack costing the equivalent of $2^{57.5}$ compression calls, together with identical- and chosen-prefix collision attacks estimated at approximately 2^{61} and $2^{77.1}$ calls. More recently, Nejati and Ganesh [NG19] proposed CDCL(Crypto), extending propagation and conflict analysis inside CDCL with primitive-specific code. This solver-extended setting is more expressive than pure CNF input, while our clauses target the more drop-in CNF-only path.

Gaussian elimination in SAT. Soos et al. [SNC09, SGM20] introduced CryptoMiniSat, which can handle XOR clauses via Gauss–Jordan elimination integrated into CDCL search. Modern XOR engines use dedicated watch schemes and selectively applied matrix reduction for formula-wide linear reasoning. Our construction instead exposes bounded local implications as ordinary 4- and 6-literal CNF clauses. The artifact exports ordinary DIMACS after expanding XOR constraints; its CryptoMiniSat row records neither recovered-XOR nor matrix-activation statistics. That bundle-level row therefore neither proves active Gaussian reasoning nor establishes complementarity with the proposed clauses.

MILP-based differential search. Mouha et al. [MWGP12] introduced a MILP method for proving differential and linear security bounds by lower-bounding active S-box counts in word-oriented ciphers. Sun et al. [SHW+14] extended MILP-based evaluation and (related-key) differential-characteristic search to bit-oriented ciphers at ASIACRYPT 2014. In a separate line specific to SHA-256, Mendel et al. [MNS11] introduced the first automated tool for complex SHA-2 differential characteristics and coupled characteristic search with the search for conforming message pairs; their later work [MNS13] improved that custom search strategy and reported a practical 28-step collision. Those papers use custom characteristic and conforming-pair search rather than the support-restricted SAT stage evaluated here. Zhang et al. [ZLGW26] reached 37 steps using improved trail search; Li et al. [LLWS26] subsequently reported the first practical 35-step SHA-256 collision using memory-efficient SAT-driven methods, with theoretical collision attacks on SHA-256 and SHA-512 reaching 36–37 steps. A still newer preprint by Li et al. [LLQZ26] reports the first 38-step collision attacks on SHA-256 and SHA-512 with time complexities $2^{104.3}$ and $2^{125.4}$, respectively, and the first effective collision attack on 39-step SHA-512; the same method does not yield an effective 39-step attack on SHA-256. Li et al. [LLW24] achieved earlier new collision records on SHA-2 at EUROCRYPT 2024. These advances change the cryptanalytic frontier but do not study the minimum-control projection clauses evaluated here. Taka et al. [TSI+25] (ToSC 2025) introduce a divide-and-conquer SAT framework that partitions a characteristic-search SAT model into smaller UNSAT-provable sub-problems. Their work targets characteristic search, while ours targets solution search

Table 1: Encoding interventions. Only the minimal-versus-full contrast identifies the marginal contribution of the proposed clauses.

Name	Implementation mode	Added structure relative to base
Base	`False`	none
Minimal	`True`	difference lemmas and carry-difference infrastructure
Full	`degenerate`	minimal plus $\mathcal{C}_{\text{degen}}$

after a message-word support has been fixed. Our exact-support formulas constrain only which message words have nonzero difference; they are not sampled complete differential characteristics.

Programmatic SAT and IPASIR-UP. Fazekas et al. [FNP+23] proposed the IPASIR-UP interface for user propagators in CDCL solvers. Alamgir et al. [ANB24] applied this to SHA-256 collision search, using custom inconsistency blocking plus bitsliced and modular-addition propagation to guide CaDiCaL at runtime. While effective, their approach requires IPASIR-UP support and custom callback code. Our static clauses require no solver modification and are conceptually complementary; whether combining the two improves solving is untested here.

SAT solver heuristics. Audemard and Simon [AS09] introduced Glucose's LBD-based clause management. Biere et al. [BFFH20, BF22] developed CaDiCaL and Kissat with modular CDCL implementations. Our formula-level study stratifies CaDiCaL and Kissat rather than treating seeds as cross-solver replicates; the different decisions are evidence of solver interaction, not a heuristic ranking. CryptoMiniSat appears only in the one-shot production-bundle artifact replay.

Preprocessing and proof checking. Classical SAT preprocessing removes redundancy through variable and clause elimination [EB05]; later inprocessing frameworks apply such rules during search [JHB12], and tools such as Coprocessor expose many simplification techniques in modular form [Man12]. Proof checkers such as DRAT-trim [HJW14] provide an orthogonal audit path for clausal derivations. We cite these lines to position our semantic-redundancy argument. Section 5.6 tests one variable-preserving CaDiCaL preprocessing schedule; it is not a universal statement about preprocessors or proof systems.

5 Experimental Evaluation

5.1 Design, Interventions, and Units of Analysis

The evaluation uses reduced-step semi-free-start SHA-256 collision formulas and two support semantics. Legacy upper-bound mode fixes words outside the support to zero; selected words remain bitwise free and may still be zero. Exact-support mode, used by the 30-support stress test and C15/C16, instead adds to every selected word a 32-literal disjunction that requires at least one nonzero difference bit, so the active-word set equals the support. Both define message-difference search regions, not complete differential characteristics. The SAT encoding fixes the initial chaining values equal and requires equal final states after the configured steps.

The implementation exposes the three interventions in Table 1. This distinction is essential: base-versus-bundle experiments test a joint production intervention, whereas minimal-versus-full experiments isolate the proposed conditional-affine clauses by holding all pre-existing difference and carry infrastructure fixed.

Table 2 gives the predeclared evidence hierarchy. Here, *predeclared* means captured in result-free local protocol artifacts before the corresponding solver outputs and later

hash-bound to the raw inputs; no third-party registry or external timestamp is claimed. Solver seeds are repeated measurements nested inside one formula; they are never treated as independent formulas. Formula-level effects are analyzed separately by solver because a clause set can interact with solver heuristics. All intervals are paired percentile bootstrap intervals with a fixed seed, except C15 and C16, whose intervals resample formula-level means and are explicitly stability diagnostics rather than design-based confidence intervals for a support population.

Table 2: Frozen confirmatory protocols. "Bundle" denotes base versus full; "isolated" denotes minimal versus full.

Study	Formula unit	Repeats	Contrast	Cap
C12	one fixed 20-step formula	10 seeds	bundle	3,600 s
C13	same fixed 20-step formula	10 seeds	isolated	3,600 s
C14	one unrestricted 18-step formula	30 seeds	isolated	300 s
C15	15 exact 18-step supports	2 seeds/support	isolated, per solver	300 s
C16	30 stratified exact 18-step supports	2 seeds/support	isolated, per solver	300 s

Throughout C15 and C16, a formula is *informative* when at least one of its two nested-seed pairs is not a double timeout. This flag gates completeness but never filters the capped endpoint: every formula and both seeds remain in all reported means and intervals.

C15 uses all 15 non-pilot supports of weight 15 remaining in its frozen 16-support frame. Its endpoint for each formula and solver is the mean over the two nested seeds of $(t^{\mathrm{cap}}_{\mathrm{minimal}} - t^{\mathrm{cap}}_{\mathrm{full}})/300$, where $t^{\mathrm{cap}} = \min(t, 300\,\mathrm{s})$; positive values favor the proposed clauses. The predeclared benefit rule requires at least ten informative formulas, a mean above 0.05, and a formula-resampling interval wholly above zero. The two solvers receive separate decisions; there is no pooled cross-solver endpoint.

C16 retains the capped, solver-stratified estimand on a deterministic, outcome-blind hash-ranked frame of 30 previously unused supports: ten formulas each at exact weights 12, 13, and 14. For each weight w, the eligible frame contains all $\binom{16}{w}$ exact word supports after excluding every support already present under the frozen paths tree, leaving 1,788, 557, and 117 candidates, respectively. The ten lowest SHA-256 ranks of the domain-separated tuple (public seed, 18 steps, weight, ordered support) are retained in each stratum. The manifest records the seed, exclusion-source digests, ranks, and selected-path digests; an independent verifier re-enumerates the three candidate sets and exact rank order without reading solver results. Each formula has two nested seeds; both intervention orders are counterbalanced separately within each solver. The primary endpoint equally weights the three stratum means, preventing unequal eligible-stratum sizes from silently reweighting the estimand. The frozen decision first requires at least 20 informative formulas per solver and five in every stratum. Benefit then requires primary mean > 0.05, bootstrap lower bound > 0, and every stratum mean ≥ -0.05; harm is symmetric. Near-neutral requires the bootstrap interval to lie within $[-0.05, 0.05]$; otherwise the decision is inconclusive. The bootstrap and finite-population-corrected intervals are stability diagnostics, not design-based coverage guarantees for either the eligible supports or the full SHA-256 formula population. Only the stratified bootstrap interval enters the frozen decision; the finite-population correction is secondary artifact diagnostics and cannot upgrade or repair a solver label.

This endpoint is a prespecified loss under a 300s resource budget, not an estimator of uncensored mean runtime. Capped runtime is itself an explicit objective in algorithm-configuration work [WGS19]; when the target is instead the latent runtime distribution, timeouts are right-censored observations and require a model such as survival analy-

sis [TWW+20]. We use capped loss because it is defined for every paired run without imputing post-cap runtimes. Consequently, a double timeout contributes zero observed capped difference, not evidence of equal latent runtimes. The endpoint also differs from the PAR-2 competition score, which charges an unsolved instance twice the timeout [SAT24].

Host and container load are sampled at nominal 60-second intervals as a diagnostic; no timing row is discarded or reweighted based on observed load.

5.2 Encoding Size Profile

Table 3: In-memory encoding sizes before solver-specific export. Ordinary CNF clauses exclude the separately counted XOR constraints, which are expanded when writing DIMACS for CaDiCaL/Kissat.

Instance	Variables	Ordinary clauses	XOR constraints
18-Step Baseline	32,790	64,791	18,432
18-Step + Bundle	37,146	101,163	18,432
20-Step Baseline	37,452	74,061	21,248
20-Step + Bundle	42,468	115,525	21,248

The conditional-affine layer itself introduces no variables and contributes exactly $512n$ ordinary CNF clauses for n steps (9,216 at 18 steps and 10,240 at 20 steps). The larger table deltas include the minimal-rule infrastructure: at 18 steps, 4,356 carry-difference variables and 27,156 associated definition, unit, and propagation clauses, plus 9,216 conditional-affine clauses; at 20 steps, the corresponding counts are 5,016 variables, 31,224 infrastructure clauses, and 10,240 conditional-affine clauses. Table 3 therefore reports bundle overhead, not the marginal cost of Algorithm 1. For clarity, the frozen five-word 20-step `pattern` instance adds 352 injection clauses; after XOR-to-DIMACS expansion its actual solver files contain 216,749 base clauses and 258,213 bundle clauses. These DIMACS counts are formula-artifact identities and are not the pre-expansion table entries.

5.3 Evidence Ladder and Decision Rules

Table 4 reports the completed frozen sequence through C16. A later study does not retroactively change an earlier estimand: C12 establishes a bundle effect on one formula, C13 and C14 test clause isolation on fixed formulas, C15 moves the isolated contrast to formula-level replication, and C16 prospectively extends that contrast to a multistratum frame.

5.4 C12: Fixed-Formula Production-Bundle Effect

C12 pairs base and full encodings of one frozen five-word, 20-step formula over ten CaDiCaL seeds. Nine pairs complete SAT/SAT and one base run times out while its full counterpart completes. Among the nine doubly completed pairs, every wall-time ratio favors the bundle; the geometric-mean base/full ratio is 16.4267 with paired bootstrap interval $[9.8270, 26.2822]$. The corresponding geometric-mean conflict ratio is 10.8225. Omitting the base-timeout/full-complete pair is conservative for the directional bundle comparison, but the reported ratio and interval remain conditional on joint completion. The protocol passes its predeclared completion and direction gates.

This result is deliberately formula- and intervention-specific. It shows that the production bundle materially changes search on the frozen formula. It cannot attribute the change to $\mathcal{C}_{\text{degen}}$ because the full mode also introduces the minimal-rule carry and difference infrastructure. In light of C13, the C12 contrast is compatible with an effect from that

Table 4: Confirmatory evidence ladder. Decisions follow the corresponding frozen protocol; "inconclusive" is not evidence of no effect.

Study	Experimental unit	Primary result	Decision
C12	20-step formula, seeds nested	bundle time ratio 16.43	bundle pass
C13	same formula, seeds nested	isolated time ratio 1.55	effect fail
C14	18-step formula, seeds nested	conflict ratio 0.853	inconclusive
C15	15 formulas, seeds nested	formula-level capped effect	solver-specific
C16	30 formulas, stratified	formula-level capped effect	too censored (both)

Table 5: BCP bridge checks on isolated nonlinear kernels. "Observed" counts successful implications in both directions, over both target truth values.

Kernel	Checks	Baseline	Preprocessed	Augmented
Ch	8	0	0	8
Maj	24	0	0	24
Total	32	0	0	32

infrastructure or from its interaction with the proposed clauses; the present design does not decompose those alternatives.

5.5 C13 and C14: Clause-Isolation Stress Tests

C13 reuses the C12 formula and seeds but compares minimal against full. All ten pairs complete SAT/SAT. Full is faster in seven pairs, below the predeclared eight-pair direction threshold. The geometric-mean minimal/full wall-time ratio is 1.5487, with interval $[0.9056, 2.6161]$; the interval crosses one. The secondary conflict ratio is 1.4567. The frozen decision is therefore *effect fail*: this experiment does not establish a conditional-affine clause effect on that formula.

C14 removes the reduced-model support proposal by using one unrestricted-message 18-step formula and 30 paired CaDiCaL seeds. Its primary conflict ratio (minimal/full) is 0.8527, with interval $[0.6555, 1.1064]$ and a 15:15 direction split. The secondary wall-time ratio is 0.8694, with interval $[0.6927, 1.0931]$. The result is *inconclusive*; if anything, the point estimates lean against the full encoding, but neither interval excludes one. C13 and C14 thus rule out using the large C12 bundle ratio as evidence for a general marginal clause effect.

5.6 Variable-Preserving Preprocessing Falsification Probe

We additionally isolate one word-level Ch formula and one word-level Maj formula and ask, on one fixed bit slice, whether CaDiCaL 3.0.0 preprocessing materializes the tested propagation bridges from the baseline Tseitin CNF. To preserve the meaning of the original variables, the probe disables elimination, fast elimination, decomposition, congruence, factorization, conditioning, blocking, covering, and decision search; it then executes ten preprocessing rounds. For every synthesized projection identity, both truth values and both propagation directions are checked by an independent unit-propagation implementation. Table 5 reports the result.

The preprocessed formulas preserve all 384 variables and contain neither an exact nor a subsuming copy of any of the four Ch or twelve Maj bridge clauses. Thus this schedule does not falsify the local BCP-blind-spot explanation, whereas explicitly augmenting the CNF exposes all tested implications. This is a deliberately weak negative result:

the script `experiments/run_inprocessing_falsification.py` and machine-readable receipt `experiments/results/inprocessing_falsification_20260713.json` cover one preserving preprocessing schedule on isolated kernels, not all production inprocessing schedules or equivalent clauses over eliminated or introduced variables.

5.7 Clause Overhead

Table 6 separates the clause templates by nonlinear kernel; the production-instance totals in Table 3 multiply these per-word counts by the number of steps.

Table 6: Conditional-affine clause statistics per 32-bit word

Function	Clauses/Bit	Clauses/Word	Lits/Clause
$\mathrm{Ch}(e, f, g)$	4	128	4
$\mathrm{Maj}(a, b, c)$	12	384	6

5.8 External-Validity Stress Test

Before C15, 30 frozen exact 12-word supports compared base against the full production bundle under one CaDiCaL seed and a 600s cap.

The predeclared gate required 15 joint completions; Table 7 reports the complete outcome matrix. Only four pairs complete in both modes, so no completed-only speedup or bootstrap interval is promoted. The five favorable and two unfavorable one-sided completions also cannot be converted into point ratios. This failure motivated the capped, formula-level C15 endpoint and remains evidence that the bundle result does not automatically generalize across denser supports.

5.9 C15: Formula-Level Conditional-Clause Isolation

Within our evidence ladder, C15 is the first study whose experimental unit is the formula. Its frozen frame contains the 15 weight-15 supports remaining after one pilot support was excluded; it is therefore a census of that restricted remainder, not a random sample of all SHA-256 formulas. Each support is solved in minimal and full mode under two nested seeds, separately for CaDiCaL 3.0.0 and Kissat 4.0.4. The common 300s cap is retained in the normalized endpoint rather than dropping timeouts.

The artifact retains the audited per-formula C15 figure at `figures/c15_formula_effects_v2_20260714.pdf`, including uninformative double-timeout markers and the predeclared ± 0.05 practical band. Table 8 gives the complete C15 decision-level summary retained in the evidence ladder.

The CaDiCaL endpoint corresponds to a mean capped-time saving of 23.68s per formula and passes the predeclared rule. The Kissat mean corresponds to 17.97s, but its interval crosses zero. Both medians are much smaller than their means, showing that a few supports carry much of the average effect. Moreover, CaDiCaL's interval excludes zero but not the entire 0.05 practical-margin band. The joint conclusion is therefore *CaDiCaL-specific benefit only*, not cross-solver benefit.

Post-confirmatory robustness. These checks do not alter the frozen decision rule. For CaDiCaL, every leave-one-formula-out mean remains above the practical margin (range $[0.0642, 0.0874]$), and the two seed means are 0.0727 and 0.0852. Its cap-specific mean remains positive from 30s through 300s but falls below 0.05 at the 60s and 120s caps; the median is 0.0266, and the three largest positive effects contribute 64.6% of positive mass.

Table 7: Outcome matrix for the earlier exact-support stress test. The completeness gate failed.

Base/full outcome	Count
SAT/SAT	4
TIMEOUT/SAT	5
SAT/TIMEOUT	2
TIMEOUT/TIMEOUT	19
UNSAT or error	0

Table 8: C15 formula-level results. Decisions are solver-stratified and follow the frozen benefit rule.

Solver	Informative	Mean	Median	Stability interval	Decision
CaDiCaL	14/15	0.078940	0.026626	$[0.026822, 0.135924]$	benefit
Kissat	12/15	0.059910	0.000214	$[-0.000574, 0.153975]$	inconclusive

The exact 300s sign-flip sensitivity calculation gives a two-sided value of 0.0106, but this is not a design-based p-value. Kissat is less stable: its leave-one-out mean reaches 0.0194, its seed means are 0.0918 and 0.0280, its 30s-cap mean is -0.0406, and its three largest positive effects contribute 94.0% of positive mass. These diagnostics narrow the CaDiCaL claim to a positive capped mean on the declared frame and retain the inconclusive Kissat decision.

The complete 30-pair outcome counts (both complete, minimal-timeout/full-complete, minimal-complete/full-timeout, both timeout) are respectively 22/2/0/6 for CaDiCaL and 21/3/0/6 for Kissat. The capped endpoint retains every category and never conditions on joint completion.

5.10 C16: Stratified Formula-Population Extension

C16 applies the isolated minimal-versus-full contrast to 30 exact-support 18-step formulas: ten each at weights 12, 13, and 14. The frame is a deterministic, outcome-blind hash-ranked selection from previously unused supports; it is not a probability sample. Each formula retains two nested seeds and every runtime is capped at 300 s. The primary estimand equally weights the three stratum means within each solver, while the frozen completeness rule requires at least 20 informative formulas overall and at least five in every stratum.

The independent reporting guard emits `CROSS_SOLVER_TOO_CENSORED`. Both tested solvers are too censored for a directional C16 effect decision. In each solver only 15 of 30 formulas are informative, and the weight-12 and weight-13 strata fall below the required five informative formulas. The capped point estimates and stability intervals in Table 9 remain descriptive summaries of the complete frozen endpoint; they cannot be promoted to a completed-only or favorable-stratum headline.

Table 10 makes the timeout mass explicit: double timeouts account for 35 of 60 nested-seed pairs under CaDiCaL and 36 of 60 under Kissat, while neither solver exhibits a terminal-status mismatch.

Figure 2 retains every formula and displays the stratum structure rather than pooling solvers or support weights. CaDiCaL stratum median and formula signs (positive/negative/zero): $w = 12$ 0.000000 (1/1/8); $w = 13$ 0.000000 (1/3/6); $w = 14$ -0.007573 (4/5/1). Kissat stratum median and formula signs (positive/negative/zero): $w = 12$ 0.000000 (2/2/6); $w = 13$ 0.000000 (3/1/6); $w = 14$ 0.003517 (6/1/3). These stratum summaries show substantial zero mass and no uniform directional pattern.

Table 9: C16 solver-stratified normalized capped differences. Each stratum cell gives the mean with informative formulas in parentheses; positive means favor full, and intervals are stratified stability diagnostics.

Solver	Informative	$w = 12$	$w = 13$	$w = 14$	Overall	Stability interval
CaDiCaL	15/30	0.027203 (2/10)	-0.106827 (4/10)	0.040046 (9/10)	-0.013193	$[-0.078432, 0.049078]$
Kissat	15/30	-0.015584 (4/10)	0.026822 (4/10)	0.021812 (7/10)	0.011017	$[-0.046145, 0.064076]$

Table 10: C16 paired outcome counts over 60 nested-seed pairs per solver. Full-only and minimal-only retain the opposite-mode timeout at the 300 s cap.

Solver	matching terminal	terminal mismatch	full only	minimal only	double timeout
CaDiCaL	11	0	7	7	35
Kissat	13	0	8	3	36

CaDiCaL secondary diagnostics: nested-seed means 0.012295/-0.038680; leave-one-formula-out range $[-0.027511, 0.007089]$; top-three positive-mass share 73.8%. Eligible-count weighting gives -0.002509; finite-population correction gives $[-0.079390, 0.053004]$.

Kissat secondary diagnostics: nested-seed means -0.040429/0.062462; leave-one-formula-out range $[-0.000803, 0.027911]$; top-three positive-mass share 60.3%. Eligible-count weighting gives -0.004213; finite-population correction gives $[-0.046026, 0.068060]$.

The top-three positive-mass share is defined as zero when no positive formula effect exists. The eligible-count-weighted mean is a secondary descriptive reweighting of the retained stratum means, not a design-based population estimator. The finite-population correction and the deletion, seed, concentration, and eligible-count reweighting diagnostics are secondary and cannot upgrade or repair a frozen solver decision. The retained shared-host load trace is diagnostic only. The two nested seeds, deterministic frame, capped endpoint, stratum heterogeneity, and shared-host measurements therefore remain explicit limits on interpretation.

5.11 Fresh Fixed-Support Artifact Replay

A separate artifact-evaluation replay checks that the current production bundle remains executable under a pinned environment. A single nine-word support $\{3, 4, 5, 6, 7, 8, 11, 12, 13\}$ is generated once and reused across modes. The retained container is restricted to four CPUs and 8 GiB memory, has no network, a read-only root filesystem, dropped capabilities, and an executable temporary filesystem. The image contains Python 3.10.12, CaDiCaL 3.0.0, Kissat 4.0.4, and CryptoMiniSat 5.8.0.

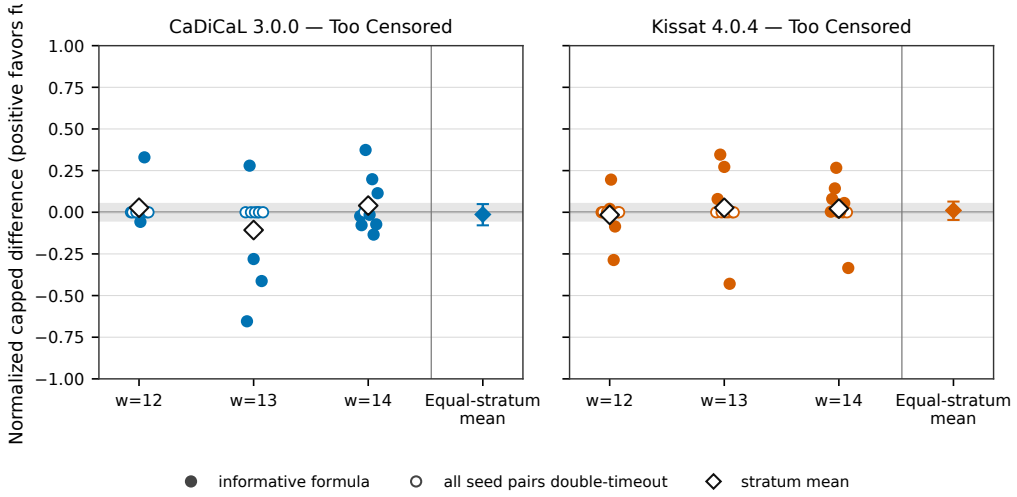


Figure 2: C16 formula effects by solver and support-weight stratum. Hollow points denote formulas whose two nested-seed pairs are double timeouts; diamonds denote stratum means, and the overall marker carries the frozen stratified stability interval.

Table 11: Fresh 18-step fixed-support replay. Each cell is one run; full precedes base for every solver. Ratios are base/full, are computed from full-precision raw times rather than the rounded table entries, and are descriptive functional-artifact evidence for the production bundle: a bundle-level execution check, not clause-attributable.

Solver	Base (s)	Full (s)	Time ratio
CaDiCaL	40.876	1.884	21.69
Kissat	23.808	0.761	31.27
CryptoMiniSat	246.342	28.898	8.52

Table 11 reports six completed runs, and the audit verifies identical support identity, base and full CNF digests, zero DIMACS count drift, and the expected 18,816 counted delta rules in full mode: 9,600 minimal-infrastructure rules plus 9,216 conditional-affine clauses. Conflict reductions are 95.83%, 97.84%, and 89.77% for CaDiCaL, Kissat, and CryptoMiniSat, respectively. The raw-result and audit hashes in Table 12 bind the runtimes, conflicts, and rule counts. These observations validate the packaged workflow and reproduce a large bundle contrast on one support. They are not replicated formula-level estimates, do not isolate $\mathcal{C}_{\text{degen}}$, and retain a fixed-order confound. In particular, the ratios cannot be cited as empirical support for the proposed clause layer; they are equally compatible with an effect from the minimal infrastructure or an interaction within the bundle. CryptoMiniSat propagation counts are omitted because parser comparability was not established.

5.12 Artifact Identity and Claim Binding

Table 12 gives short review identifiers; full artifact hashes bind the raw data, analysis code, and image before any headline decision is reproduced.

Table 12: Short SHA-256 identifiers for the promoted evidence.

Evidence	Object	SHA-256 prefix
C12	raw paired result	`40e8d9de539ee3a6`
C13	raw isolated result	`dfb38d9e22fa8ab6`
C14	raw unrestricted result	`827427e133b991a8`
C15	frozen formula frame	`f5d17d517247b840`
C15	CaDiCaL/Kissat raw results	`cb048eda/31d8a83c`
C15	final analysis	`b32356116d8a36bb`
C15	robustness analysis	`0a9baff4f5cdf31d`
C15	container image	`63b7c11f24aacaa6`
C16	frame manifest	`5021567d8d761469`
C16	CaDiCaL/Kissat raw results	`318a3557/fe0614ad`
C16	terminal analysis	`e11f4f5b33fbe7af`
C16	sanitized host-load receipt	`e10073c7c72f776d`
C16	final figure	`3754b2eae638c957`
AE replay	container image	`468b753636d5d949`
AE replay	raw result/audit	`86f00fed/05bca0f9`

6 Limitations

Cryptanalytic scope and formula population. The experiments concern 18- and 20-step semi-free-start collision formulas; they neither produce a new SHA-256 collision nor justify extrapolation to the full compression function. A support is an admissible message-difference region rather than a complete characteristic. C12–C14 each use one formula, so their seeds quantify search variability only. C15 covers the 15 remaining exact weight-15 supports after excluding one pilot and is a census only of that restricted remainder. C16 adds 30 deterministically ranked exact supports across weights 12–14, not a random or representative draw from all supports, step counts, differential paths, or SHA-256 formulas.

Estimand and uncertainty. C15 and C16 use only two seeds per formula. Averaging them before resampling prevents pseudoreplication but leaves within-formula seed variability coarsely measured. Their percentile intervals measure stability under empirical formula resampling; they are not design-based confidence intervals for an eligible or full support population. C15's CaDiCaL mean passes its frozen rule but remains concentrated and cap-sensitive. C16 does not pass its completeness gate for either solver, so its point estimates, deletion ranges, seed means, and alternative reweightings support no directional upgrade.

Censoring and practical threshold. Capped differences retain all timeout pairs but collapse runtimes beyond 300s. They estimate performance under that resource budget, not the uncensored runtime distribution or its mean. Double timeouts therefore contribute zero observed capped difference without implying equal latent runtimes. CaDiCaL's C15 interval lies above zero but overlaps the predeclared 0.05 practical-margin region. In C16, both solver labels are too censored, including severe double-timeout mass and sub-threshold informative counts in two strata; no completed-pair subset repairs that failure. Estimating post-cap behavior would require a separately specified censoring model and diagnostics; it cannot be recovered by reinterpreting the present endpoint after observing the results.

Intervention and measurement boundaries. The fixed-support replay and C12 compare base with the whole production bundle; only C13–C16 isolate the conditional-affine clauses.

The replay has one run per cell and a fixed full-before-base order, so it is functional artifact evidence rather than an effect estimate. Wall time is hardware- and load-sensitive; conflicts and decisions are solver-configuration-specific. Seed replication cannot substitute for formula replication.

Clause overhead. At 18 steps, the full bundle adds approximately 56% to the pre-expansion non-XOR clause count. The proposed layer contributes 9,216 clauses, or 14.2% of the base non-XOR count, and no variables; the remaining overhead belongs to the minimal infrastructure. We have not located a step-count or support-density crossover at which watched-literal cost dominates propagation benefit.

Synthesis and preprocessing scope. Algorithm 1 exhaustively verifies minimum-control projection identities for the three-input Ch and Maj kernels. It does not enumerate every affine consequence, establish tractability for larger S-boxes, or show that the same construction helps other primitives. The preprocessing probe covers one variable-preserving CaDiCaL schedule on isolated kernels. More aggressive inprocessing may derive propagation-equivalent constraints through eliminated or introduced variables; evaluating that possibility requires proof-trace-aware equivalence analysis.

Reproducibility boundary. The retained image digest, source hashes, CNF digests, raw-result hashes, and resource limits bind the reported replay to a concrete environment. Package registries and base images can change, so the Dockerfile alone does not promise a bitwise-identical future rebuild. The artifact verifier therefore checks the retained image and receipts rather than inferring identity from version strings.

7 Conclusion

We identified a precise BCP blind spot in differential Tseitin encodings of SHA-256 Ch and Maj, proved that conditional-affine bridge clauses preserve the model set while strictly strengthening unit propagation on explicit gate fragments, and replaced hand-written templates with an exhaustively verified minimum-control projection synthesizer.

The experiments delimit the performance claim. The production bundle yields a large paired contrast on one frozen 20-step formula and in a fresh fixed-support artifact replay, but these comparisons cannot support clause-layer performance and may reflect the minimal infrastructure or an interaction within the bundle. When the clause layer is isolated, C13 fails its fixed-formula effect gate and C14 is inconclusive. Formula-level C15 passes its predeclared CaDiCaL benefit rule over the restricted frame, while Kissat remains inconclusive; the broader prospective C16 extension fails its completeness gate under both tested solvers. C16 broadens the deterministic frame to exact support weights 12–14, with frozen decisions of *too censored* for CaDiCaL and *too censored* for Kissat; the solver-specific labels are retained in the overall claim boundary. The supported result is therefore a sound propagation construction with bounded, solver- and frame-specific performance evidence—not a universal SAT speedup or a new cryptanalytic attack. Broader claims require more step counts, less restricted formula populations, more nested seeds, and independent solver families under a predeclared analysis.

References

[ANB24] Nahiyan Alamgir, Saeed Nejati, and Curtis Bright. SHA-256 collision attack with programmatic SAT. In *Joint Proceedings of the 9th Workshop on Practical Aspects of Automated Reasoning (PAAR) and the 9th Satisfiability Checking*

and Symbolic Computation Workshop (SC-Square), volume 3717 of *CEUR Workshop Proceedings*, pages 91–110, 2024. arXiv:2406.20072. URL: https://ceur-ws.org/Vol-3717/paper5.pdf, doi:10.48550/arXiv.2406.20072.

[AS09] Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern SAT solvers. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 399–404, 2009.

[BBČ+13] Martin Babka, Tomáš Balyo, Ondřej Čepek, Štefan Gurský, Petr Kučera, and Václav Vlček. Complexity issues related to propagation completeness. *Artificial Intelligence*, 203:19–34, 2013. doi:10.1016/j.artint.2013.07.006.

[BF22] Armin Biere and Mathias Fleury. Gimsatul, isasat and kissat entering the sat competition 2022. In *Proc. of SAT Competition 2022 – Solver and Benchmark Descriptions*, pages 10–11, 2022.

[BFFH20] Armin Biere, Katalin Fazekas, Mathias Fleury, and Maximilian Heisinger. CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT competition 2020. In *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*, pages 51–53, 2020.

[BMS12] Lucas Bordeaux and Joao Marques-Silva. Knowledge compilation with empowerment. In *SOFSEM 2012: Theory and Practice of Computer Science*, volume 7147 of *Lecture Notes in Computer Science*, pages 612–624. Springer, 2012. doi:10.1007/978-3-642-27660-6_50.

[EB05] Niklas Eén and Armin Biere. Effective preprocessing in SAT through variable and clause elimination. In *Theory and Applications of Satisfiability Testing – SAT 2005*, volume 3569 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 2005. doi:10.1007/11499107_5.

[FNP+23] Katalin Fazekas, Aina Niemetz, Mathias Preiner, Markus Kirchweger, Stefan Szeider, and Armin Biere. IPASIR-UP: User propagators for CDCL. In *Theory and Applications of Satisfiability Testing – SAT 2023*, volume 271 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.SAT.2023.8, doi:10.4230/LIPIcs.SAT.2023.8.

[GK14] Matthew Gwynne and Oliver Kullmann. On SAT representations of XOR constraints. In *Language and Automata Theory and Applications – LATA 2014*, volume 8370 of *Lecture Notes in Computer Science*, pages 409–420. Springer, 2014. doi:10.1007/978-3-319-04921-2_33.

[HJW14] Marijn J. H. Heule, Warren A. Hunt Jr., and Nathan Wetzler. DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In *Theory and Applications of Satisfiability Testing – SAT 2014*, volume 8561 of *Lecture Notes in Computer Science*, pages 422–429. Springer, 2014. doi:10.1007/978-3-319-09284-3_31.

[HvM05] Marijn J. H. Heule and Hans van Maaren. Aligning CNF- and equivalence-reasoning. In *Theory and Applications of Satisfiability Testing – SAT 2004*, volume 3542 of *Lecture Notes in Computer Science*, pages 145–156. Springer, 2005. doi:10.1007/11527695_12.

[ISSL16] Jeevana Priya Inala, Rohit Singh, and Armando Solar-Lezama. Synthesis of domain specific CNF encoders for bit-vector solvers. In *Theory and Applications of Satisfiability Testing – SAT 2016*, volume 9710 of *Lecture Notes in Computer Science*, pages 302–320. Springer, 2016. doi:10.1007/978-3-319-40970-2_19.

[JHB12] Matti Järvisalo, Marijn J. H. Heule, and Armin Biere. Inprocessing rules. In *Automated Reasoning – IJCAR 2012*, volume 7364 of *Lecture Notes in Computer Science*, pages 355–370. Springer, 2012. doi:10.1007/978-3-642-31365-3_28.

[Li03] Chu Min Li. Equivalent literal propagation in the DLL procedure. *Discrete Applied Mathematics*, 130(2):251–276, 2003. doi:10.1016/S0166-218X(02)00407-9.

[LLL+25] Guozhen Liu, Shun Li, Huina Li, Weidong Qiu, and Siwei Sun. SAT-based space partitioning and applications to Ascon-Hash256 cryptanalysis. *IACR Transactions on Symmetric Cryptology*, 2025(3):577–632, 2025. ePrint 2025/1542. doi:10.46586/tosc.v2025.i3.577-632.

[LLQZ26] Yingxin Li, Fukang Liu, Haifeng Qian, and Jinwei Zhu. Pushing collision attacks on SHA-2 to 39 steps. Cryptology ePrint Archive, Report 2026/1120, 2026. Preprint. URL: https://eprint.iacr.org/2026/1120.

[LLW24] Yingxin Li, Fukang Liu, and Gaoli Wang. New records in collision attacks on SHA-2. In *Advances in Cryptology – EUROCRYPT 2024*, volume 14651 of *Lecture Notes in Computer Science*, pages 158–186. Springer, 2024. doi:10.1007/978-3-031-58716-0_6.

[LLWS26] Yingxin Li, Fukang Liu, Gaoli Wang, and Jiali Shi. Pushing the limit of memory-efficient collision attack framework for SHA-2. Cryptology ePrint Archive, Report 2026/1080, 2026. To appear at CRYPTO 2026. URL: https://eprint.iacr.org/2026/1080.

[Man12] Norbert Manthey. Coprocessor 2.0: A flexible CNF simplifier. In *Theory and Applications of Satisfiability Testing – SAT 2012*, volume 7317 of *Lecture Notes in Computer Science*, pages 436–441. Springer, 2012. doi:10.1007/978-3-642-31612-8_34.

[MCS13] Amit Metodi, Michael Codish, and Peter J. Stuckey. Boolean equi-propagation for concise and efficient SAT encodings of combinatorial problems. *Journal of Artificial Intelligence Research*, 46:303–341, 2013. doi:10.1613/jair.3809.

[MNS11] Florian Mendel, Tomislav Nad, and Martin Schläffer. Finding SHA-2 characteristics: Searching through a minefield of contradictions. In *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 288–307. Springer, 2011. doi:10.1007/978-3-642-25385-0_16.

[MNS13] Florian Mendel, Tomislav Nad, and Martin Schläffer. Improving local collisions: New attacks on reduced SHA-256. In *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 262–278. Springer, 2013. doi:10.1007/978-3-642-38348-9_16.

[MWGP12] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In *Information Security and Cryptology – Inscrypt 2011*, volume 7537 of *Lecture Notes in Computer Science*, pages 57–76. Springer, 2012. doi:10.1007/978-3-642-34704-7_5.

[MZ06] Ilya Mironov and Lintao Zhang. Applications of SAT solvers to cryptanalysis of hash functions. In *Theory and Applications of Satisfiability Testing – SAT 2006*, volume 4121 of *Lecture Notes in Computer Science*, pages 102–115. Springer, 2006. doi:10.1007/11814948_13.

[Nat15] National Institute of Standards and Technology. Secure hash standard (SHS). Technical Report FIPS PUB 180-4, National Institute of Standards and Technology, 2015. URL: https://csrc.nist.gov/pubs/fips/180-4/upd1/final, doi:10.6028/NIST.FIPS.180-4.

[NG19] Saeed Nejati and Vijay Ganesh. CDCL(Crypto) SAT solvers for cryptanalysis. In *Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering (CASCON 2019)*, pages 311–316. ACM, 2019. arXiv:2005.13415. URL: https://dl.acm.org/doi/10.5555/3370272.3370307.

[Nos12] Vegard Nossum. SAT-based preimage attacks on SHA-1. Master's thesis, University of Oslo, 2012. URL: https://hdl.handle.net/10852/34912.

[Pre09] Steven D. Prestwich. CNF encodings. In *Handbook of Satisfiability*, pages 75–97. IOS Press, 2009. doi:10.3233/978-1-58603-929-5-75.

[RHB23] Joseph E. Reeves, Marijn J. H. Heule, and Randal E. Bryant. Preprocessing of propagation redundant clauses. *Journal of Automated Reasoning*, 67(3):31, 2023. doi:10.1007/s10817-023-09681-3.

[SAT24] SAT Competition 2024 Organizers. SAT Competition 2024: Competition tracks. Official competition website, 2024. Accessed 2026-07-15. URL: https://satcompetition.github.io/2024/tracks.html.

[SGM20] Mate Soos, Stephan Gocht, and Kuldeep S. Meel. Tinted, detached, and lazy CNF-XOR solving and its applications to counting and sampling. In *Computer Aided Verification – CAV 2020*, volume 12224 of *Lecture Notes in Computer Science*, pages 463–484. Springer, 2020. doi:10.1007/978-3-030-53288-8_22.

[SHW+14] Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. Automatic security evaluation and (related-key) differential characteristic search: Application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In *Advances in Cryptology – ASIACRYPT 2014*, volume 8873 of *Lecture Notes in Computer Science*, pages 158–178. Springer, 2014. doi:10.1007/978-3-662-45611-8_9.

[SNC09] Mate Soos, Karsten Nohl, and Claude Castelluccia. Extending SAT solvers to cryptographic problems. In *Theory and Applications of Satisfiability Testing – SAT 2009*, volume 5584 of *Lecture Notes in Computer Science*, pages 244–257. Springer, 2009. doi:10.1007/978-3-642-02777-2_24.

[Ste13] Marc Stevens. New collision attacks on SHA-1 based on optimal joint local-collision analysis. In *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 245–261. Springer, 2013. doi:10.1007/978-3-642-38348-9_15.

[Tse83] Grigori S. Tseitin. On the complexity of derivation in propositional calculus. In *Automation of Reasoning*, pages 466–483. Springer, 1983.

[TSI+25] Kazuma Taka, Kosei Sakamoto, Ryoma Ito, Rentaro Shiba, Shion Utsumi, and Takanori Isobe. Divide-and-conquer SAT for exploring optimal differential and linear characteristics and its applications. *IACR Transactions on Symmetric Cryptology*, 2025(3):516–576, 2025. doi:10.46586/tosc.v2025.i3.516-576.

[TWW+20] Alexander Tornede, Marcel Wever, Stefan Werner, Felix Mohr, and Eyke Hüllermeier. Run2Survive: A decision-theoretic approach to algorithm selection based on survival analysis. In *Proceedings of the 12th Asian Conference on Machine Learning*, volume 129 of *Proceedings of Machine Learning Research*, pages 737–752. PMLR, 2020. URL: https://proceedings.mlr.press/v129/tornede20a.html.

[WGS19] Gellert Weisz, András György, and Csaba Szepesvári. CapsAndRuns: An improved method for approximately optimal algorithm configuration. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6707–6715. PMLR, 2019. URL: https://proceedings.mlr.press/v97/weisz19a.html.

[ZHL25] Zhongyi Zhang, Chengan Hou, and Meicheng Liu. Preimage attacks on up to 5 rounds of SHA-3 using internal differentials. In *Advances in Cryptology – EUROCRYPT 2025*, volume 15601 of *Lecture Notes in Computer Science*, pages 333–363. Springer, 2025. ePrint 2025/439. doi:10.1007/978-3-031-91107-1_12.

[ZLGW26] Zhuolong Zhang, Muzhou Li, Lei Gao, and Meiqin Wang. Collision attacks on SHA-256 up to 37 steps with improved trail search. In *Advances in Cryptology – EUROCRYPT 2026*, volume 16546 of *Lecture Notes in Computer Science*, pages 91–120. Springer, 2026. IACR Cryptol. ePrint Arch. 2026/232. doi:10.1007/978-3-032-25333-0_4.