The Carry Adder Wall

Peter Hollows

June 12, 2026

Abstract

Bitcoin mining searches for a block header whose double-SHA-256 hash falls below a target, a weak-preimage problem solved in practice by brute force. We ask whether any classical algorithm beats brute force, and answer it by decomposing SHA-256d into two layers with opposite properties. The message schedule is a fixed recurrence whose computation is shareable across candidates. We show that every sharing win is confined to three known, bounded regions (midstate caching, the ASICBoost message-expansion region, and a provably unshareable second hash), so the per-candidate cost is close to fixed. The round function is pseudorandom. We prove, in the random-oracle model, that the expected candidate count cannot fall below 2^D, a bound that survives preprocessing (Bitcoin is salted) and quantum search (Grover gives only a quadratic speedup), and we reduce the standard-model claim to a SHA-256 distinguisher. We then explain why the round function resists prediction: the carry chain of modular addition is its only position-coupling nonlinearity. We introduce *carry depth*, the count of carry layers between a controllable input and a target output, as a coordinate of the proof-of-work function space; we compute it exactly at 386 layers for SHA-256d, and we measure that the strongest local advantage we tested falls below detection (consistent with zero) within a single round. That single-round collapse, not the size of the in-round advantage, is the empirical content; the strongest local attack we tested therefore expires roughly 100 rounds before reaching the output. We are explicit about the ceiling: the result is conditional on the carry/local attack class, and the entire residual reduces to a global algebraic shortcut that would constitute a SHA-256 break, whose unconditional exclusion is, in the asymptotic framing, the $\mathsf{P} \neq \mathsf{NP}$ problem.

1 The mining search problem

Bitcoin mining is a search problem over an iterated hash. A block header H is 80 bytes; the mining hash is the double application

$$\mathsf{M}(H) \;=\; \text{SHA-256}\big(\text{SHA-256}(H)\big),$$

read as a 256-bit integer. Mining succeeds when a miner finds an H (within its allowed degrees of freedom: the 32-bit nonce, the coinbase extranonce that varies the Merkle root, version bits, and the timestamp) such that $\mathsf{M}(H) < T$, where $T = 2^{256} \cdot 2^{-D}$ and D is the difficulty expressed in leading zero bits. Equivalently, H must land in a target set $S \subseteq \{0,1\}^{256}$ of density $|S|/2^{256} = 2^{-D}$.

The question is operational: *is there any classical algorithm that finds such an H at materially less expected cost than brute force?* Brute force tries candidates until one lands in S, at expected cost $\approx 2^D$ evaluations. An attack "beats" it only if it reduces the total work by more than a constant factor.

Cost model. The atomic unit is one SHA-256 compression-function call $h : \{0,1\}^{256} \times \{0,1\}^{512} \to \{0,1\}^{256}$. It applies the block cipher E (SHACAL-2, keyed by the message m) to the chaining value and feeds the input forward by *wordwise modular addition*, $h(C, m) = C \boxplus E_m(C)$, where $\boxplus$ adds each of the eight 32-bit words modulo 2^{32}. The feed-forward is thus itself a carry operation, which matters in Section 5. One header evaluation decomposes into exactly three compression calls,

$$\underbrace{C_1 = h(\mathrm{IV}, B_0)}_{\text{block 0 (nonce-independent)}} \quad , \quad \underbrace{C_2 = h(C_1, B_1 \| \mathrm{pad})}_{\text{block 1 (nonce in word } W_3)} , \quad \underbrace{\mathrm{out} = h(\mathrm{IV}, C_2 \| \mathrm{pad})}_{\text{second hash}},$$

with B_0 the first 64 header bytes and B_1 the remaining 16 plus padding. Total mining cost is the number of candidate evaluations multiplied by the compression calls per candidate that sharing cannot remove. These two factors are governed by different structure, which the next section makes precise.

2 Two layers

The analysis rests on one structural fact: SHA-256d is built from two kinds of structure with opposite properties.

layer	**property**	**consequence**
message schedule	fixed recurrence (same per block)	computation is *shareable* across candidates
round / compression	pseudorandom	output is *unpredictable*; search is *irreducible*

The only avenues to beat brute force live in one layer or the other: either reduce the *per-candidate cost* by sharing schedule computation (Section 3), or reduce the *candidate count* by predicting or skipping evaluations (Sections 4–5). We show the first avenue is bounded to known constant-factor techniques, and the second is impossible in the ideal model and equivalent to breaking SHA-256 in the standard model. Because the two avenues act on different layers, a gain in one neither requires nor enables a gain in the other, and the rest of the paper addresses them in turn.

3 The amortization envelope

Written as a product, the cost is

$$\text{cost} \;\geq\; \underbrace{(\text{candidate evaluations})}_{\text{locked at } 2^D,\ §4} \;\times\; \underbrace{(\text{compressions per candidate after maximal sharing})}_{\text{this section}}.$$

The left factor is locked at 2^D by the search lower bound. Hence any economically meaningful win against brute force must come from the right factor: per-candidate computation sharing. We bound it by classifying which of the three compressions can be shared across *distinct* candidates.

- **Block 0 (midstate).** Nonce-independent, and shared across the entire nonce sweep of a fixed header prefix, so its amortized cost per candidate falls to zero. This is midstate caching, standard practice in deployed miners.

- **Second hash.** Effectively unshareable. Two candidates share the second-hash compression only if they share a first-hash digest, which is a SHA-256 collision. Finding one costs about 2^{128}, far more than mining the block itself (cost 2^D with $D < 128$), so the second hash cannot be shared profitably and every candidate pays one full second-hash compression. This rests on the collision resistance of SHA-256, the mildest cryptographic assumption in the paper, and no input-side technique avoids it.
- **Block 1 expansion.** Block 1 of the first hash contains the nonce, and only its message-schedule expansion is shareable, across small batches of candidates engineered to collide in the expansion. ASICBoost [2] exploits exactly this region—covertly via the Merkle root, or overtly via version rolling, standardized in the reserved `nVersion` bits [27]—a deployed technique worth about 20%, and it is the only region where a sharing win can live.

These three regions are the three compressions of the cost model, so they exhaust the per-candidate work: the sharing envelope is exactly midstate (at floor), the ASICBoost block-1 expansion (near-saturated), and the second hash (provably unshareable), with nothing left over. Sharing *between* the compressions is excluded by the same data dependency: each compression consumes the previous one's output, so two candidates share a later compression only if they collide in an earlier digest, which is again a SHA-256 collision at cost 2^{128}. The amortization avenue is therefore known, bounded, and already occupied, and the remainder of this paper concerns the candidate-count factor, where the open question lay.

4 The search lower bound

We first establish, unconditionally in the ideal model, that the candidate-count factor cannot be reduced below 2^D in expectation; then we transfer the statement to the standard model by a tight reduction.

4.1 Ideal model

Model the search map $H \mapsto \mathsf{M}(H)$ as a random oracle: with the compression function idealized, fresh inputs receive uniform, independent outputs (composition of ideal primitives is not automatic in general; the construction-level caveats are handled in the standard-model reduction below). This is the standard idealization under which proof-of-work hardness is proved [1].

Theorem 1 (ROM search optimality)**.** *Let $\mathcal{A}$ make at most q queries to the oracle and output a candidate H^*. Then*

$$\Pr[\mathsf{M}(H^*) \in S] \;\le\; (q+1)\,2^{-D},$$

over the oracle and $\mathcal{A}$'s coins. Consequently $\mathcal{A}$ needs $q \ge 2^{D-1} - 1$ queries to succeed with probability $\ge \frac{1}{2}$, and the expected number of queries to the first success is $\ge 2^D$.

Proof sketch. Without loss of generality the queries $x_1, \ldots, x_q$ are distinct. Each $\mathrm{RO}(x_i)$ is uniform on $\{0,1\}^{256}$, independent of all prior responses, so $\Pr[\mathrm{RO}(x_i) \in S] = 2^{-D}$. If H^* was queried, success is among these events; if not, $\mathrm{RO}(H^*)$ is uniform and independent, contributing a further 2^{-D}. A union bound over the q queries and the final output gives $(q+1)2^{-D}$. The query statement is immediate. For the expectation, random-oracle freshness makes each query succeed with probability at most 2^{-D} conditioned on all prior responses, so the first-success time Q stochastically dominates a Geometric(2^{-D}) variable: $\Pr[Q > q] \ge (1 - 2^{-D})^q$. Hence $\mathbb{E}[Q] = \sum_{q \ge 0} \Pr[Q > q] \ge \sum_{q \ge 0} (1 - 2^{-D})^q = 2^D$. □

Brute force achieves expected 2^D evaluations, so it is exactly query-optimal in expectation: no candidate ordering, feature, or filter reduces the count below 2^D in this model.

4.2 Closing the realistic loopholes

Precomputation and time–memory trade-offs. A miner might precompute an enormous table offline. The auxiliary-input (non-uniform) random-oracle framework [3, 4, 5] models an adversary with S bits of arbitrary advice and q online queries and shows that *salting* generically defeats preprocessing. For the salted inversion problem the advantage is bounded by $ST/(K\alpha) + T/\alpha$, which for a salt as wide as the function's output ($K = N$) collapses to $\mathcal{O}(T/N)$ [13], i.e. the non-preprocessing bound up to constants. Bitcoin is salted by construction: each header commits to `prev_block_hash`, a fresh, unpredictable 256-bit value, and to the Merkle root of fresh transactions, so the per-block salt is effectively as wide as the function output. Classical time–memory trade-offs [14] and their rainbow-table refinement [15] require a *fixed* target function so that one precomputed table amortizes over all instances; a fresh per-block salt makes each block a distinct function, and offline computation cannot target a block whose predecessor is not yet known, so these trade-offs do not apply.

Quantum. Grover search gives a quadratic speedup, $\mathcal{O}(2^{D/2})$ queries, and this is optimal (the BBBV lower bound [6]): no exponential speedup exists even quantumly. This carries over to the mining setting specifically: the post-quantum analysis of the Bitcoin backbone proves that each quantum query is worth only $\mathcal{O}(p^{-1/2})$ classical queries, with a unified hybrid quantum/classical search bound interpolating the two regimes [11, 12]. The quadratic factor is economically irrelevant at current scale, since a coherent SHA-256d Grover oracle costs orders of magnitude more per query than a classical ASIC gate-pass.

Standard-model reduction. A concrete miner $\mathcal{M}$ against the true SHA-256 with expected cost $2^D/\alpha$ for super-constant α succeeds at a rate exceeding the ceiling that Theorem 1 imposes on every algorithm in the ROM. Running $\mathcal{M}$ against a real or an ideal primitive and checking whether it beats the bound is then a distinguishing experiment, formalized by indifferentiability [16]: if the mining map is indifferentiable from a random oracle, the ROM bound transfers to the standard model up to the distinguishing advantage, and standard-model proof-of-work primitives of exactly this lower-bound shape have been formalized [10]. One subtlety must be handled. Plain (strengthened) Merkle–Damgård is *not* indifferentiable from a random oracle, because of length extension: from SHA-256(x) one can predict SHA-256($x\|\text{pad}\|y$) without knowing x [8]. This is the structure that the *outer* application in SHA-256d = SHA-256(SHA-256($\cdot$)) is designed to remove (length extension specifically): re-hashing the full digest is intended to seal the inner chaining value behind a second, length-committed invocation, so that the length-extension distinguisher does not reach the mining map. The literature comes close to this composition from both sides without covering it. Coron et al. [8] prove plain Merkle–Damgård indifferentiable (with birthday bounds) on prefix-free inputs, of which fixed-length inputs are a special case—so a *single* SHA-256 restricted to 80-byte headers is covered—and separately prove indifferentiability of the double hash $H(H(0^\kappa\|M))$, which differs from SHA-256d only by the prepended zero block whose role is to domain-separate the outer single-compression call from adversarially controlled first message blocks; SHA-256d omits that block, so the theorem does not literally apply. Dodis et al. [9] analyze the unrestricted double hash $H^2(M) = H(H(M))$ and show it is indifferentiable only with weak bounds (any successful simulator must make $\Omega(\min\{q_1 q_2, 2^{n/2}\})$ queries, capping composition-derived security near $2^{n/4}$); their differentiating attack, however, requires feeding n-bit digests back

into the construction, which the mining map's fixed 80-byte domain forbids, and they prove no positive result for the domain-restricted composition. No published theorem covers SHA-256d on its fixed-length domain, so we state indifferentiability of the mining map as an explicit assumption, consistent with formal treatments of Bitcoin that model the mining function as a random oracle directly [1]. Indifferentiability composes for single-stage games; multi-stage settings can require more care [17], but the mining experiment is single-stage—one adversary, one search.

Theorem 2 (conditional, informal)**.** *If the mining map* $\mathsf{M} = \text{SHA-256}(\text{SHA-256}(\cdot))$ *is indifferentiable from a random oracle (with its compression function modeled as ideal), then no mining algorithm achieves expected cost below* $2^D/\mathcal{O}(1)$ *candidate evaluations, up to the indifferentiability advantage; an algorithm that does yields a distinguisher against* SHA-256*.*

The provability wall. A fully unconditional, standard-model version of Theorem 2 is beyond current techniques. Mining for the fixed function SHA-256 is finite, so the statement is not literally about P versus NP; an unconditional proof would be an explicit circuit lower bound, which no method can establish for a function of this complexity. In the asymptotic framing, for a scaled family of SHA-256-like functions, proving that search requires $\sim 2^D$ work would exhibit an explicit one-way function and so separate P from NP. Either way, the conditional reduction above is the strongest conclusive statement available, and it localizes the entire risk to one assumption.

4.3 Localization

Corollary 1. *Every economically meaningful win against brute force is confined to the per-candidate sharing factor of Section 3. Output prediction and search-skipping are impossible in the ideal model (Theorem 1) and equivalent to a* SHA-256 *break in the standard model (Theorem 2).*

The remaining sections explain, mechanistically and quantitatively, why the round-function layer admits no such prediction. This is the content that makes Theorem 2's assumption believable.

5 The carry-depth mechanism

5.1 Carries are the entire nonlinearity of addition

For $c = a + b \bmod 2^n$, define the carry vector γ by $\gamma_0 = 0$ and $\gamma_{i+1} = \text{MAJ}(a_i, b_i, \gamma_i)$. Then bit by bit,

$$c_i = a_i \oplus b_i \oplus \gamma_i, \qquad \text{i.e.} \qquad a + b = a \oplus b \oplus \gamma(a, b).$$

The $a \oplus b$ term is GF(2)-linear; *all* nonlinear, cross-position behaviour of addition is carried by γ. If γ were known, addition would be pure XOR, i.e. linear. Fixing every nonlinear-gate output (each carry bit, each Ch/Maj term) collapses SHA-256d to a GF(2)-affine map, and finding an input that hits the target becomes a solvable linear system. *Hardness is the cost of resolving carries.*

5.2 Carries are the only position-coupling nonlinearity

Classify each SHA-256 operation by how it moves information:

operation	linear over GF(2)?	couples bit positions?
rotations, shifts, XOR, $\Sigma_0, \Sigma_1, \sigma_0, \sigma_1$	yes	linearly
Ch, Maj	no (degree 2)	no; bit-local (out_i sees only in_i)
modular +	no	**yes**; out_i depends on all $\text{in}_{<i}$

The modular adder is the unique operation whose own nonlinearity couples different bit positions: a high output bit depends on all lower input bits, through the carry chain. (Ch and Maj are nonlinear but bit-local; they inject no carry.) The round function is a bijection on the state (SHACAL-2, invertible given the message word), so the adder destroys no information. What it removes is *locality*: the high bits of a sum cannot be read without resolving the entire low-side carry chain.

5.3 The carry Markov chain

Exploitable structure contracts with carry distance rather than merely becoming unknown, and under an idealizing assumption the contraction has a precise rate. For a single adder with operand bits a_i, b_i *uniform and independent*, the carry is a two-state Markov chain $\gamma_{i+1} = \mathrm{MAJ}(a_i, b_i, \gamma_i)$ with

$$\Pr[\gamma_{i+1}{=}1 \mid \gamma_i{=}0] = \tfrac{1}{4}, \qquad \Pr[\gamma_{i+1}{=}1 \mid \gamma_i{=}1] = \tfrac{3}{4},$$

i.e. transition matrix $\left(\begin{smallmatrix} 3/4 & 1/4 \\ 1/4 & 3/4 \end{smallmatrix}\right)$, stationary $(\frac{1}{2}, \frac{1}{2})$, second eigenvalue $\frac{1}{2}$. Starting from the known $\gamma_0 = 0$, the bias decays geometrically,

$$\Pr[\gamma_i = 0] = \tfrac{1}{2} + 2^{-(i+1)}.$$

The carry-in becomes a fair coin $\sim i$ positions above the anchor, and chaining adders composes these contractions, which suggests that exploitable advantage decays as roughly $2^{-\Theta(\text{carry depth})}$, where carry depth counts the modular-add layers between the bits an attacker controls and the target bit. The uniform-independent assumption does not hold inside SHA-256, where state words are structured and correlated, so this is a heuristic and a prediction to test rather than a theorem; Section 5.5 measures the actual decay. This carry recurrence is precisely the finite state of the addition viewed as an S-function [18], the modern framework for ARX differential analysis [7], in which difference propagation through modular addition is governed entirely by the carry state machine.

5.4 Depth accounting

Carry depth is the structural coordinate of the proof-of-work function space: a two-round reduced hash is solvable because its input-to-target depth is $\mathcal{O}(1)$, whereas full SHA-256d sits hundreds of layers past that. We compute the coordinate exactly by static dataflow analysis (no hash evaluation; see `repro/carry_depth.py`). We treat the block-1 message words as the source inputs (the miner controls them through the nonce, the extranonce-driven Merkle root, the version, and the timestamp), use the minimum-depth associative tree for each multi-operand sum (conservative: it credits the attacker with the shallowest legal circuit), and count Ch/Maj as depth 0 (they inject no carry, so the figure is a *floor* on the nonlinear separation, ignoring their additional bit-local nonlinearity). The figure is convention-dependent: a sequential or nonce-only accounting gives a somewhat larger number, so the minimum-depth value below is the conservative choice.

quantity	value
one compression (64 rounds + feed-forward)	194 adder layers
SHA-256d input→output adder depth (min-tree)	**386**
per-round increment	≈ 3 layers
modular adds on the path	1200
carry bits (≈ 31 per 32-bit add)	$\approx 37{,}200$

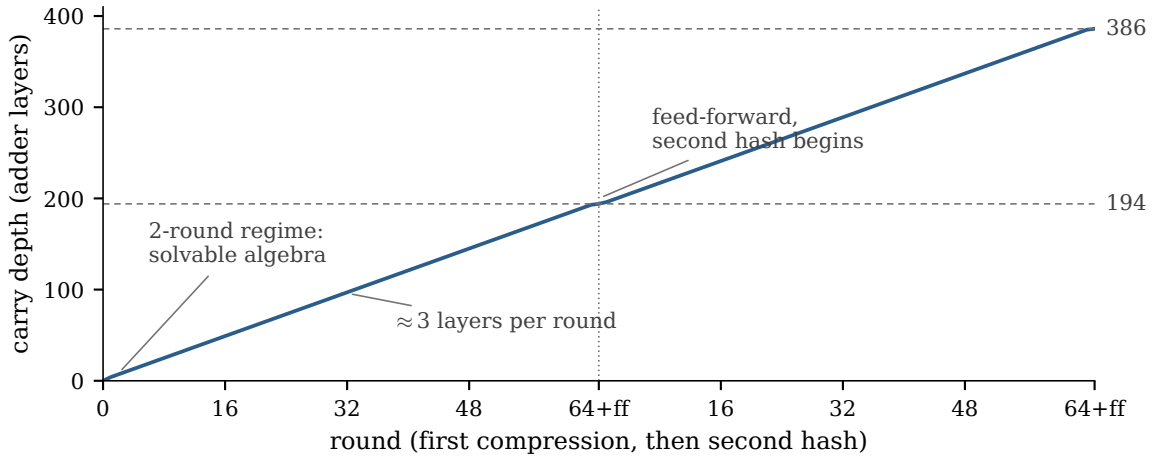

Figure 1: Carry depth along the SHA-256d dataflow, from the minimum-depth accounting of `repro/carry_depth.py`. The 64 rounds of the first compression lift the state by roughly three adder layers per round, each Davies–Meyer feed-forward adds a full-width carry layer, and the second hash repeats the climb to 386. The solvable two-round regime sits at the origin.

The two feed-forwards and the second hash each act as an independent carry reset (a fresh full-width operand), which is why they contribute more separation than their round count alone would suggest; Figure 1 traces the profile.

5.5 Measuring the per-layer retention: the anchor sweep

The depth fixes the exponent's base count (386); the margin is $386 \cdot \log_2(1/\rho)$, where ρ is the advantage retained per adder layer. We measure ρ directly on real reduced SHA-256 (see `repro/anchor_sweep.py`). The carry-aware score $(T_1 \gg 24) + (T_2 \gg 24)$, read at an interior round, strongly predicts that the top byte of `state`[0] is small one layer downstream; we measure how that control decays as the observed target moves deeper. Over 24 message stems (header-prefix stand-ins; see the code for the simplified setup) and 2^{22} candidates each (standard error on the mean advantage $\approx 10^{-3}$):

depth from read point	retained advantage
1 adder layer (same round)	0.8856 ± 0.0005
one round deeper (≈ 3 layers)	≤ 0.0011 (consistent with 0)

The advantage *cliffs* (Figure 2): it falls from 0.886 to undetectable within a single round. There is no geometric tail to fit; $\rho \approx 0$ (a one-layer reset). The mechanism is explicit: at the next round `state`[0] enters Σ_0, whose rotations scatter the controlled top byte across all bit positions, and is then re-randomized by a fresh carry chain, so the controlled structure is delocalized off the top-byte observable in one round. Information is preserved (the round is invertible); accessibility is destroyed.

5.6 The combined statement

The target output sits 386 adder layers from the controllable header inputs, while the strongest local advantage we measured has reach of about one layer and is dead within a single round (≈ 3 layers). The margin therefore does not rest on a thin ρ^{386} extrapolation; the advantage is already

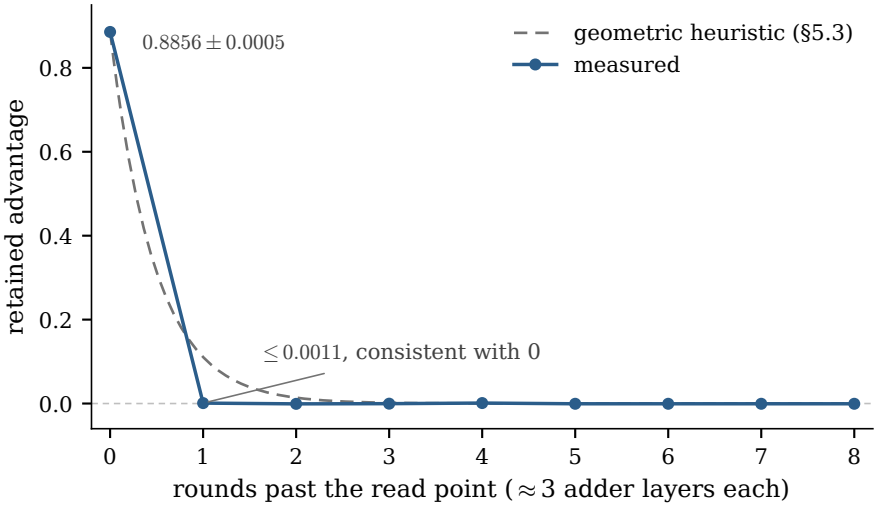

Figure 2: The anchor sweep (Section 5.5): selection advantage of the carry-aware score read j rounds past the read point, against the geometric decay suggested by the carry Markov heuristic of Section 5.3. The measurement returns a cliff rather than a slope; past one round every value sits within the $\approx 10^{-3}$ detection limit of zero.

gone roughly 100 rounds before the output. We deliberately do not compound the measured one-sided bound $\rho_{\text{round}} \leq 0.0011$ (a 2σ upper bound on the deeper-round mean advantage) across the remaining rounds: that bound is a detection limit consistent with $\rho = 0$, not a measured decay rate, and multiplying it would manufacture a precise-looking exponent from what is in fact a null result. The honest statement needs no extrapolation—the advantage is already below detection after a single round, and SHA-256d applies 128 rounds.

Relation to circuit-complexity measures. Carry depth should not be conflated with the circuit measures it superficially resembles. The closest prior notion is the carry-truncation depth of Field and Jones [19], who independently identify the carry as the sole nonlinearity of ARX; their depth is a local approximation parameter on a single addition, whereas carry depth is an exact, global count of carry layers along the input-to-output dataflow of the whole function. It is likewise distinct from the FHE/MPC notion of *multiplicative* (AND) *depth*: the standard Bristol-Fashion circuit for one SHA-256 compression has AND-depth 1607 [22, 21], far above our 386, because AND-depth counts every nonlinear gate on the path—including the bit-local Ch/Maj gates and every gate of a ripple-carry adder—while carry depth counts only carry layers and assigns Ch/Maj depth 0. Crucially, AND-depth is a property of a chosen circuit: the same modular addition has linear AND-depth as ripple-carry but logarithmic depth as a parallel-prefix adder [23], since the carry-merge operator is associative, and published heuristics shrink a 128-bit adder from AND-depth 255 to 9 without altering its logic [21]. Carry depth, by contrast, counts the logical carry-dependency layers and is invariant to the realization—it is a coordinate of the function, not of an implementation. Nor is it multiplicative *complexity*, a gate *count* (an n-bit adder needs exactly $n-1$ AND gates [20]) carrying no notion of depth.

6 Scope and the residual

Every statement in Section 5 is conditional on the attack belonging to the *carry / local-structure class*. This is the natural class, since carries are the only position-coupling nonlinearity. The carry-aware score we measured is a strong, natural member of it—a single observable read by a single score at a single interior round—and its reach is about one layer, dead within a single round; we do not claim to have exhausted the class, and a broader sweep over observables, scores, and read points is the obvious next test. The analysis does not bound a hypothetical local attacker with longer reach; that possibility is the same residual isolated by Theorem 2.

The residual is a single, well-defined object: *a global algebraic shortcut that resolves the target output without resolving the carries on the path.* By the standard-model reduction, such a shortcut is a distinguisher against SHA-256, which is to say a cryptographic break. We cannot prove it does not exist; as discussed in Section 4, an unconditional exclusion would require explicit circuit lower bounds beyond current techniques. The public cryptanalytic record of SHA-256 is the empirical statement that no such shortcut has been found.

The honest ceiling is therefore that mining cannot beat brute force unless SHA-256 is broken. We have bounded the search unconditionally in the ideal model, reduced the standard-model claim to a SHA-256 distinguisher, explained why the natural attack class has no reach, and quantified that the structural separation (386 layers) is far larger than the measured reach (~ 1 layer).

Empirical consistency. Independent of the theory, we ran the natural attack families against SHA-256d targets and found no advantage, exactly as the theory predicts: selection by any function of the first-hash digest's lexicographic order (an oracle for the smallest first-hash digest produces no signal at the SHA-256d output, because the second hash decorrelates input order from output order); shallow round-internal carry, Boolean, and message-schedule features under pre-registered, multiple-testing-corrected tail-hit gates; and per-block-template stem selection under an extreme-value yield model. To check that the cliff is not an artifact of a weak hand-built score, we also trained a neural distinguisher in the style of Gohr's attack on round-reduced ARX [28]: a residual network given, as features, everything computed through the read round—the state words, their carry-free derivations, the round's modular sums, and the schedule words, in bit, value, and Fourier encodings—and free to learn any function of them. Although it *exceeds* the hand-built carry-aware score one adder layer downstream (0.998 versus 0.88 retained advantage), its advantage collapses to the noise floor one round deeper, across independent stems and at a shifted read point, and remains null under a $90\times$ model-capacity, $64\times$ training-data, and $8\times$ training-time scaling of the learned attacker (each scaled run compared against its own shuffled-label noise ceiling)—the same single-round cliff, now traced by a strictly stronger member of the attack class (Figure 3a). Control runs locate the failure at the mechanism itself: the same architecture learns the top byte of pure k-operand modular sums for $k \leq 3$ and fails for $k \geq 4$ (Figure 3b), so the learned attacker expires on carry composition as such, on data with no SHA-256 structure at all. The same carry-depth mechanism that bounds the attack class also *retrodicts* the qualitative behaviour observed in those experiments, in particular the round-invariance of the local advantage (it is always a depth-one quantity) and its destruction by the chaining-variable addition (a single extra carry layer).

7 Reproduction

The two quantitative results are reproducible from self-contained code in `repro/`, built on a vendored minimal SHA-256 verified bit-exact against `hashlib`:

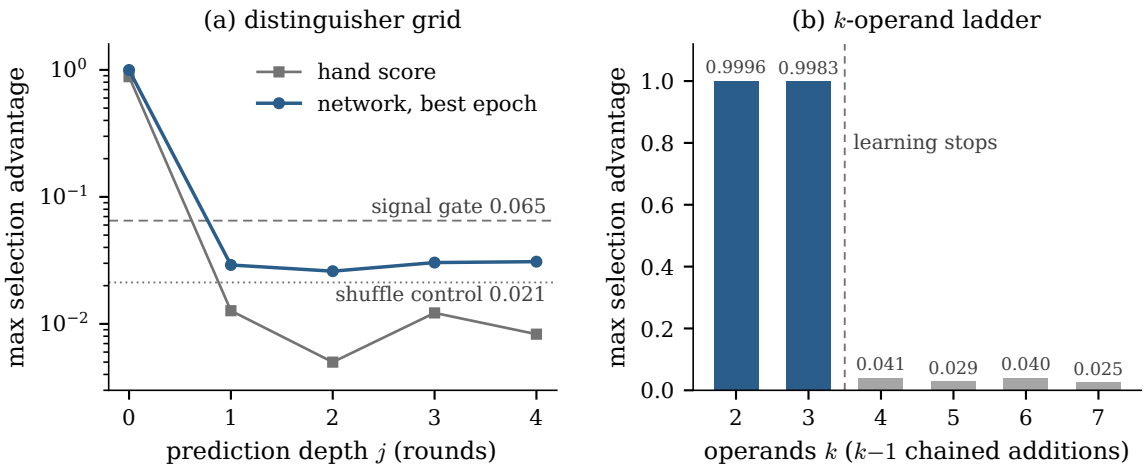

Figure 3: The learned attacker. (a) Maximum selection advantage of the neural distinguisher and the hand-built score across prediction depths, against the pre-registered candidate-signal gate and the shuffled-label control: the network beats the hand score where signal exists ($j = 0$) and joins the noise floor one round deeper. (b) The same architecture on pure k-operand modular sums, no SHA-256 structure present: two chained additions are learnable, three are not.

- `carry_depth.py`, the exact 386-layer dataflow accounting (pure Python, no dependencies, instant);
- `anchor_sweep.py`, the per-layer advantage measurement (0.886 at depth one, consistent with 0 one round deeper); the full run uses `numpy`/`cupy`, and a `--pure` standard-library-only smoke path reproduces the cliff (and a `hashlib` parity check) in seconds on any interpreter;
- `sha256_ref.py`, the vendored SHA-256 and its parity check.

The neural-distinguisher experiment of Section 6 is likewise reproducible, from `neural/` (pre-registered protocol with documented amendments, training code, and raw per-cell results); unlike the two results above it requires a GPU and the JAX/optax stack.

8 Related work and conclusion

Proof-of-work security is conventionally proved in the random-oracle model [1], with standard-model [10] and post-quantum [11, 12] analyses extending it. As we make explicit, the impossibility of beating brute force is not provable unconditionally in the standard model without resolving open hardness questions. ASICBoost [2] is the canonical input-side win, and it occupies the only shareable region we identify. The role of modular addition as the nonlinear primitive of ARX constructions is well studied [7, 18], and the carry as its sole nonlinearity has been observed before [19]; our contribution is to turn this into *carry depth*, an explicit, circuit-independent coordinate of the proof-of-work function space, to compute it exactly for SHA-256d, and to measure that the strongest local advantage has reach of about one layer (dead within a single round), so that the 386-layer separation is decisive rather than marginal. Auxiliary-input bounds [4, 5, 3, 13] and the BBBV quantum lower bound [6] close the preprocessing and quantum loopholes. The public cryptanalytic record is consistent with this ceiling: reduced-round collision attacks reach only 31 of 64 steps (made

practical in [24], building on [25]) and 39 steps semi-free-start [24], and preimages reach 43 steps at cost $2^{254.9}$ [26], all far from the full function.

SHA-256d mining cannot beat brute force by any route we or the cryptanalytic record can find, and cannot at all unless SHA-256 is broken. The only standing economic wins are the known constant-factor sharing techniques, and they are bounded.

References

[1] J. Garay, A. Kiayias, N. Leonardos. *The Bitcoin Backbone Protocol: Analysis and Applications.* EUROCRYPT 2015.

[2] T. Hanke. *AsicBoost: A Speedup for Bitcoin Mining.* 2016, `arXiv:1604.00575`.

[3] D. Unruh. *Random Oracles and Auxiliary Input.* CRYPTO 2007.

[4] S. Coretti, Y. Dodis, S. Guo, J. Steinberger. *Random Oracles and Non-Uniformity.* EUROCRYPT 2018.

[5] S. Coretti, Y. Dodis, S. Guo. *Non-Uniform Bounds in the Random-Permutation, Ideal-Cipher, and Generic-Group Models.* CRYPTO 2018.

[6] C. Bennett, E. Bernstein, G. Brassard, U. Vazirani. *Strengths and Weaknesses of Quantum Computing.* SIAM J. Comput. 26(5), 1997.

[7] H. Lipmaa, S. Moriai. *Efficient Algorithms for Computing Differential Properties of Addition.* FSE 2001.

[8] J.-S. Coron, Y. Dodis, C. Malinaud, P. Puniya. *Merkle-Damgård Revisited: How to Construct a Hash Function.* CRYPTO 2005.

[9] Y. Dodis, T. Ristenpart, J. Steinberger, S. Tessaro. *To Hash or Not to Hash Again? (In)differentiability Results for H^2 and HMAC.* CRYPTO 2012.

[10] J. Garay, A. Kiayias, G. Panagiotakos. *Proofs of Work for Blockchain Protocols.* IACR Cryptology ePrint Archive, Report 2017/775, 2017.

[11] A. Cojocaru, J. Garay, A. Kiayias, F. Song, P. Wallden. *Quantum Multi-Solution Bernoulli Search with Applications to Bitcoin's Post-Quantum Security.* Quantum 7:944, 2023.

[12] A. Cojocaru, J. Garay, F. Song. *Generalized Hybrid Search with Applications to Blockchains and Hash Function Security.* ASIACRYPT 2024.

[13] Y. Dodis, S. Guo, J. Katz. *Fixing Cracks in the Concrete: Random Oracles with Auxiliary Input, Revisited.* EUROCRYPT 2017.

[14] M. E. Hellman. *A Cryptanalytic Time-Memory Trade-Off.* IEEE Trans. Inf. Theory 26(4), 1980.

[15] P. Oechslin. *Making a Faster Cryptanalytic Time-Memory Trade-Off.* CRYPTO 2003.

[16] U. Maurer, R. Renner, C. Holenstein. *Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology.* TCC 2004.

[17] T. Ristenpart, H. Shacham, T. Shrimpton. *Careful with Composition: Limitations of the Indifferentiability Framework.* EUROCRYPT 2011.

[18] N. Mouha, V. Velichkov, C. De Cannière, B. Preneel. *The Differential Analysis of S-functions.* SAC 2010.

[19] R. E. Field, B. C. Jones. *Using Carry-Truncated Addition to Analyze Add-Rotate-Xor Hash Algorithms.* 2013, `arXiv:1303.4448`.

[20] J. Boyar, R. Peralta. *Tight Bounds for the Multiplicative Complexity of Symmetric Functions.* Theoretical Computer Science 396, 2008.

[21] P. Aubry, S. Carpov, R. Sirdey. *Faster Homomorphic Encryption is not Enough: Improved Heuristic for Multiplicative Depth Minimization of Boolean Circuits.* CT-RSA 2020.

[22] N. P. Smart et al. *'Bristol Fashion' MPC Circuits.* `https://nigelsmart.github.io/MPC-Circuits/`.

[23] T. G. Draper, S. A. Kutin, E. M. Rains, K. M. Svore. *A Logarithmic-Depth Quantum Carry-Lookahead Adder.* Quantum Inf. Comput. 6(4), 2006.

[24] Y. Li, F. Liu, G. Wang. *New Records in Collision Attacks on SHA-2.* EUROCRYPT 2024.

[25] F. Mendel, T. Nad, M. Schläffer. *Improving Local Collisions: New Attacks on Reduced SHA-256.* EUROCRYPT 2013.

[26] K. Aoki, J. Guo, K. Matusiewicz, Y. Sasaki, L. Wang. *Preimages for Step-Reduced SHA-2.* ASIACRYPT 2009.

[27] *nVersion Bits for General Purpose Use (BIP 320).* Bitcoin Improvement Proposals, 2018.

[28] A. Gohr. *Improving Attacks on Round-Reduced Speck32/64 Using Deep Learning.* CRYPTO 2019.