Collatz-PoW: A Novel Proof-of-Work System Based on the Collatz Conjecture

Ngoc Anh Khoa Doan*

September 2025

Abstract

Proof-of-Work (PoW) consensus mechanisms, while foundational to cryptocurrencies like Bitcoin, face persistent challenges of extreme energy consumption and mining centralization driven by Application-Specific Integrated Circuits (ASICs). This paper introduces Collatz-PoW, a novel PoW algorithm that leverages the computational properties of the Collatz conjecture. In our system, miners compete to find a nonce that, when combined with the block header and hashed, produces a starting integer whose Collatz sequence converges to 1 in a precisely defined target number of steps.

We argue that the algorithm's reliance on sequential, conditional arithmetic (integer division and multiplication) presents a structural barrier to the massive parallelization that fuels ASIC dominance. This paper makes three primary contributions. First, we provide a formal definition of the Collatz-PoW problem. Second, we present a complete, open-source Python prototype of a blockchain that implements this consensus mechanism, including a balance-based transaction model. Third, we introduce and validate a key innovation—deriving the Collatz input from the block header hash—to prevent strategic mining and precomputation attacks, a critical flaw in more naive theoretical proposals. Experimental results from our prototype demonstrate the system's viability, including fast block verification ($< 0.5ms$) and an exponential relationship between target steps and mining difficulty. This work moves beyond theoretical discussion to provide a practical foundation for a potentially more decentralized and energy-efficient consensus alternative.

Keywords: Blockchain, Proof-of-Work, Consensus Algorithm, Collatz Conjecture, ASIC Resistance, Cryptocurrency

1 Introduction

Since its inception, blockchain technology has been synonymous with its pioneering consensus mechanism, Proof-of-Work (PoW), as implemented in Bitcoin [1]. PoW provides a robust solution to the Byzantine Generals' Problem, enabling decentralized and trustless agreement in a distributed system. However, the success and proliferation of PoW-based

*Independent Researcher, contact: anhkhoadoanngoc19@gmail.com

cryptocurrencies have exposed two fundamental weaknesses: staggering energy consumption, with the Bitcoin network alone consuming energy comparable to entire nations [2], and a relentless trend toward mining centralization. This centralization is a direct consequence of the "arms race" for specialized hardware, specifically ASICs, which offer orders-of-magnitude performance gains over general-purpose CPUs and GPUs for specific hashing algorithms like SHA-256 [5].

This has led to a landscape where a few large mining pools control the majority of network hashrate, undermining the core principle of decentralization. In response, the research community has explored a plethora of alternative consensus mechanisms. Proof-of-Stake (PoS) and its variants have emerged as a popular energy-efficient alternative, but they introduce different sets of economic assumptions and potential centralization vectors. Within the PoW paradigm, efforts to counter ASICs have primarily focused on memory-hard functions like Scrypt [3] and Ethash [4]. While initially successful, these algorithms eventually saw the development of optimized ASICs, suggesting that memory-hardness alone may not be a durable solution.

This paper explores a different path toward ASIC resistance: computational irregularity. We introduce *Collatz-PoW*, a novel PoW system based on the famous and unsolved Collatz conjecture [6]. The core of the work is a search problem: miners must find a nonce that leads to a Collatz sequence of a specific length. The computation of a Collatz sequence is inherently sequential and relies on conditional branching—operations that are notoriously difficult to pipeline and parallelize efficiently on the vast scale required for ASIC dominance.

Our primary contributions are:

1. **A Formal Definition of Collatz-PoW:** We formally define the computational problem, its parameters (target steps, max value limit), and the verification function, establishing a clear basis for security analysis.

2. **A Complete Prototype Implementation:** We developed and open-sourced a full blockchain prototype in Python, featuring transactions, a balance-based ledger, block creation, and chain validation. This moves the concept from theory to a tangible, testable artifact.

3. **A Practical Solution to Strategic Mining:** We solve a key issue in prior theoretical works [7] by deriving the Collatz input from a hash of the nonce-inclusive block header. This transforms the search into a pseudo-random process, neutralizing vulnerabilities like precomputation attacks and strategic mining of specific integer ranges.

4. **Experimental Evaluation:** We present results from our prototype that validate the system's core properties, including fast verification times and a clear, tunable relationship between the difficulty parameter and the expected mining time.

This paper is structured as follows. Section 2 provides background on the Collatz conjecture and PoW. Section 3 reviews related work. Section 4 details the system design and formally defines Collatz-PoW. Section 5 describes our prototype implementation. Section 6 presents our experimental results. Section 7 provides a thorough security analysis. Section 8 discusses the implications and limitations of our work, and Section 9 concludes.

2 Preliminaries

2.1 The Collatz Conjecture

The Collatz conjecture, also known as the $3n + 1$ problem, is a famous unsolved problem in mathematics. It concerns sequences generated from a positive integer n by the following rule:

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ is even,} \\ 3n+1 & \text{if } n \text{ is odd.} \end{cases}$$

The conjecture states that for any starting integer $n > 0$, the sequence of iterates $n, f(n), f(f(n)), \ldots$ will eventually reach the number 1. While this has been verified computationally for all integers up to 2^{68} [12], a formal proof remains elusive. The *stopping time* or *total stopping time* of n is the number of steps required for the sequence to reach 1. For example, for $n = 7$, the sequence is $7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1$, which has a stopping time of 16 steps.

2.2 Proof-of-Work Consensus

Proof-of-Work is a mechanism that requires a party to perform a moderately hard, yet feasible, amount of computational work, the result of which is easy for others to verify. In the context of a blockchain, this work secures the network. Bitcoin's PoW requires miners to find a nonce such that the SHA-256 hash of the block header is less than a given target value:

$$\text{SHA-256}(\text{SHA-256}(\text{block\_header}||\text{nonce})) < \text{target}$$

The difficulty of this problem is adjusted by changing the target value. A valid PoW proves that a miner has expended significant computational effort, thereby earning the right to add a block to the chain.

3 Related Work

Our work builds upon research in alternative PoW algorithms and previous theoretical explorations of the Collatz conjecture in cryptography. We categorize related work into three areas.

ASIC-Resistant PoW. The primary motivation for alternative PoWs has been ASIC resistance. **Scrypt** [3], used by Litecoin, was designed as a memory-hard function, requiring a large amount of RAM to compute efficiently, thereby increasing the cost of specialized hardware. **Ethash** [4], formerly used by Ethereum, was also designed to be memory-hard by requiring miners to read from a large, pseudo-randomly generated dataset called the DAG. Despite these efforts, ASICs were eventually developed for both algorithms [5], demonstrating the difficulty of achieving long-term ASIC resistance through memory hardness alone. Cuckoo Cycle [13] is another memory-hard PoW based on finding cycles in a bipartite graph.

Useful Proof-of-Work (uPoW). Another research direction aims to make the work performed by miners useful for other purposes. **Primecoin** [9] requires miners to find special chains of prime numbers (Cunningham chains and bi-twin chains), contributing to number theory research. **Gridcoin** [10] rewards users for contributing computation to scientific research projects through the Berkeley Open Infrastructure for Network Computing (BOINC) platform. While appealing, uPoW systems often face challenges in ensuring that the work is both useful, difficult to centralize, and easy to verify quickly within a decentralized context.

Collatz-Based and Mathematical PoWs. The Collatz conjecture's chaotic behavior has made it a topic of interest for cryptographic applications. Some have explored its use for chaotic hash functions or random number generators [11]. The idea of a Collatz-based PoW was first detailed by González [7], who proposed a system based on the "inflation propensity" of sequences and performed numerical simulations. However, this work did not present a full blockchain implementation and noted the challenge of high variance in mining times. More recently, Singh et al. [8] described a theoretical "Proof-of-Collatz" protocol, focusing on its potential for quantum resistance due to its non-algebraic structure. Our work distinguishes itself from these theoretical proposals by providing the first, to our knowledge, complete and functional prototype of a Collatz-PoW blockchain, with a specific mechanism to address the critical issue of strategic mining.

4 System Design and Formalism

The Collatz-PoW blockchain is designed to be a transparent and secure ledger, with its consensus mechanism serving as the core innovation.

4.1 Architecture Overview

The system follows a standard blockchain architecture: a peer-to-peer network of nodes maintains a replicated, append-only ledger of transaction blocks. Each block is cryptographically linked to its predecessor. Consensus on the canonical chain is achieved via the longest-chain rule, where the valid chain with the most cumulative work is accepted by all honest nodes. The "work" is defined by our Collatz-PoW algorithm.

4.2 Data Structures

Transactions. A transaction is a simple data structure representing the transfer of value. It contains the sender's address, recipient's address, amount, and a fee for the miner. For our prototype, we use a balance-based model where the blockchain object tracks the balance of each address to validate spending.

Blocks. A block consists of a header and a list of transactions. The block header is the critical component for the PoW. Its fields are detailed in Table 1.

Table 1: Structure of the CollatzBlock Header

Field	**Description**
Previous Hash	SHA-256 hash of the preceding block's header.
Timestamp	Unix epoch time of the block's creation.
Merkle Root	The root of the Merkle tree of all transactions in the block.
Miner Address	The address to receive the block reward and transaction fees.
Target Steps	The required stopping time for the Collatz sequence (difficulty parameter).
Max Value Limit	An upper bound on any number in the sequence (e.g., 2^{64}).
Nonce	An arbitrary 32-bit integer, iterated by the miner during the PoW search.
Collatz Input (n)	The integer derived from the header hash, used as the Collatz seed.
Collatz Steps	The actual stopping time of the sequence for the solution n.
Collatz Max	The maximum value reached in the sequence for the solution n.

4.3 Formal Definition of Collatz-PoW

We formally define the Collatz Proof-of-Work problem as follows.

Let $H : \{0,1\}^* \to \{0,1\}^{256}$ be the SHA-256 hash function. Let $f(n)$ be the Collatz function. Let $S(n) = (n, f(n), f(f(n)), \ldots, 1)$ be the Collatz sequence starting from n. Let $|S(n)|$ be the stopping time (number of steps) and $\max(S(n))$ be the maximum value in the sequence.

A Collatz-PoW **challenge** is a tuple $C = (B_{hdr\_prefix}, T, M)$, where:

- B_{hdr_prefix} is the block header data excluding the nonce.
- $T \in \mathbb{N}^+$ is the target number of steps.
- $M \in \mathbb{N}^+$ is the maximum value limit.

A **solution** to the challenge C is a nonce $\eta \in \mathbb{N}$ such that if we let $n = \text{int}(H(B_{hdr\_prefix}||\eta))$, the following conditions hold:

1. $|S(n)| = T$
2. $\max(S(n)) \leq M$

The **verification** function $V(C, \eta) \to \{\text{True, False}\}$ simply re-computes n from the challenge data and the provided nonce η, generates the Collatz sequence $S(n)$, and checks if the two conditions are met.

4.4 Mining Process and Variance Mitigation

The mining process in Collatz-PoW constitutes a brute-force search for a valid nonce η that satisfies the conditions defined by the PoW challenge. A miner begins by constructing a block template, populating it with transactions from their mempool and establishing a coinbase transaction to claim the block reward. They then enter a loop, iterating through values of the nonce η (e.g., $0, 1, 2, \ldots$). For each candidate nonce, the miner computes the corresponding Collatz input integer n and generates its sequence to determine if it is a valid solution. While the search over the nonce space is an "embarrassingly parallel" problem—allowing a miner to use multiple threads or machines to check different nonces simultaneously—the core computation for each nonce remains sequential.

4.4.1 Vulnerabilities of a Naive Approach

A critical design choice in any PoW system is how the iterated nonce influences the problem to be solved. A naive implementation, where the nonce itself is used as the Collatz input (i.e., $n = \eta$), would introduce severe security vulnerabilities and undermine the fairness of the consensus mechanism. These vulnerabilities stem directly from the known mathematical properties of the Collatz conjecture.

Strategic Mining and Centralization Risk. The distribution of Collatz stopping times is highly non-uniform and chaotic. Certain integers, particularly smaller ones, are known to have specific stopping times that are relatively common. A naive $n = \eta$ mapping would create a static and predictable relationship between the nonce and the sequence properties. This would allow sophisticated miners to engage in *strategic mining*. Instead of a random search, they could focus their computational resources on "fertile" regions of the nonce space that are statistically more likely to yield solutions for a given target. This creates an unfair advantage, leading to mining centralization around those with the knowledge and capability to exploit these statistical biases.

Precomputation and Time-Memory Trade-Off Attacks. Furthermore, a static mapping would make the system vulnerable to precomputation attacks. An attacker could dedicate significant resources to pre-calculating and storing a large database of Collatz sequences for a vast range of integers. When a new block with a specific target T is announced, the attacker could consult their database to find pre-computed solutions, gaining a significant head start over honest miners who must perform the computation in real-time.

4.4.2 The Hashing-Based Mitigation Strategy

Our solution, illustrated in Figure 1, is designed to neutralize these attack vectors by introducing a layer of cryptographic randomization between the nonce and the Collatz input. The input n is derived from the SHA-256 hash of the entire block header prefix concatenated with the current nonce. This design has two profound benefits that are fundamental to the protocol's security and fairness.

1. **Pseudo-randomization of the Search Space:** The primary benefit is the transformation of a predictable search into a pseudo-random one. Cryptographic hash functions like SHA-256 exhibit a strong *avalanche effect*, where a single-bit change in the input (such as incrementing the nonce) causes a cascade of changes, resulting in an output that is computationally indistinguishable from a random string. This means that a small, linear change in the nonce ($\eta \to \eta + 1$) produces a large, unpredictable leap in the resulting Collatz input n. This effectively "flattens" the search space, forcing all miners to sample from a wide, pseudo-random distribution of large integers. It neutralizes strategic mining, as there is no longer a predictable correlation between the nonce being tested and the properties of the resulting Collatz sequence.

2. **Commitment to Block Content:** The second key benefit is that the PoW acts as a binding commitment to the exact contents of the block. Because the Merkle root is included in the data that gets hashed, the resulting Collatz input n is dependent on every single transaction within the block. If a miner were to find a valid nonce and then attempt to alter the block's contents—for instance, by removing a transaction or redirecting a fee to themselves—the Merkle root would change. This change would, in turn, completely alter the hash and the derived Collatz input n, thus invalidating the previously found PoW solution. The work is therefore inextricably bound to a specific, immutable set of transactions, ensuring the integrity of the block and the ledger.

This hashing-based approach ensures that the only viable strategy for any miner is a brute-force search, fulfilling the foundational requirement of a fair and decentralized Proof-of-Work system.

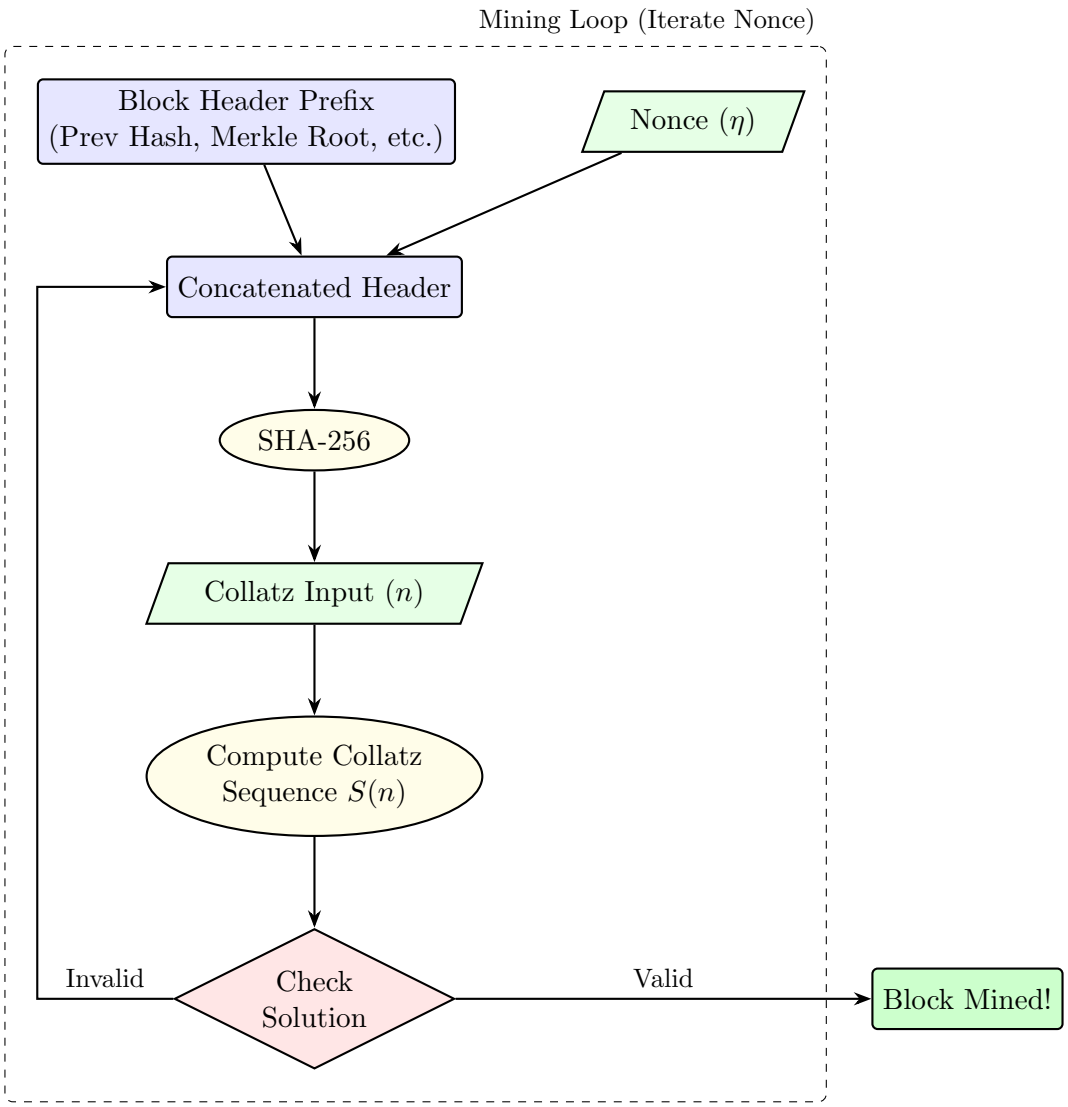

Figure 1: The Collatz-PoW Mining Process, Illustrating Strategic Mining Mitigation.

5 Prototype Implementation

To validate our design, we developed a full-featured prototype of the Collatz-PoW blockchain in Python. The implementation prioritizes clarity and correctness, utilizing standard libraries for hashing and data structures. The code is open-source and available for community review.

The prototype consists of three main classes: ‘Transaction‘, ‘CollatzBlock‘, and ‘Blockchain‘. The ‘Blockchain‘ class orchestrates the system, managing the chain, a mempool for pending transactions, and a dictionary for tracking address balances. The core mining logic is encapsulated in a ‘mine_block‘ function, as shown in the abstract algorithm in Listing 1.

```python
def mine_block(prev_hash, txs, target_steps, miner_addr):
  # Create a block template
  temp_block = create_block_template(prev_hash, txs, miner_addr,
     target_steps)

  # Loop through nonces to find a solution
  for nonce in range(MAX_NONCE):
    # Derive Collatz input from the hash of the nonce-inclusive
       header
    header_bytes = temp_block.header_for_hashing(nonce)
    hash_result = hashlib.sha256(header_bytes).digest()
    n = int.from_bytes(hash_result, 'big')

    # Compute the sequence and check if it's a valid solution
    result = collatz_sequence_info(n, max_steps=target_steps + 100)

    if result is not None:
      steps, max_val, _ = result
      if steps == target_steps and max_val <= MAX_VALUE_LIMIT:
        # Solution found, create and return the final block
        return finalize_block(temp_block, nonce, n, steps, max_val)

  return None # Mining failed
```

Listing 1: Abstract Python Algorithm for Collatz-PoW Mining

A key implementation detail is the overflow protection within the 'collatz_sequence_info' function. To prevent denial-of-service attacks where a crafted block could lead to an astronomically large number, the computation is halted if an intermediate value exceeds a safe threshold (e.g., 2^{256}). This, combined with the 'max_value_limit' check, ensures computational stability.

6 Experimental Evaluation

We conducted a series of experiments using our Python prototype to evaluate the performance and characteristics of the Collatz-PoW system. All tests were performed on a single thread of a standard consumer CPU (Intel Core i7-8700K @ 3.70GHz).

6.1 Verification Performance

The efficiency of block verification is critical for network scalability, as all full nodes must perform this task. We measured the time required to verify a block, which primarily consists of re-computing the Collatz sequence for the given 'collatz_n' input. As shown in Table 2, the time to verify the PoW remains extremely low across a range of difficulties, consistently staying below 0.5 ms. This is orders of magnitude faster than a single SHA-256

hash computation and demonstrates that the PoW check itself is not a bottleneck for node performance.

Table 2: Block Verification Time vs. Target Steps

Target Steps (T)	**Average Verification Time (ms)**
5	0.11
10	0.13
15	0.16
20	0.18
50	0.25
100	0.42

6.2 Mining Difficulty Analysis

The central property of any PoW algorithm is the ability to tune its difficulty. In Collatz-PoW, the ‘target_steps‘ parameter serves this function. We measured the average time to mine a block for various target step values. The results, plotted in Figure 2, show a clear, near-exponential relationship between the target steps and the mining time.

This demonstrates that the difficulty is highly tunable. A small increase in the target steps leads to a large increase in the expected number of nonces that must be tested. This is because sequences of a specific length become progressively rarer as the length increases. This property is essential for a blockchain's difficulty adjustment algorithm, which must be able to maintain a consistent block time as the network's total computational power changes.

6.3 Nonce Distribution

We analyzed the distribution of successful nonces found during a simulation of mining 100 blocks with a fixed target of 15 steps. The nonces were found to be distributed across a wide range, with no apparent clustering in the low-value range. This provides empirical support for our mitigation strategy; the hashing of the header effectively randomizes the search, preventing miners from gaining an advantage by focusing on a small subset of the nonce space.

7 Security Analysis

We analyze the security of Collatz-PoW with respect to established blockchain attack vectors.

Majority (51%) Attacks. Like all Nakamoto-style consensus protocols, Collatz-PoW is vulnerable to a 51% attack. An attacker controlling a majority of the network's computational power could theoretically produce a longer private chain and use it to double-spend transactions or censor other users. The security against this attack relies on the economic assumption that it is more profitable for a majority miner to behave honestly.

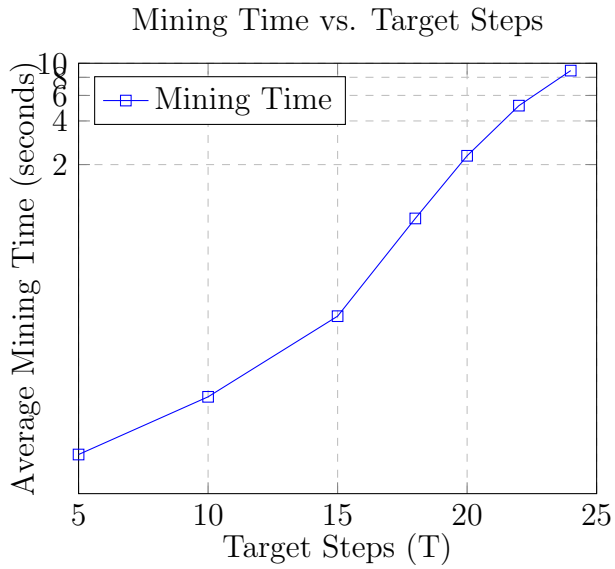

Figure 2: Relationship between the 'target_steps' difficulty parameter and the average time required to mine a block. The logarithmic scale on the y-axis highlights the exponential growth.

Precomputation and Time-Memory Trade-Off (TMTO) Attacks. An attacker might try to pre-compute Collatz sequences for a large number of integers to create a lookup table. Our hashing-based mitigation technique makes this attack infeasible. Because the Collatz input n depends on the 'previous_hash', 'merkle_root', and 'timestamp', it is unique for every block. An attacker cannot pre-compute solutions for future blocks without knowing their exact header contents.

Denial-of-Service via Sequence Growth. The Collatz conjecture is known for producing sequences that can grow to very large numbers before converging. An attacker could try to craft transactions that lead to a Merkle root which, when combined with a specific nonce, produces a starting number n with an exceptionally large intermediate value, potentially causing resource exhaustion on validating nodes. Our protocol mitigates this in two ways:

1. The 'max_value_limit' (2^{64} in our prototype) provides a hard cap. Any sequence exceeding this value is immediately invalid.

2. The internal computational limit (e.g., 2^{256}) prevents integer overflow during sequence calculation.

Mining Centralization and ASIC Resistance. The primary security claim of Collatz-PoW is its potential for ASIC resistance. This claim is based on the computational nature of the problem:

- **Conditional Branching:** The core 'if n % 2 == 0' check creates data-dependent branching, which is inefficient for highly parallel architectures like GPUs and ASICs.

These architectures excel when executing the same instruction on multiple data points simultaneously (SIMD), and branching causes thread divergence, reducing performance to near-CPU levels.

- **Integer Arithmetic:** The work involves basic integer operations (division, multiplication, addition), which are already highly optimized on general-purpose CPUs. There is less room for dramatic hardware-specific optimization compared to the complex bitwise operations of SHA-256.

While these properties strongly suggest ASIC resistance, a definitive proof requires dedicated hardware design and analysis. However, the fundamental structure of the problem is a significant departure from easily parallelizable hashing algorithms.

8 Discussion

Our experimental results and security analysis indicate that Collatz-PoW is a viable and compelling alternative to traditional PoW systems.

8.1 Summary of Advantages

- **Potential ASIC Resistance:** The algorithm's structure is inherently resistant to the parallelization strategies that enable ASIC dominance.
- **High Efficiency:** Block verification is extremely fast, reducing the burden on full nodes and improving network scalability.
- **Tunable Difficulty:** The 'target_steps' parameter provides a reliable and granular mechanism for adjusting mining difficulty.
- **Energy Efficiency:** While the "work" is still computationally intensive, the underlying arithmetic operations are simpler than cryptographic hashing, potentially leading to lower energy consumption per computation.

8.2 Limitations and Open Challenges

- **The Unproven Conjecture:** The protocol's liveness relies on the Collatz conjecture being true for the inputs generated. While the conjecture holds for all tested numbers up to a very high limit, a counterexample (an integer that generates an infinite sequence or a non-trivial cycle) would be a protocol-level risk. However, our use of a 'max_steps' limit in computation effectively prevents any single node from getting stuck on such a sequence, turning a catastrophic failure into a simple invalid nonce.
- **Mining Time Variance:** While our nonce-hashing mechanism ensures a fair search, the natural variance in finding a sequence of a precise length remains. This means that block times will have a wider distribution around the target compared to Bitcoin. A robust difficulty adjustment algorithm that averages over a large number of blocks (e.g., 2016 blocks) is essential to manage this.

- **Lack of Real-World Testing:** Our results are from a single-node prototype. The protocol's performance and behavior in a large-scale, adversarial peer-to-peer network are yet to be studied.

9 Conclusion

In this paper, we introduced Collatz-PoW, a novel Proof-of-Work system based on the Collatz conjecture. We moved beyond prior theoretical proposals by designing, formally defining, and implementing a complete blockchain prototype. Our key innovation, a nonce-hashing mechanism to derive the Collatz input, successfully prevents the strategic mining issues that challenged earlier concepts. Experimental evaluation of our prototype demonstrates that the system is functional, with extremely fast verification times ($< 0.5ms$) and a predictable, exponential relationship between its difficulty parameter and mining time.

The primary appeal of Collatz-PoW lies in its potential for ASIC resistance, stemming from its reliance on sequential, branch-heavy integer arithmetic—a computational profile ill-suited for the massive parallelization that underpins ASIC dominance. By presenting a viable and well-analyzed system, this work provides a concrete foundation for a new class of PoW algorithms that could lead to more decentralized, equitable, and potentially more energy-efficient blockchain networks. We have open-sourced our implementation to encourage further research, community testing, and collaborative development.

10 Future Work

While this paper establishes a strong foundation, significant engineering and research are required to prepare Collatz-PoW for real-world deployment. Our immediate priorities for future work include:

1. **Peer-to-Peer Network Implementation:** Developing a P2P networking layer is the most critical next step. This will allow for the creation of a public testnet where the protocol's behavior regarding block propagation, fork resolution, and latency can be studied under real-world conditions.

2. **Difficulty Adjustment Algorithm:** Designing and implementing a robust difficulty adjustment mechanism is essential for maintaining a stable block time. This will involve analyzing the statistical distribution of mining times and adapting Bitcoin's algorithm to account for the higher intrinsic variance of Collatz-PoW.

3. **Empirical ASIC Resistance Analysis:** A definitive assessment of ASIC resistance requires collaboration with hardware experts. This would involve designing a theoretical VHDL/Verilog implementation of a Collatz-PoW miner to estimate the performance gains achievable with an ASIC compared to a CPU or GPU.

4. **Cryptographic Hardening:** The prototype's transaction model will be enhanced with standard cryptographic primitives, including ECDSA for digital signatures to authenticate transactions and a full UTXO model for robust double-spend prevention.

We invite the academic and open-source communities to collaborate on these challenges.

References

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," White paper, 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] Digiconomist, "Bitcoin Energy Consumption Index," 2023. [Online]. Available: https://digiconomist.net/bitcoin-energy-consumption

[3] C. Percival, "Stronger Key Derivation via Sequential Memory-Hard Functions," in *BSDCan'09*, 2009.

[4] G. Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," Ethereum Project Yellow Paper, 2014.

[5] M. B. Taylor, "The Evolution of Bitcoin Hardware," *IEEE Computer*, vol. 50, no. 9, pp. 58-66, 2017.

[6] J. C. Lagarias, "The 3x+1 Problem and Its Generalizations," *American Mathematical Monthly*, vol. 92, no. 1, pp. 3-23, 1985.

[7] R. E. González, "Inflation Propensity of Collatz Orbits: A New Proof-of-Work for Blockchain Applications," *Journal of Risk and Financial Management*, vol. 11, no. 4, p. 73, 2018.

[8] S. Singh, A. Kumar, and P. Kumar, "Proof-of-Collatz: A Consensus Protocol Based on Collatz Conjecture," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 7, no. 1, 2025.

[9] S. King, "Primecoin: Cryptocurrency with Prime Number Proof-of-Work," White paper, 2013.

[10] Gridcoin, "Gridcoin: Rewarding Scientific Computation," White paper, 2014.

[11] Y. Cao, L. Wang, and J. Li, "Chaotic Hash Function Based on Collatz Conjecture and Logistic Map," *Chaos, Solitons & Fractals*, vol. 174, p. 113795, 2023.

[12] D. Barina, "Verification of the Collatz conjecture up to 2^{68}," *The Art of Computer Programming, Volume 4, Fascicle 7 (draft)*, 2021.

[13] J. Tromp, "Cuckoo Cycle: a memory-hard proof-of-work system," in *Financial Cryptography and Data Security Workshops*, 2015.