The Integrity of Proof of Work: Nonce Spamming

Toni Lukkaroinen[1]

[1] *Hoosat Network*

26 May 2025
Version 1.1

Abstract

Proof-of-Work (PoW), first introduced by Bitcoin[8], underpins the security and consensus of numerous decentralized blockchain systems by requiring demonstrable expenditure of computational resources for block creation.

In this work, we reveal a critical vulnerability in this assumption: miners can produce valid blocks without performing the expected local computation by exploiting the network protocols through *nonce spamming*. This technique involves broadcasting large volumes of candidate block headers with varied nonce values to pools and nodes, thereby offloading the hash computation and validation to other participants. If a valid nonce is found through this distributed probing, the attacker can claim the block with minimal local computation effort.

While feasible, this attack loses effectiveness as network difficulty and competition increase, shrinking the attacker's window of opportunity. This vulnerability challenges the assumption that every accepted block corresponds to a fair and provable investment of work, calling for stronger mechanisms to enforce localized computational effort and a reconsideration of trust models in PoW-based systems.

1 Introduction

Bitcoin [8] revolutionized digital trust by introducing Proof-of-Work (PoW), a decentralized consensus mechanism that secures blockchain networks through computational effort. In this system, miners race to find a special number, called a *nonce*, which, when combined with block data as input to a Proof-of-Work algorithm, produces a hash meeting a strict difficulty target. The miner who discovers this nonce first gains the right to add the next block to the blockchain and collect its rewards.

At the heart of PoW's security lies a critical assumption: miners must invest significant local computational power to find these valid nonce values. This effort ensures fairness and prevents malicious actors from cheaply rewriting history. Traditionally, miners submit their discovered solutions either directly to the network or through mining pools using protocols like Stratum [11].

However, in this work, we expose vulnerability in this model, a technique we call *nonce spamming*. Rather than performing costly local computations, an attacker rapidly broadcasts vast quantities of candidate block headers, each with different nonce values, to reachable nodes in the network. These nodes independently verify the validity of these candidates, unknowingly performing the PoW calculation on behalf of the attacker. By exploiting this, the attacker effectively outsources the computational burden, undermining the fundamental premise that block creation reflects a miner's own effort.

Beyond nonce spamming, other attacks exploit nonce mechanisms in blockchain systems. Nonce reuse attacks occur when the same nonce is used for multiple transaction signatures, enabling attackers to recover private keys and compromise wallets, as demonstrated in studies on Bitcoin and Ethereum blockchains [10, 3]. Predictable nonce attacks exploit patterns in nonce generation, potentially allowing attackers to gain an advantage in mining by anticipating valid nonce values [5]. These vulnerabilities underscore the critical need for robust nonce management to ensure blockchain security.

While nonce spamming loses effectiveness in high difficulty environments, where competition is fierce and valid blocks are found rapidly, it remains a practical threat in smaller, test, or poorly optimized networks. This discrepancy reveals a significant gap between the theoretical security model of PoW and its real-world implementation.

Our work makes two key contributions: (1) a rigorous formalization of the nonce spamming technique and its operational impact within Bitcoin [8]-like Proof of Work systems, and (2) an in-depth analysis of the consequences this attack poses to miner accountability and the integrity of decentralized consensus. We conclude with a discussion on the urgent need for improved protocols and defenses that can restore trust in the fairness and security guarantees of Proof-of-Work, because even if Bitcoin is safe there are many other slower Proof of Work algorithm cryptocurrencies that are in danger of nonce spamming as I have witnessed itself while developing Hoosat Network.[6]

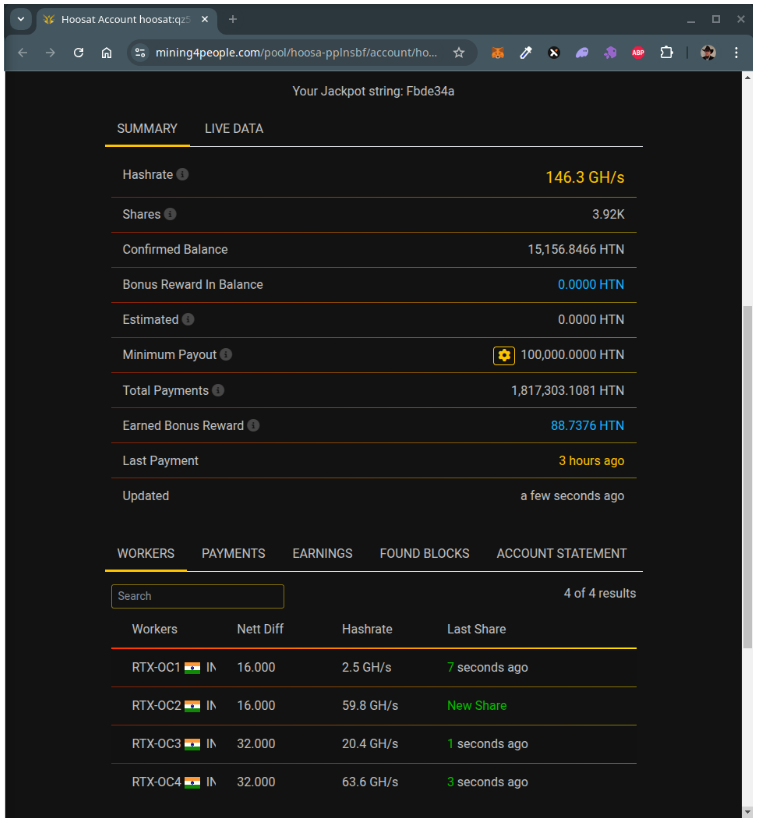
2 Attack Methodology

Nonce spamming exploits the fact that peer-to-peer (P2P) networks accept and validate block headers broadcast by miners without requiring proof that the miner has performed the necessary local computation. By rapidly submitting large volumes of candidate block headers with varying nonce values to reachable nodes, an adversary can effectively offload much of, if not all of the Proof-of-Work validation effort onto the network itself. Full nodes and mining pools independently verify each header's hash against the current target, unknowingly assisting the attacker in probabilistically discovering a valid block.

This technique has been observed in practice, particularly in certain forks of the Kaspa[7] blockchain that employ CPU- and GPU-friendly algorithms. In these networks, kHeavyHash ASIC miners have been used to perform filtered nonce spamming. Flooding the network with candidate nonce values. Because the kHeavyHash algorithm executes significantly faster than the intended algorithm (such as Hoohash), miners can rapidly cycle through nonce values. When a valid nonce is found for kHeavyHash, it is broadcast to the network, allowing ASIC miners to claim block rewards without performing the full computational effort required by the intended algorithm. This real-world exploitation of nonce spamming demonstrates its practical feasibility, particularly in Proof-of-Work systems that rely on intentionally slower hashing algorithms.

Nonce spamming can also enable a form of share stealing[4] within mining pools. Pools reward miners based on shares, nonce submissions that meet the pool's own target which is higher than the network target. As in rewarding for nonce values that do not create blocks. By rapidly submitting a large volume of candidate nonce values, an attacker can flood the pool with shares, some of which may be valid, without performing the full amount of local computation typically required to find each share. Since information about nonce spamming began circulating widely, in February 2025, we observed machines in the Mining4People pool engaging in share stealing for HTN and AIX using nonce spamming method. This was evident first from the hashrate and secondly as they frequently submitted shares that met only the pool's target, which resulted in barely any block discoveries. Suggesting they were selectively submitting shares without doing the normal target check miner software usually does before submitting work to stratum or node.

This flooding can skew the pool's payment schemes, allowing the attacker to claim disproportionate rewards relative to their actual computational contribution. Since the pool cannot distinguish between shares generated by genuine hashing effort and those resulting from nonce spamming, the attacker effectively "steals" shares by outsourcing the expensive proof-of-work calculations to the pool's infrastructure.



Your Jackpot string: Fbde34a

SUMMARY LIVE DATA

Hashrate	146.3 GH/s
Shares	3.92K
Confirmed Balance	15,156.8466 HTN
Bonus Reward In Balance	0.0000 HTN
Estimated	0.0000 HTN
Minimum Payout	100,000.0000 HTN
Total Payments	1,817,303.1081 HTN
Earned Bonus Reward	88.7376 HTN
Last Payment	3 hours ago
Updated	a few seconds ago

WORKERS PAYMENTS EARNINGS FOUND BLOCKS ACCOUNT STATEMENT

Search 4 of 4 results

Workers	Nett Diff	Hashrate	Last Share
RTX-OC1 🇮🇳 IN	16.000	2.5 GH/s	7 seconds ago
RTX-OC2 🇮🇳 IN	16.000	59.8 GH/s	New Share
RTX-OC3 🇮🇳 IN	32.000	20.4 GH/s	1 seconds ago
RTX-OC4 🇮🇳 IN	32.000	63.6 GH/s	3 seconds ago

Figure 1: Screenshot of the nonce-spammer in M4P that was share stealing HTN.

2.1 Basic Attack Model

In this attack, the adversary deliberately avoids performing extensive local hash computations. Instead, they generate valid block templates and systematically vary only the nonce field, rapidly broadcasting these candidate block headers to numerous reachable nodes. These nodes independently validate and propagate any valid headers they receive, unintentionally performing the attacker's Proof-of-Work on their behalf. The attack exploits the fact that the cost of transmitting many small headers over the network is significantly lower than the computational cost of hashing locally at scale.

2.2 Implementation Steps

1. **Retrieve a valid block template:** Acquire a valid block template from a full Bitcoin[8] node's `getblocktemplate`[2]), for creating the block headers for spamming.

2. **Filtering nonce submissions:** To optimize the spam attack and avoid overwhelming the network with obviously invalid nonce values, the attacker may implement lightweight filtering or heuristic checks locally before broadcasting candidate headers. This can include fast approximate hashing, early rejection of nonce values that clearly cannot meet the difficulty, or pruning strategies that limit nonce ranges based on partial hash results. By filtering out low-probability or redundant candidates, the attacker reduces bandwidth consumption and improves the chance that submitted headers are closer to valid, making the spamming more efficient and less likely to be discarded outright.

   For example we analyzed 886,931 existing Bitcoin nonce values finding some exciting results for nonce spamming techniques. The highest occurring distance between two separate nonce values in the nonce value dataset is exactly 300 with 233 occurrences.

3. **Iterate and broadcast candidate headers:** Systematically vary the nonce field, serialize the 80-byte block header for each candidate, and broadcast these headers to reachable nodes, relying on them to perform full Proof-of-Work validation.

4. **Select broadcast channels:**

   - *Stratum[11] protocol:* Connect to mining pools and submit manipulated shares or headers as if a legitimate miner.
   - *Direct JSON-RPC[2]:* Submit candidate headers directly to reachable full nodes using RPC[2] methods such as `submitblock` or `submitheader`.
   - *Modified full node:* Operate a Bitcoin[8] node patched to bypass local PoW validation and immediately relay candidate headers over the P2P network via `headers` and `inv` messages.

5. **Monitor for successful block acceptance:** Because local PoW verification is skipped, the attacker must monitor external indicators, such as wallet balances, public block explorers, or connected node logs, to detect when a broadcast header is accepted and mined into the blockchain.

2.3 Nonce Spamming vs. Difficulty Adjustment

The filtered nonce spamming methods used for share stealing in pools are not about modifying mining difficulty. In the Stratum protocol, miners can request specific difficulty levels, and pools may adjust these settings dynamically based on a miner's performance or hashing capacity. However, altering difficulty directly increase or decrease mining rewards, because each share should be credited proportionally to its difficulty in a fair stratum implementation. The pool tracks how much valid work a miner contributes by measuring the difficulty of each accepted share, not just the number of shares submitted.

Nonce spamming, does not attempt to lower the difficulty to inflate reward rates. Instead, it involves submitting a high volume of carefully constructed nonce values intended to maximize the likelihood of submitting accepted shares ahead of others, often by exploiting the job assignment, timing, or filtering mechanisms within the protocol. This can lead to cases of share hijacking or reward redirection. In this way, the attack abuses the share validation pipeline rather than manipulating the difficulty settings or the reward calculation logic itself.

As in if you can gain more profits by modifying pool difficulty, you are mining to pool that has incorrect implementation of pooled payment schemes.

2.4 Nonce spamming vs Hash collision

While nonce spamming aims to manipulate block submissions, it is important to distinguish this from hash collisions, which are a fundamentally different concept in the mining process. A hash collision occurs when two distinct inputs produce the same hash output. In the context of proof-of-work mining, collisions are theoretically possible but extremely rare due to the cryptographic properties of hash functions like SHA-256 or Keccak.

Nonce spamming does not rely on generating hash collisions; instead, it exploits protocol-level, where miners only submit the block template and found nonce back to the network. The attacker floods the network with multiple nonce attempts that meet the current difficulty target, aiming to outpace honest miners in submitting valid shares.

Hash collisions, if they were to occur, could theoretically cause unintended behavior in share validation, but their practical impact is negligible due to the extremely low probability. Hash collisions can happen in any kind of mining, even if it's nonce spamming.

In summary, nonce spamming is a strategic exploitation of the mining protocol's operational logic, whereas hash collisions are a cryptographic rarity that do not directly enable share stealing or difficulty manipulation.

2.5 Failure of deterministic nature

At first glance, the deterministic nature of PoW algorithms might seem like a safeguard against nonce-spamming attacks. After all if you think according to the words definition:

- deterministic: believing that everything that happens must happen as it does and could not have happened any other way, or relating to this belief.

Bitcoin's Proof-of-Work (PoW) relies on a deterministic hash function, meaning that for a fixed block data and nonce, the resulting PoW hash is always the same. This predictability is essential for miners to verify solutions and achieve network consensus. However, this deterministic nature does not prevent nonce-spamming attacks, where adversaries exploit the nonce generation process to flood the network with multiple block candidates.

Nonce spamming is possible because the deterministic function $H(\text{block data} \,||\, \text{nonce})$ allows attackers to generate numerous valid hashes by rapidly iterating through nonce values or slightly modifying fields like timestamps. Each variation produces a unique but potentially valid hash under the difficulty target, and the protocol cannot inherently distinguish these spam attempts from legitimate mining efforts. The deterministic property only ensures that a given input yields a consistent output, it does not regulate how nonce values are chosen or restrict the rate at which block candidates are submitted.

Consequently, while determinism supports consensus by enabling reproducible verification, it offers no security against attackers manipulating the nonce generation process to overwhelm the network. This vulnerability underscores the need for supplementary defenses, such as rate-limiting mechanisms, penalty systems, or stricter block validation rules, to curb nonce-spamming and protect the network beyond PoW's deterministic core.

2.6 Nonce spamming vs Banning for incorrect job submissions

Popular Stratum implementations, like MiningCore, are designed to be forgiving when miners submit incorrect results, known as "job submissions." This is because not every mistake is intentional. Sometimes, honest errors happen due to issues like faulty hardware (e.g., a graphics card overheating), slow internet connections causing delays, mismatched settings between the miner's software and the pool, or even bugs in the mining software or algorithms. If mining pools banned miners instantly for every mistake, they might unfairly kick out honest miners who are trying their best. To avoid this, Stratum pools use a system called a "threshold," which looks at the percentage of incorrect submissions (called "invalid shares") over time. Only if a miner submits too many invalid shares are they banned, reducing the risk of punishing innocent mistakes.

Now, let's imagine a situation where a Stratum implementation pool allows up to 50% of a miner's submissions to be invalid nonce values. In this case, the rule means that for every second submissions a miner makes, at least one must be valid (correctly solving the puzzle). This setup creates a loophole that a clever but dishonest miner could exploit. They could send one invalid nonce value, along with every block they mine correctly. This is essentially nonce spamming. If any of those invalid nonce values happen to meet the pool's difficulty target (the level of challenge set by the pool), the miner could get rewarded for work they didn't properly do. This gives them a chance to earn more rewards than they deserve compared to the actual computing power they're contributing, which can unfairly skew the pool's reward system and harm honest miners.

The same applies network wide between nodes. For example before Hoosat Networks nonce-spamming solution implementation, Golang implementation of Hoohash 1.0.0 algorithm occasionally worked differently on Arm and AMD64. Leading to blocks being created on the nodes running on ARM, which would not be accepted by AMD64 nodes. This to display that there are possibilities for false positives so the network also must use thresholds for banning other nodes.

This highlights a key challenge in designing fair banning mechanism to disable nonce-spamming: balancing leniency for honest errors with protections against deliberate cheating. Stratum pools and cryptocurrency nodes must carefully set their invalidity thresholds and add more monitoring patterns to detect and prevent nonce spamming without punishing miners for genuine mistakes.

2.7 Analysis of existing Bitcoin block nonce values

Since we have tried to explain the Bitcoin protocol does not require the Proof of Work output to be validated between the sender and recipient. We have done research on how to attack Bitcoin with nonce-spamming, even though it is highly improbable because Bitcoin is has very high difficulty and fast Proof of Work algorithm.

As such let's start from our research results, from our analysis results you can see that 0-655,349 is the most frequent number range for bitcoin nonce values.

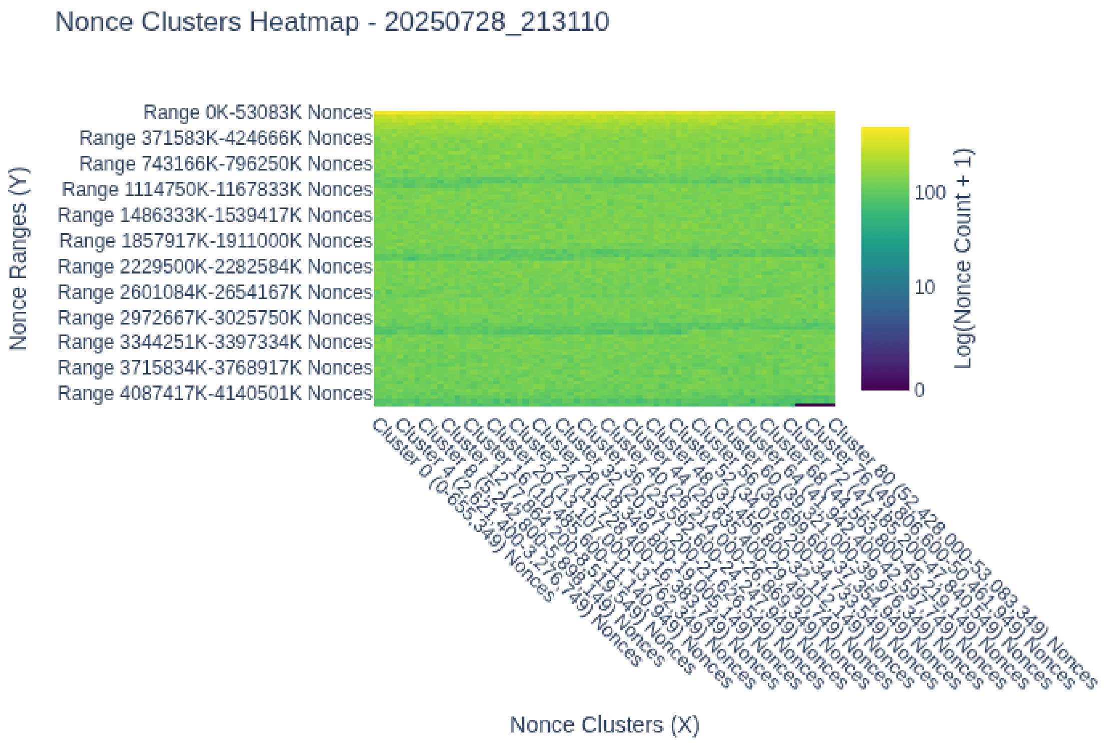


Figure 2: Heatmap of nonce clusters

This makes us see that if you don't want to waste work you can just keep spamming 655,349 different values out of the 4,295,163,899 nonce values and have about 0.000519% probability to hit a valid nonce with every submission. When solo change with 4.8 Th's at the time of writing this was 0.0000004776%. Think about the optimizations we could do to actual bitcoin mining by first trying the most probable nonce value sectors? We could easily optimize 15-30% of the hash rate out.

2.8 Excercise to understand nonce spamming

This exercise introduces a simplified version of the Bitcoin proof-of-work (PoW) protocol, focusing on the nonce-submission process. For clarity, complexities such as the Stratum protocol and extra nonce values are omitted.

In this exercise, you will collaborate with a partner to simulate the nonce-spamming methodology used in mining. Your partner acts as the node, while you take the role of the miner. Follow these steps to complete one mining cycle, then repeat the process to explore a common oversight:

1. **Request a block template**: Ask your partner for a new block template.
2. **Receive the block template**: Your partner provides a block template, including the previous block header, bits (difficulty target), transactions, and coinbase transaction.
3. **Submit a nonce**: Iteratively Compute the proof-of-work hash for the block template using new nonce values of each iteration and return the block template with the nonce to your partner once you find PoW hash satisfies the network's difficulty target.
4. **Verify the solution**: Your partner runs the PoW algorithm again with the submitted block template and nonce, confirming that the PoW hash satisfies the network's difficulty target.

Now, repeat the cycle while trying to guess nonce instead of calculating it.

1. **Request another block template**: Ask your partner for a new block template.
2. **Receive the block template**: Your partner provides a new block template with the same structured components as before.
3. **Submit a nonce**: Instead of computing the proof-of-work hash, guess a nonce and return the block template with this nonce to your partner.
4. **Verify the solution**: Your partner runs the PoW algorithm again with the submitted block template and nonce, confirming that the PoW hash does not satisfy the network's difficulty target.

Now, repeat the guess cycle with a successful nonce submission to highlight the key concept:

1. **Request another block template**: Ask your partner for a new block template.
2. **Receive the block template**: Your partner provides a new block template with the same components as before.
3. **Submit a nonce**: Instead of computing the proof-of-work hash, guess a nonce and return the block template with this nonce to your partner.
4. **Verify the solution**: Your partner runs the PoW algorithm again with the submitted block template and nonce, confirming that the PoW hash satisfies the network's difficulty target.

Reflection: Did you or your partner notice that in the third cycle, the miner submitted a guessed nonce without performing the proof-of-work hash? Please discuss why this is possible.

3 Classical vs. Quantum Nonce-Spamming

Nonce spamming has traditionally been analyzed in the context of classical computing architectures. However, the advent of quantum computing introduces fundamentally different computational paradigms that significantly impact the feasibility and efficiency of nonce-spamming attacks, particularly in Proof-of-Work (PoW) systems.

3.1 Nonce-Spamming with Classical CPUs

In classical systems, nonce spamming refers to the brute-force generation and rapid broadcast of block headers with varying nonce values, aiming to overwhelm a network or maximize the likelihood of discovering a valid PoW hash. This process is inherently linear and constrained by hardware limitations.

A typical CPU operating at 3 GHz can perform roughly 3×10^9 increment operations per second. Exhaustively scanning the 32-bit nonce space ($2^{32} = 4.29 \times 10^9$ possible values) is feasible within seconds. However, modern PoW systems often combine multiple fields (e.g., `nonce`, `extranonce2`), resulting in a significantly larger search space.

Consider the AMD Ryzen Threadripper PRO 7995WX, which features 96 cores and can theoretically execute:

$$96 \times 5 \times 10^9 = 4.8 \times 10^{11} \text{ increment operations/second.}$$

We will use the Threadrippers 480 Gh/s increment operation speed assuming perfect efficiency, later in the classical nonce-spamming calculation examples. This upper bound of 480 Gh/s excludes real-world limitations such as instruction latency, memory bottlenecks, and pipeline stalls, which typically reduce practical throughput to 30–70% of peak capacity. Even under ideal conditions, classical systems are orders of magnitude too slow to traverse large nonce spaces in a reasonable timeframe.

3.2 Quantum Nonce-Spamming

Quantum computing offers a fundamentally different model. A quantum processor with n qubits can represent all 2^n possible states simultaneously in superposition. For example, 32 qubits can represent the entire 32-bit nonce space, and 96 qubits can encompass the combined `nonce` + `extranonce2` space used in Bitcoin.

Although measurement yields only a single classical output, quantum algorithms (particularly Grover's algorithm) exploit superposition and interference to achieve quadratic speedup in unstructured search tasks. Specifically, Grover's algorithm requires $O(2^{n/2})$ operations to find a marked item in a search space of size 2^n, compared to $O(2^n)$ classically.

For nonce-spamming:

- Classical: $O(2^n)$ operations for an n-bit nonce.
- Quantum (Grover): $O(2^{n/2})$ queries to the oracle.

Thus, for a 32-bit nonce:

$$\text{Grover's runtime} \approx 2^{16} = 65{,}536 \text{ operations,}$$

which is dramatically faster than $2^{32} \approx 4.3 \times 10^9$ classical attempts.

Quantum hardware platforms, such as superconducting qubits from IBM or Google, can perform qubit measurements in microseconds. Measuring a 32–96 qubit register may take approximately 1–10 μs, making this approach extremely efficient provided that quantum coherence, gate fidelity, and error correction can be maintained throughout the measurement.

Moreover, storing all 2^{96} possible values explicitly would require on the order of 950 **yottabytes** (YB) of classical memory, an amount that far exceeds the estimated total global data storage capacity, which is currently measured in tens of zettabytes (ZB). In stark contrast, a 96-qubit quantum system can represent this entire search space compactly and simultaneously as a quantum superposition, leveraging the exponential growth of Hilbert space with each additional qubit. As quantum measurement times improve and become more amenable to parallelization, this capability could enable dramatically faster brute-force nonce generation, surpassing classical methods by many orders of magnitude.

3.3 Why Quantum Offers Speedup in Nonce-Spamming

Quantum computing[1] can offer significant speedup for nonce-spamming due to:

- **Superposition:** Simultaneous representation of all nonce candidates.
- **Quantum parallelism:** Execution of operations on all basis states simultaneously.
- **Grover's algorithm:** Reduces search complexity from $O(2^n)$ to $O(2^{n/2})$.
- **Fast measurement:** Rapid extraction of high-probability candidate solutions.

These advantages suggest that, once scalable quantum hardware is available, it could outperform classical brute-force nonce-spamming by many orders of magnitude, posing potential threats to current PoW systems. However, practical deployment remains contingent on overcoming current challenges in quantum error correction, qubit coherence, and large-scale integration.

3.4 Quantum Code Example

The following Python code, written using Qiskit for IBM Quantum, initializes a 32-qubit quantum circuit and performs repeated sampling using 8192 shots per iteration within a session loop.

Listing 1: Example Qiskit code to retrieve 32 bit nonce values

```python

from qiskit import QuantumCircuit
from qiskit_ibm_runtime import QiskitRuntimeService, SamplerV2 as Sampler
from qiskit.transpiler.preset_passmanagers import
    generate_preset_pass_manager
from collections import defaultdict

service = QiskitRuntimeService(channel="ibm_quantum", token="TOKEN")

backend = service.least_busy(operational=True, simulator=False,
    min_num_qubits=32)
print(f"Backend: {backend.name}, Qubits: {backend.configuration().n_qubits
    }")

qubits = 32
qc = QuantumCircuit(qubits, qubits)
qc.h(range(qubits))  # Apply Hadamard gates for superposition
qc.measure(range(qubits), range(qubits))  # Measure all qubits

pm = generate_preset_pass_manager(backend=backend, optimization_level=1)
isa_circuit = pm.run(qc)

total_shots = 2**qubits
shots_per_job = 8192
num_jobs = total_shots // shots_per_job

with service.session(backend=backend) as session:
    sampler = Sampler(backend=backend, session=session)
    for i in range(num_jobs):
        job = sampler.run([isa_circuit], shots=shots_per_job)
        print(f"Job {i+1}/{num_jobs}, Job ID: {job.job_id()}")
        result = job.result()
        print(f"Job {i+1} results:", result[0].data.c0.get_counts())
```

3.5 Feasibility and Limitations

While the probability that any given 80-byte block header submission with a random nonce yields a valid block is mathematically governed by the network difficulty target D, the practical effectiveness of nonce spamming is constrained by multiple real-world factors. Unlike traditional mining, which performs hash computations locally, nonce spamming attempts to offload this work onto the network by broadcasting large volumes of candidate headers for validation. However, this approach faces significant limitations.

First, bandwidth, latency, and per-node rate limiting reduce the feasible submission rate. Nodes, particularly full nodes, are not optimized for high-throughput validation and often impose anti-spam measures, connection caps, or drop unsolicited or invalid blocks. Second, mining pools, while more computationally capable usually have banning enabled for too many header submissions that do not meet their target.

At the time of writing, the Bitcoin [8] network includes approximately 21,745 publicly reachable nodes and 72 active mining pools. Hypothetically, if an attacker could distribute candidate headers at a rate of 480 GHz of PoW attempts per second across these endpoints, and if all nodes processed them as valid attempts, the network would collectively validate approximately 480 billion hashes per second. However, in practice, only a small fraction of this rate is achievable due to the constraints mentioned.

To contextualize this, the current Bitcoin network difficulty requires approximately $2^{70} \approx 1.18 \times 10^{21}$ hash attempts to probabilistically find a valid block. Even at an idealized rate of 4.8×10^{11} hash attempts per second, an attacker would require:

$$\frac{1.18 \times 10^{21}}{4.8 \times 10^{11}} \approx 2.458 \times 10^{9} \text{ seconds} \approx 77.97 \text{ years}$$

to discover a valid block, making the approach clearly infeasible on Bitcoin's mainnet. However, as will be explored in the case study of the Hoosat Network[6], smaller or less secure networks may offer significantly reduced difficulty thresholds, altering the attack feasibility.

Requirements for a Successful Nonce-Spamming Attack:

- **Protocol Compatibility:** The attacker must understand the exact protocol used for block submission, including the required structure and hashing algorithm. Nonce spamming cannot be performed across incompatible protocols. For instance, Bitcoin[8] and Kaspa[7] differ significantly: Kaspa employs a single 64-bit nonce, while Bitcoin uses a combination of a 32-bit nonce and a 64-bit `extranonce2`. Attempting to spam one network using a header structure from another is ineffective and will result in rejection.
- **Access to Public Nodes:** A large number of publicly reachable nodes is essential. Whether they accept block submissions via Stratum, RPC[2], or the peer-to-peer (P2P) network, these nodes validate incoming headers. The more nodes an attacker can reach, the more distributed hash computations they can offload, amplifying the scalability and impact of the attack.
- **Slow Proof-of-Work Algorithm:** Attacks are more effective on networks where the PoW algorithm has inherently slower block discovery rates. This delay increases the time window during which spammed nonce candidates can be submitted and validated, improving the probability of success before a legitimate miner finds the next block.
- **Low Network Difficulty:** Networks with low mining difficulty make each nonce attempt more likely to succeed. This enhances the attack's efficiency, as a smaller fraction of hash attempts is needed to produce a valid block hash below the target threshold.
- **Mining Pools with Weak Defenses:** Mining pools that rely on the Stratum protocol are particularly vulnerable if they lack proper validation mechanisms. Poorly secured pools may accept and reward spammed shares without verifying whether they were the result of actual local computation, enabling the attacker to extract disproportionate rewards.

Limitations of the nonce-spamming attacks:

- **High competition:** In networks with high aggregate hashrate, honest miners rapidly discover valid blocks, leaving minimal opportunity for spammed headers to win the race.
- **Rate limiting and bans:** Bitcoin[8] nodes often implement DoS protections, which throttle or outright ban peers that flood them with malformed or excessive headers.
- **Limited bandwidth and peer count:** Broadcasting headers to thousands of nodes at high frequency requires substantial network bandwidth and highly optimized peer management.
- **Ethical and reputational risk:** Engaging in nonce spamming may breach terms of use for mining pools or service providers and may be viewed as malicious behavior.

Nonetheless, in low-difficulty environments such as testnets, new altcoins, or during early-stage, nonce spamming can provide a viable, low-cost alternative to traditional mining by probabilistically leveraging the validation work of the network itself.

Case Study: Hoosat Network

As of writing, the Hoosat Network[6] has a difficulty of 0.140 T and a total network hashrate of approximately 280.29 GH/s. This gives us a much lower threshold for block discovery.

We can estimate the number of hashes needed per valid block by converting the difficulty into the equivalent hash target:

$$\text{Hoosat difficulty} = 0.140 \times 10^{12} = 1.4 \times 10^{11}$$

Now, applying the same nonce spamming rate as in the Bitcoin[8], the expected time to find a block through nonce spamming in Hoosat becomes:

$$\frac{1.4 \times 10^{11}}{480 \times 10^{9}} \approx 291.67 \text{ seconds} \approx 4.86 \text{ minutes}$$

This highlights a stark contrast in Pow network security[9] models. On the Hoosat network[6], an attacker spamming headers at an effective rate of 480 GH/s could statistically find a valid block roughly every 4.9 minutes, a dramatic improvement over the 77.97 years the same scale of attack would require on Bitcoin[8].

However, Hoosat Network[6] is the first cryptocurrency that actively defends against such attacks network wide by enforcing stricter protocol validation rules by enforcing correct algorithm to be used by the miner.

Key Takeaway: While nonce spamming is computationally impractical on high-difficulty networks like Bitcoin[8], it poses a real and immediate threat to emerging or low-difficulty networks, and cryptocurrencies which utilize CPU/GPU algorithms. The asymmetry between computational cost and validation outsourcing becomes more exploitable when the network scale is modest.

4 Comparison

Table 1 provides a comparative overview of several prominent Proof-of-Work cryptocurrencies, evaluating their susceptibility to classical nonce-spamming attacks aimed at discovering a valid block. The feasibility of such attacks is assessed assuming an attacker can perform 480 GH/s (4.8 $\times 10^{11}$ hashes per second) nonce spamming.

The table presents each system's nonce space, the estimated time required to find a valid block using classical brute-force nonce-spamming, the equivalent number of operations needed under Grover's quantum search algorithm, and the network difficulty, which reflects the computational effort required to mine a block.

Table 1: Comparison of Nonce Spamming Feasibility Across Systems

System	Nonce Space	Classical Time	(Grover's) Time	Difficulty
Bitcoin	$2^{32} + 2^{64}$	77.97 years	2^{48} operations	1.18×10^{21}
Kaspa	2^{64}	2.53 days	2^{32} operations	1.048622×10^{17}
Warthog	$2^{32} + 2^{64}$	1.45 minutes	2^{48} operations	4.17×10^{13}
Monero	$2^{32} + 2^{64}$	9.85 minutes	2^{48} operations	2.836×10^{11}
Karlsen	2^{64}	6.69 minutes	2^{32} operations	1.927323×10^{11}
Hoosat	2^{64}	4.86 minutes	2^{32} operations	1.4×10^{11}
Litecoin	$2^{32} + 2^{64}$	0.092 milliseconds	2^{48} operations	4.422×10^{7}
Dogecoin	$2^{32} + 2^{64}$	0.049 milliseconds	2^{48} operations	2.331×10^{7}

The expected time to find a block is calculated as:

$$\text{Time (seconds)} = \frac{\text{Network Difficulty}}{\text{Nonce Spamming Rate}}$$

This result is converted to appropriate units (e.g., years, days, hours, or minutes) for readability.

Bitcoin's exceptionally high network difficulty (1.18×10^{21}) results in a classical nonce-spamming time of approximately 77.97 years, rendering such attacks impractical. In contrast, Litecoin's significantly lower difficulty (4.422×10^{7}) allows a valid block to be found in just 0.092 milliseconds at a spamming rate of 480 GH/s. This stark difference arises from Litecoin's Scrypt algorithm, which, while memory-intensive and slow, results in lower difficulty targets compared to Bitcoin's SHA-256, making Litecoin far more susceptible to nonce-spamming attacks.

in Bitcoin's protocol. In contrast, solo mining with a 480 GH/s machine currently requires an estimated 39,838 years to find a block. By employing nonce spamming, you can reduce the time spent on this task by approximately 99.76%, significantly accelerating the block discovery process.

5 How fast to exhaust nonce space.

The time to exhaust a nonce space of 2^{32} at a spamming rate of 480 GH/s is calculated as:

$$\frac{2^{32}}{480 \times 10^9} \approx 0.008948 \text{ seconds} \approx 8.948 \text{ milliseconds}$$

The time to exhaust a nonce space of 2^{64} at a spamming rate of 480 GH/s is calculated as:

$$\frac{2^{64}}{480 \times 10^9} \approx 3.842653349 \times 10^{10} \text{ seconds} \approx 1,218.37 \text{ years}$$

The time to exhaust a nonce space of 2^{96} at a spamming rate of 480 GH/s is calculated as:

$$\frac{2^{96}}{480 \times 10^9} \approx 1.650587136 \times 10^{26} \text{ seconds} \approx 5.23 \text{ quintillion years}$$

This duration is approximately 379 million times the current age of the universe (13.8 billion years).

The point here is that is how long nonce-search space would take to produce valid nonce if the networks difficulty would be at the maximum possible difficulty, meaning that only one Proof of Work hash value is acceptable for a block. Of course we assume here, that all the block template data and timestamp is correct to reach valid nonce from the whole nonce-search space.

Bitcoin's theoretical maximum difficulty is derived from the ratio of the maximum target to the minimum target in its proof-of-work protocol:

$$\text{Difficulty} = \frac{\text{Maximum Target}}{\text{Current Target}}$$

The maximum target is approximately 2^{224}, and the minimum target is 2^{32}, yielding:

$$\text{Maximum Difficulty} = \frac{2^{224}}{2^{32}} = 2^{192} \approx 6.277 \times 10^{57}$$

Bitcoin remains highly secure, because of fast algorithm that directly affects the difficulty, which requires modifying the coinbase transaction with extranonce2, to even have a possibility of finding valid input data for SHA256D. In contrast, cryptocurrencies like Litecoin, Dogecoin, Monero, Kaspa, and Karlsen, with significantly lower difficulties, are far more susceptible to nonce-spamming, highlighting a critical distinction in their algorithms and network difficulty's ability to mitigate nonce-spamming methods.

6 Conclusion

In this work, we have identified, modeled, and analyzed a previously underappreciated attack vector against Proof-of-Work systems: nonce spamming. By exploiting the natural validation behavior of decentralized networks, attackers can shift the computational burden of mining onto other participants, effectively decoupling block creation from the local expenditure of work. This undermines one of the core assumptions of PoW-based consensus, that every valid block is a provable artifact of local computation and investment.

Our case study of the Hoosat Network[6] illustrates both the practical feasibility of nonce spamming in lower-difficulty environments and how careful protocol design, such as requiring miners to submit verifiable PoW hashes with each block, can render such attacks ineffective. In contrast fast algorithm and high-difficulty networks like Bitcoin[8] are more resilient in practice, but not necessarily by design; their vast computational scale merely raises the cost of exploitation beyond reach.

The broader implication of this study is that validation asymmetries in PoW networks, where verification is cheap and mining is expensive, can be strategically abused when trust assumptions about miner behavior are left unchecked. As decentralized systems continue to evolve, it is imperative to revisit the architectural and protocol-level assumptions that govern miner accountability. The level of

We advocate for the adoption of stricter validation schemes, such as explicit PoW hash submission and network-level relay policies, to restore integrity to the work being claimed by miners. Future research should explore additional defenses, including lightweight PoW proofs, hash commitments, or consensus-layer penalties for unverifiable mining activity.

Ultimately, ensuring the integrity of Proof-of-Work systems requires not only cryptographic robustness, but also thoughtful enforcement of the economic and computational principles that secure these networks in practice.

7 Acknowledgements

First and Foremost, I would like to thank the community of Hoosat Network[6] for keeping up with me. I would like to thank M. Cukrowski for his research with using Quantum machine for nonce-spamming. Lastly I would like to thank T. Høye from Mining4People for telling that Pyrinhash could be used to mine Hoohash which drove me to research this issue.

References

[1] Divesh Aggarwal, Gavin K. Brennen, Tony Lee, Miklos Santha, and Marco Tomamichel. Quantum attacks on bitcoin, and how to protect against them. *Ledger*, 3:68–90, 2018. doi: 10.5195/ledger.2018.127.

[2] Bitcoin Developers. BIP 22: getblocktemplate - fundamentals of a long polling mining protocol, 2012. URL https://github.com/bitcoin/bips/blob/master/bip-0022.mediawiki. Bitcoin Improvement Proposal.

[3] Joachim Breitner and Nadia Heninger. Biased nonce sense: Lattice attacks against weak ecdsa signatures in cryptocurrencies. *IACR Cryptology ePrint Archive*, 2019. URL https://www.researchgate.net/publication/336437771_Biased_Nonce_Sense_Lattice_Attacks_Against_Weak_ECDSA_Signatures_in_Cryptocurrencies.

[4] Max Cochran and Amr M. Youssef. Exploiting miner misbehavior: Share stealing and difficulty manipulation in mining pools. In *International Conference on Information Security and Cryptology (ICISC)*, pages 147–165. Springer, 2021.

[5] Cointelegraph. What is a nonce in blockchain? explained, 2024. URL https://cointelegraph.com/explained/what-is-a-nonce-in-blockchain-explained.

[6] Hoosat team. Hoosat: Hoosat network whitepaper, 2024. URL https://network.hoosat.fi/public/htn-whitepaper.pdf. Project Documentation.

[7] Kaspa Team. Kaspa: The fastest open-source, decentralized and fully scalable layer-1, 2022. URL https://kaspa.org/. Project Documentation.

[8] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. URL https://bitcoin.org/bitcoin.pdf. White paper.

[9] Ivan Pustogarov et al. On the security and performance of proof of work blockchains. In *IFIP International Information Security Conference*, pages 406–417. Springer, 2014.

[10] Kudelski Security. Polynonce: A tale of a novel ecdsa attack and bitcoin tears, 2023. URL https://research.kudelskisecurity.com/2023/03/06/polynonce-a-tale-of-a-novel-ecdsa-attack-and-bitcoin-tears/.

[11] Slushpool Team. Stratum mining protocol, 2012. URL https://en.bitcoin.it/wiki/Stratum_mining_protocol. Bitcoin Wiki Entry.