HASHISH Protocol

The First True Proof-of-Work Protocol on Solana

Mine off-chain. Verify on-chain. Earn rewards.

Whitepaper v1.0

February 2026

Abstract — HASHISH is a Proof-of-Work mining protocol built natively on Solana. It combines off-chain SHA-256 mining with on-chain proof verification, adaptive difficulty adjustment, dual mining pools with Seeker device attestation, a PPLNS-based pool system, and optional privacy-preserving mining through Arcium multi-party computation. The protocol introduces a deflationary tokenomics model with exponential reward decay, progressive fee scaling, and automated buyback-and-burn mechanics.

Contents

1 Introduction

Proof-of-Work (PoW) consensus has historically been the most battle-tested mechanism for decentralized token distribution. Bitcoin demonstrated that computational work can serve as an objective measure for fair issuance. However, existing PoW implementations either operate as standalone Layer 1 blockchains or rely on fragile bridging mechanisms.

HASHISH brings native Proof-of-Work mining to Solana, the highest-throughput blockchain, by separating the mining computation from the verification layer. Miners perform SHA-256 hashing off-chain using CPUs or GPUs, then submit valid proofs on-chain where the Solana program verifies and distributes rewards in a single atomic transaction.

1.1 Design Goals

1. **Fair Distribution** — 99.9% of token supply distributed exclusively through mining.
2. **Decentralized Mining** — Support for solo miners, mining pools, and privacy-preserving modes.
3. **Adaptive Difficulty** — Fast-converging difficulty adjustment using a moving average algorithm.
4. **Deflationary Economics** — Automated buyback-and-burn funded by mining fees.
5. **Privacy Option** — Optional encrypted mining via Arcium MPC where miner identities and balances remain confidential.
6. **Hardware Diversity** — Dual-pool architecture with Seeker device attestation to promote hardware decentralization.

2 Protocol Architecture

HASHISH consists of three on-chain Solana programs working in concert:

Program	Role	Program ID
`pow-protocol`	Core mining, verification, fees	`Ai9Xr...GyqER`
`pow-pool`	PPLNS mining pools	Operator-deployed
`pow-privacy`	Arcium MPC integration	`DJB2P...Y5f`

Table 1: On-chain program overview.

2.1 Account Layout

The protocol state is stored in the following Program Derived Accounts (PDAs):

- **PowConfig** — Per-pool global state: difficulty, challenge, block count, fee accumulators, reward trackers, and a 10-slot circular buffer for difficulty adjustment.
- **MinerStats** — Per-miner per-pool statistics: blocks mined, tokens earned, fees paid, timestamps.

- **MintAuthority** — Shared PDA across both pools that signs `mint_to` CPI calls, preventing duplicate minting.
- **DeviceAttestation** — Seeker pool attestation records with 60-second validity and single-use consumption.
- **Fee Vaults** — Separate PDAs for fee collection, team allocation, buyback, and LP provisioning.

3 Mining Mechanism

3.1 Hash Function & Proof Construction

HASHISH uses SHA-256 as its proof-of-work function. A valid proof requires finding a nonce n such that:

$$\text{SHA-256}\left(\texttt{challenge}\|\texttt{miner\_pubkey}\|n\|\texttt{blocks\_mined}\right) < \text{target} \tag{1}$$

where:

- `challenge` $\in \{0,1\}^{256}$ is a 32-byte per-pool challenge seed updated after each block,
- `miner_pubkey` $\in \{0,1\}^{256}$ is the miner's Solana public key (prevents work theft),
- $n \in [0, 2^{128})$ is a 128-bit nonce,
- `blocks_mined` $\in \mathbb{N}$ is the current block height (64-bit).

The total hash input is 88 bytes. The first 16 bytes of the resulting hash are interpreted as a little-endian `u128` value and compared against the target:

$$\text{target} = \left\lfloor \frac{2^{128} - 1}{\text{difficulty}} \right\rfloor \tag{2}$$

Higher difficulty produces a smaller target, requiring more computational work on average.

3.2 Challenge Generation

After each block, the challenge is updated deterministically:

$$\texttt{challenge}_{i+1} = \text{SHA-256}\left(\texttt{challenge}_i\|n_i\|\texttt{slot}\|i\right) \tag{3}$$

where `slot` is the current Solana slot number and i is the block number. This ensures unpredictability and per-pool uniqueness.

3.3 Difficulty Adjustment Algorithm

The protocol employs an adaptive proportional difficulty adjustment using a moving average of the last 10 block timestamps. Let $\bar{t}$ denote the average block time over the window. The adjustment ratio $r = \bar{t}/60$ determines the multiplier:

Condition	Block Time Range	Difficulty Multiplier
$r < 0.5$	< 30 s	$\times 2.0$
$r < 0.75$	30–45 s	$\times 1.5$
$r < 0.9$	45–54 s	$\times 1.1$
$0.9 \leq r \leq 1.1$	54–66 s	$\times 1.0$ (no change)
$r \leq 1.33$	66–80 s	$\times 0.9$
$r \leq 2.0$	80–120 s	$\times 0.7$
$r > 2.0$	> 120 s	$\times 0.5$

Table 2: Difficulty adjustment schedule. Target block time: 60 seconds.

Convergence. The moving-average approach converges 81% faster than fixed 2% step adjustments — approximately 38 blocks to stabilize versus 580 blocks.

Bounds. Difficulty is clamped to $[\,1{,}000,\ (2^{128} - 1)/1{,}000\,]$ to prevent degenerate states.

3.4 Mining Backends

The reference miner client supports three backends:

1. **CPU** — Multi-threaded via Rayon work-stealing scheduler. Typical hashrate: 3–5 MH/s.
2. **CUDA** — NVIDIA GPU kernel execution via `cudarc`. Hashrate varies by card (10–200+ MH/s).
3. **OpenCL** — AMD/cross-vendor GPU support via `ocl`.

Auto-detection probes CUDA → OpenCL → CPU fallback.

4 Dual-Pool Architecture

The protocol operates two independent mining pools at the protocol level, each with its own `PowConfig`, difficulty, challenge, and block counter.

Pool ID	Name	Requirement
0	Standard	Open to all miners
1	Seeker	Requires TEE device attestation

Table 3: Protocol-level mining pools.

4.1 Seeker Pool & Device Attestation

The Seeker pool is designed to promote hardware diversity and prevent ASIC/FPGA dominance. Miners must present a valid `DeviceAttestation` account to submit proofs to Pool 1.

Attestation lifecycle:

1. A backend authority validates the miner's TEE (Trusted Execution Environment) hardware.
2. The authority creates an on-chain `DeviceAttestation` record.
3. The attestation is valid for **60 seconds** and consumed after a single proof submission.
4. Miners must re-attest before each block submission.

This ensures that Seeker pool miners are running on verified consumer hardware, maintaining a fair distribution channel separate from high-performance mining rigs.

4.2 Shared Mint Authority

Both pools share a single `MintAuthority` PDA with seeds `[b"pow_mint_auth"]`. This guarantees that the combined minting across both pools cannot exceed the maximum supply, regardless of the independent block counters.

5 Mining Pools — PPLNS System

Beyond the protocol-level dual pools, HASHISH supports operator-run mining pools using a **Pay-Per-Last-N-Shares (PPLNS)** reward distribution model.

5.1 Pool Configuration

Pool operators deploy a `PoolConfig` account with the following parameters:

- **Difficulty Divisor** (d): Pool difficulty = protocol_difficulty/d, where $d \in [10, 10{,}000]$.
- **Operator Fee**: Up to 10% (1,000 basis points) of block rewards.
- **PPLNS Window** (W): Circular buffer of W shares, where $W \in [100, 10{,}000]$.
- **Share Fee**: Per-share submission cost, default 0.00001 SOL.

5.2 Share Submission & Reward Distribution

Shares. Pool members mine against the reduced pool difficulty. Each valid partial proof is recorded as a share in the PPLNS circular buffer with the miner's public key, timestamp, and round number.

Block Discovery. When a member's hash also satisfies the full protocol difficulty, the pool operator submits a CPI call to `pow_protocol::submit_proof`. The protocol mints tokens to the pool's vault.

Distribution. Let s_i denote miner i's shares in the PPLNS window and $S = \sum_i s_i$. For a block reward R and operator fee rate f:

$$\text{reward}_i = \frac{s_i}{S} \times R \times (1 - f) \tag{4}$$

Members claim their accumulated rewards via the `claim_rewards` instruction.

6 Tokenomics

6.1 Token Specifications

Parameter	**Value**
Token Standard	SPL-2022
Maximum Supply	1,000,000 tokens
Decimals	9
Premint (LP seed)	1,000 tokens (0.1%)
Mining Allocation	999,000 tokens (99.9%)

Table 4: Token parameters.

6.2 Emission Schedule

Block rewards follow an exponential decay model. Let R_0 denote the initial reward per block per pool:

$$R(b) = R_0 \times k^b \tag{5}$$

where b is the block number and $k = 0.999999943$ is the per-block decay factor.

Period	R_0 **per pool**	**Effective** R_0 **(2 pools)**
Year 1 (Boost)	0.04435 tokens	0.0887 tokens
Year 2+ (Normal)	0.02870 tokens	0.0574 tokens

Table 5: Initial reward rates. The boost period lasts 31,536,000 seconds from launch.

Yearly decay. With 525,600 blocks per year per pool:

$$k^{525{,}600} \approx 0.9705$$

This means rewards decrease by approximately 2.95% per year, asymptotically approaching but never reaching zero.

Minimum floor. A reward floor of 0.001 tokens per block prevents dust emissions.

6.3 Fee Structure

6.3.1 Mining Fee

Each proof submission requires a SOL fee that scales progressively:

$$\text{fee}(y) = \min\left(0.001 \times 1.5^{\lfloor y/2 \rfloor},\ 0.5\right) \quad \text{SOL} \tag{6}$$

where y is the number of years since launch.

Period	**Fee (SOL)**
Year 0–2	0.001
Year 2–4	0.0015
Year 4–6	0.00225
Year 6–8	0.003375
...	...
Cap	0.5

Table 6: Progressive fee scaling.

6.3.2 Fee Distribution

Collected fees are distributed as follows:

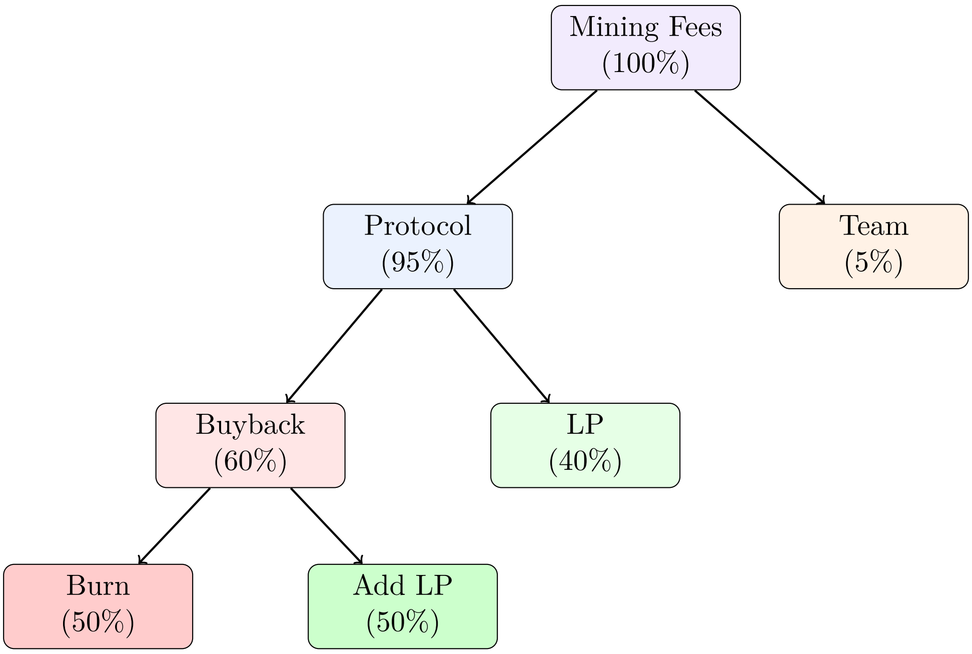

Figure 1: Fee distribution flow. 57% of total fees fund buyback (half burned, half to LP). 38% goes directly to LP. 5% to team.

6.4 Deflationary Dynamics

The protocol is structurally deflationary through two burn mechanisms:

1. **Buyback Burns**: 50% of tokens acquired via SOL buyback are burned (funded by 57% of mining fees).

2. **Diminishing Emissions**: Exponential decay ensures mining rewards approach zero asymptotically.

Over time, the burn rate surpasses the emission rate, creating net deflation.

7 Privacy-Preserving Mining with Arcium MPC

7.1 Motivation

Standard on-chain mining reveals the miner's public key, reward destination, and balance in every transaction. This transparency enables:

- Competitor surveillance of hashrate and earnings,
- Targeted attacks against profitable miners,
- Wealth profiling and transaction graph analysis.

HASHISH addresses these concerns with an optional privacy layer powered by **Arcium MXE (Multi-party eXecution Environment)**, enabling miners to mine, accumulate, and withdraw rewards without revealing their identity or balance on-chain.

7.2 Cryptographic Primitives

The privacy system employs a hybrid encryption scheme:

1. **x25519 Key Exchange** — Elliptic-curve Diffie-Hellman for shared secret derivation.
   - The Arcium MXE cluster holds a long-term x25519 public key generated during Distributed Key Generation (DKG).
   - Each miner generates an ephemeral x25519 keypair per operation.
   - Shared secret: $s = \text{x25519}(\text{client\_private}, \text{mxe\_public})$.
2. **RescueCipher** — A symmetric cipher optimized for MPC arithmetic.
   - Takes plaintext, a 128-bit nonce, and the shared secret.
   - Outputs ciphertexts that the MXE cluster can decrypt and compute over.
   - MPC-friendly: operates over finite fields without branching.
3. **BLS Signatures** — Batch-verifiable signatures on MPC outputs for on-chain verification of computation integrity.

7.3 Encrypted Data

Data	Encoding	Purpose
Destination wallet	4× `u64` ciphertexts	Hides reward recipient
Miner balance	3× `u64` ciphertexts	Balance + nonce + reserved
Claim secret	4× `u64` ciphertexts	Proves claim ownership
Protocol fee	1× `u64` ciphertext	Hides fee details
Withdrawal amount	1× `u64` ciphertext	Hides cashout size

Table 7: Encrypted fields and their encoding.

7.4 MPC Computation Circuits

The Arcium MXE executes six computation definitions (circuits) for the privacy layer:

7.4.1 `store_claim`

Stores the encrypted destination wallet for a mining reward claim. The MPC cluster receives the ciphertexts and persists them for later verification.

7.4.2 `verify_and_claim`

Verifies the miner's secret against the stored hash, decrypts the destination address, and triggers token transfer. The destination is *never revealed on-chain* — only the MPC cluster learns it during the callback.

7.4.3 `deposit_fee`

Adds SOL to the miner's encrypted balance. The MPC performs the addition in the encrypted domain:

$$\text{balance}' = \text{balance} + \text{amount}, \quad \text{nonce}' = \text{nonce} + 1$$

7.4.4 `mine_block`

Verifies that the miner has sufficient encrypted balance to pay the protocol fee, deducts the fee, and returns a success indicator. Uses constant-time comparison to prevent side-channel leakage:

$$\text{balance}' = \begin{cases} \text{balance} - \text{fee} & \text{if balance} \geq \text{fee} \\ \text{balance} & \text{otherwise} \end{cases}$$

7.4.5 `withdraw_fee`

Deducts funds from the encrypted balance and decrypts the withdrawal destination for the callback to transfer SOL.

7.4.6 `check_miner_balance`

Queries the encrypted balance. Only the requesting miner can decrypt the result.

7.5 Private Mining Flow

Algorithm 1 Privacy-Preserving Block Submission

Miner: Find valid nonce n such that Eq. (1) holds
Miner: Generate ephemeral x25519 keypair (sk_c, pk_c)
Miner: Compute shared secret $s = \text{x25519}(sk_c, pk_{\text{mxe}})$
Miner: Encrypt destination: $\mathcal{E}_s(\text{dest}) \to [\texttt{u64}; 4]$
Miner: Generate random secret σ; compute $h = \text{SHA-256}(\sigma)$
Miner: Store $(\sigma, \text{dest}, pk_c)$ locally in SQLite database
Relayer: Submit `submit_block_private` with encrypted data
On-chain: Verify PoW, create `Claim` with encrypted destination
On-chain: Queue `mine_block` MPC computation
Arcium MPC: Decrypt balance, verify balance $\geq$ fee, deduct fee
MPC Callback: Mark claim as verified (BLS-signed result)

— *Later (after $\geq 30s$ MPC finalization)* —

Miner: Call `claim_reward` with encrypted secret $\mathcal{E}_s(\sigma)$
On-chain: Verify $\text{SHA-256}(\sigma) = h$
On-chain: Queue `verify_and_claim` MPC computation
Arcium MPC: Decrypt destination, verify secret
MPC Callback: Transfer tokens to decrypted destination

7.6 Privacy Guarantees

Property	**Guarantee**
Mining Anonymity	Miner public key never linked to reward destination on-chain
Balance Privacy	Miner balances stored encrypted in MPC; never in plaintext on-chain
Transaction Unlinkability	Mining activity cannot be correlated across blocks
Withdrawal Privacy	Cashout destination encrypted until MPC callback
Anti-Replay	Nonce incremented on every balance operation
Computation Integrity	BLS signatures on all MPC outputs verified on-chain

Table 8: Privacy guarantees of the Arcium integration.

7.7 Relayer Architecture

Privacy mining uses a **relayer** — a separate wallet that pays transaction fees on behalf of the miner. This further decouples the miner's identity from on-chain activity. The relayer:

- Submits `submit_block_private` transactions,
- Cannot access encrypted data (only the MPC cluster can decrypt),
- Is visible on-chain but reveals nothing about the actual miner.

8 On-Chain Verification

The `submit_proof` instruction performs the following verification steps atomically within a single Solana transaction:

1. **Hash Verification**: Recomputes SHA-256($\texttt{challenge} \| \texttt{miner} \| n \| \texttt{blocks}$) and verifies the result is below the target.
2. **Difficulty Check**: Confirms the hash satisfies the current difficulty threshold.
3. **Fee Collection**: Transfers the current mining fee from the miner to the fee vault.
4. **Token Minting**: Computes the current reward using the decay function and mints tokens to the miner.
5. **State Update**: Increments block counter, updates challenge, records timestamp in the circular buffer, and adjusts difficulty.
6. **Attestation Check** (Pool 1 only): Verifies the `DeviceAttestation` account is valid ($<$ 60s old) and marks it as consumed.

Compute budget: 400,000 compute units per proof submission.

9 Security Considerations

9.1 Work Theft Prevention

The miner's public key is included in the hash input (Eq. 1). A valid nonce found by miner A cannot be submitted by miner B, as the hash would differ.

9.2 Difficulty Manipulation Resistance

The 10-block moving average window smooths out individual block time variations. The dead zone ($0.9 \leq r \leq 1.1$) prevents oscillation around the target. Difficulty bounds prevent both trivial mining (diff $\geq$ 1,000) and impossibly hard targets.

9.3 Supply Integrity

The shared `MintAuthority` PDA ensures that both pools mint from the same authority, making it impossible to exceed the 1,000,000 token maximum supply. The decay function and per-pool supply tracking provide additional safeguards.

9.4 Privacy Security

- **x25519** provides 128-bit ECDH security for key exchange.
- **RescueCipher** enables arithmetic over ciphertexts without revealing plaintext.
- **Constant-time MPC operations** prevent side-channel leakage during balance checks.
- **BLS signature verification** ensures MPC output integrity.
- **Anti-replay nonces** prevent state rollback attacks on encrypted balances.
- **Claim expiry** (365 days) bounds the system's liability for unclaimed rewards.

10 Protocol Parameters

Parameter	Description	Value
Token		
Max Supply	Total token cap	1,000,000
Decimals	Token precision	9
Premint	Initial LP seed	1,000
Mining		
Hash Algorithm	Proof-of-work function	SHA-256
Target Block Time	Seconds per block	60
Blocks/Year/Pool	Annual block count	525,600
Nonce Size	Search space	128 bits
Min Difficulty	Lower bound	1,000
Difficulty Window	Moving average slots	10
Pools	Protocol-level pools	2
Rewards		
Year 1 Boost (per pool)	Enhanced early reward	0.04435 tokens
Normal Rate (per pool)	Standard reward	0.02870 tokens
Decay Factor	Per-block multiplier	0.999999943
Reward Floor	Minimum per block	0.001 tokens
Fees		
Initial Mining Fee	Year 0–2	0.001 SOL
Fee Scaling	Every 2 years	×1.5
Fee Cap	Maximum	0.5 SOL
Team Allocation	Of total fees	5%
Buyback Allocation	Of protocol fees	60%
LP Allocation	Of protocol fees	40%
Pools (PPLNS)		
Difficulty Divisor Range	Pool difficulty reduction	10–10,000
Max Operator Fee	Basis points	1,000 (10%)
PPLNS Window Range	Share history	100–10,000
Privacy		
Key Exchange	Asymmetric scheme	x25519
Symmetric Cipher	MPC-friendly cipher	RescueCipher
MPC Provider	Computation network	Arcium MXE
Attestation Validity	Seeker pool	60 seconds
Claim Expiry	Privacy claims	365 days

Table 9: Complete protocol parameter table.

11 Conclusion

HASHISH introduces a novel Proof-of-Work mining protocol natively integrated with Solana's high-throughput architecture. By separating off-chain computation from on-chain verification, the protocol achieves the fairness guarantees of PoW distribution while leveraging Solana's finality and composability.

The dual-pool system with Seeker device attestation promotes hardware diversity. The PPLNS mining pool framework enables collective mining with transparent reward distribution. The Arcium MPC privacy layer provides miners with the option to mine, deposit, and withdraw without revealing their identity or balance on-chain.

The deflationary tokenomics — exponential emission decay, progressive fee scaling, and automated buyback-and-burn — create long-term value accrual for token holders while maintaining sustainable miner incentives.

HASHISH demonstrates that Proof-of-Work and modern high-performance blockchains are not mutually exclusive, but complementary.

References

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008.

[2] A. Yakovenko, "Solana: A new architecture for a high performance blockchain," 2018.

[3] Arcium, "Multi-party eXecution Environment (MXE): Confidential computing for blockchains," https://arcium.com.

[4] Solana Labs, "SPL Token-2022 Program," https://spl.solana.com/token-2022.

[5] NIST, "Secure Hash Standard (SHS)," FIPS PUB 180-4, 2015.

[6] D. J. Bernstein, "Curve25519: new Diffie-Hellman speed records," *PKC 2006*, pp. 207–228.

[7] A. Aly et al., "Design of Symmetric-Key Primitives for Advanced Cryptographic Protocols," *IACR ToSC*, 2020.

[8] M. Rosenfeld, "Analysis of Bitcoin Pooled Mining Reward Systems," *arXiv:1112.4980*, 2011.