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	Mainnet Program ID
`pow-protocol`	Core mining, verification, fees	`PoWQ79...tiSh`
`pow-treasury`	Buyback / burn / LP cycle	`LPAtdQ...Qish`
`pow-privacy`	Arcium MPC privacy layer	`CUsP6A...dmkg`
`transfer-hook`	SPL Token-2022 transfer tax logic	`95zaGU...P6S3`

Table 1: On-chain program overview. Full program IDs: `pow-protocol` `PoWQ79wY7LXrKaU8vZBoFb4JgSytENSdpAQAPJaZiSh`, `pow-treasury` `LPAtdQ9sYQGXNs3RejVcWgi2u516nedpHABdwzmQish`, `pow-privacy` `CUsP6AJxG7VyEP2dR1LfzX7KDsn7FiWDPgMxWtUUdmkg`, `transfer-hook` `95zaGUMvrNFnCpjSqQyTNw3msyJtj9drQ5mWYm1eP6S3`. All programs are verified via OtterSec reproducible builds.

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, burn counters (buyback + transfer tax), reward trackers, and a 100-slot circular buffer of block timestamps 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:

- $\texttt{challenge} \in \{0,1\}^{256}$ is a 32-byte per-pool challenge seed updated after each block,
- $\texttt{miner\_pubkey} \in \{0,1\}^{256}$ is the miner's Solana public key (prevents work theft),
- $n \in [0, 2^{128})$ is a 128-bit nonce,
- $\texttt{blocks\_mined} \in \mathbb{N}$ 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
$r < 0.5$	$< 30\,\text{s}$	$\times 2.0$
$r < 0.75$	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	$\times 0.7$
$r > 2.0$	$> 120\,\text{s}$	$\times 0.5$

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 Tokenomics

5.1 Token Specifications

Parameter	Value
Token Standard	SPL Token-2022
Extensions	`TransferFeeConfig`, `TransferHook`, `MetadataPointer`
Maximum Supply	1,000,000 HASHISH
Decimals	9
Genesis Mint (LP + launch airdrop)	103.409135608 HASHISH ($\approx 0.0103\,\%$)
Scheduled Mint (Y+1 devnet airdrop)	1,132.969400404 HASHISH ($\approx 0.113\,\%$)
Mining Allocation	998,763.621464 HASHISH ($\approx 99.876\,\%$)

Table 4: Token parameters. The mining allocation is everything not preminted or reserved for the scheduled Y+1 devnet airdrop.

5.2 Token Distribution

The overwhelming majority of HASHISH is mined. Three allocations are seeded outside the mining curve:

1. **Initial Liquidity Pool (genesis).** 100 HASHISH paired with 1 SOL are seeded into the Meteora DAMM v2 `HASHISH/SOL` pool at launch. These tokens provide the price-discovery venue that the `pow-treasury` buyback-and-burn cycle trades against.

2. **Launch Airdrop (genesis).** 3.409135608 HASHISH are airdropped at launch to early community contributors and builders. This allocation is deliberately small so that the fair-mining property is preserved.

3. **Devnet-Miner Airdrop (launch + 1 year).** 1,132.969400404 HASHISH are reserved for miners who ran nodes during the devnet phase. These are minted on a single scheduled event exactly one year after mainnet launch and distributed pro-rata to devnet miners based on their on-chain blocks-mined snapshot. Until this date the allocation sits outside circulating supply.

Allocation	**Amount (HASHISH)**	**% of $S_{\max}$**
Initial LP seed	100.000000000	0.0100 %
Launch airdrop	3.409135608	0.0003 %
Devnet-miner airdrop (Y+1)	1,132.969400404	0.1133 %
Mining (open emission)	**998,763.621464**	**99.8764 %**
Total	**1,000,000**	**100 %**

Table 5: HASHISH allocation breakdown. All non-mining allocations sum to $\sim 0.124\,\%$ of maximum supply; the remaining $\sim 99.876\,\%$ is emitted exclusively through proof-of-work mining.

Cumulative emission accounting. The scheduled Y+1 devnet airdrop counts toward the cumulative emission E used in the fee curve (Equation 5): once it is minted, the fee ramps accordingly. The initial LP seed and launch airdrop are included in E from block zero.

5.3 Emission Schedule

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

$$R(b) = R_0 \times k^{\,b} \tag{4}$$

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 6: 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.

5.4 Fee Structure

5.4.1 Mining Fee

Each proof submission requires a SOL fee that scales **geometrically with cumulative token emission** rather than wall-clock time. This ties the cost of entering the network to the scarcity of the remaining supply:

$$\text{fee}(E) = \min\Big(0.001 \times 1000^{\,E/S_{\max}},\ 1\Big) \quad \text{SOL} \tag{5}$$

where $S_{\max} = 10^6$ is the maximum supply and E is the *cumulative emission* across both pools, computed on-chain as

$$E = \sum_{p \in \{0,1\}} \Big(\text{supply}_p + \text{burned}_p^{\text{buyback}} + \text{burned}_p^{\text{transfer}}\Big). \tag{6}$$

Because E sums current supply *plus* all historical burns, it is monotonically non-decreasing: token burns (buyback or transfer tax) do not reset the fee ramp. The fee starts at 0.001 SOL at launch and reaches the 1 SOL cap only once the protocol has emitted one full $S_{\max}$ worth of tokens.

Cumulative Emission $E/S_{\max}$	**Fee (SOL)**
0 %	0.001
10 %	0.002
25 %	0.0056
50 %	0.0316
75 %	0.178
90 %	0.501
100 %	1.000 (cap)

Table 7: Geometric fee as a function of cumulative emission.

On-chain implementation. Equation 5 is evaluated without floating-point arithmetic. The ratio $E/S_{\max}$ is expressed in basis points and decomposed into four decimal digits (tens, ones, tenths, hundredths of a percent); each digit multiplies the base fee by a precomputed constant $1000^{1/10}$, $1000^{1/100}$, $1000^{1/1000}$, or $1000^{1/10\,000}$ stored with 10^{12} precision. The full computation costs at most 36 iterations and fits well within Solana's compute budget.

5.4.2 Fee Distribution

Collected SOL fees are split at the point of collection:

- **5 %** → `team_vault` (team allocation).
- **95 %** → `treasury_sol_vault`, managed by the `pow-treasury` program.

The treasury SOL is then processed autonomously by a two-phase cycle driven by a `CycleGate` PDA that pow-protocol updates every `BLOCKS_PER_CYCLE` blocks (currently 10).

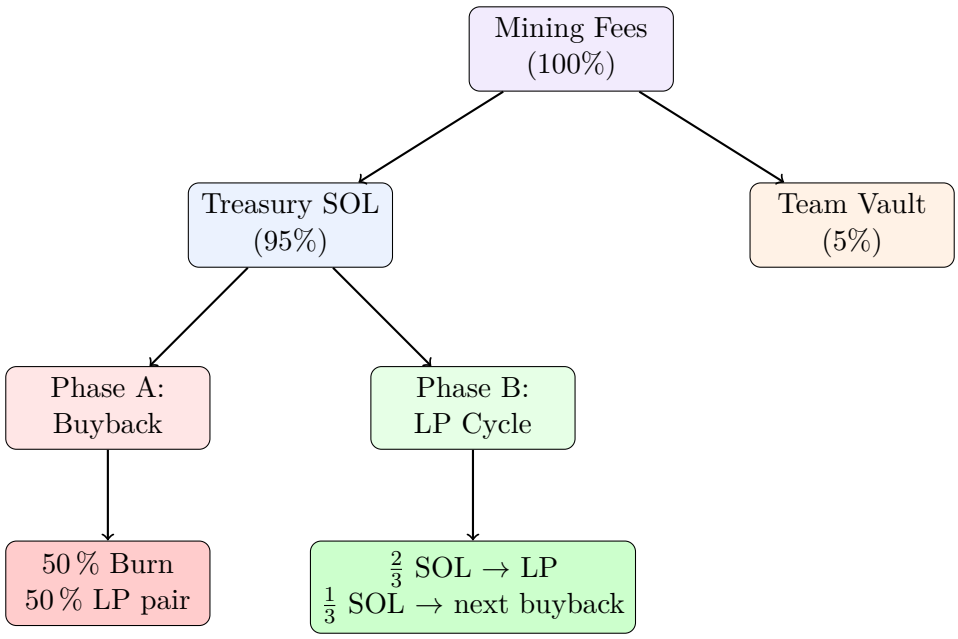

Figure 1: Fee distribution and treasury cycle. 5 % goes to the team vault; 95 % flows through an autonomous buyback / LP cycle. Each full cycle burns half of the tokens bought back, pairs the other half with wrapped SOL to add liquidity, and rolls one-third of the fresh SOL forward into the next buyback.

Treasury cycle mechanics. Phase A (`execute_buyback_cycle`) swaps accumulated SOL for HASHISH on the Meteora DAMM v2 pool. Phase B (`execute_lp_cycle`) burns 50 % of the bought tokens, pairs the remaining 50 % with $\frac{2}{3}$ of the fresh SOL (wrapped to wSOL) to add liquidity, and keeps $\frac{1}{3}$ of the SOL as seed capital for the next buyback. Slippage protection starts at 1 % and escalates by 0.5 % up to three retries per cycle.

5.5 Deflationary Dynamics

The protocol is structurally deflationary through three mechanisms:

1. **Buyback Burns**: 50 % of tokens acquired via treasury SOL buyback are permanently burned. Buybacks are funded by roughly $0.95 \times \frac{2}{3} \approx 63\,\%$ of all mining fees, compounding over every treasury cycle.

2. **Transfer Tax Burns**: A 0.01 % (1 bp) SPL Token-2022 transfer tax is collected on every HASHISH transfer through a dedicated `transfer-hook` program. Half of the collected tax is burned; the other half is credited to `pending_reward_tokens` and paid out to miners on the next block.

3. **Diminishing Emissions**: The per-block reward decays exponentially (factor 0.999999943 per block), so block rewards asymptote to the reward floor.

Cumulative burns feed back into the fee curve through E in Equation 5: each burn already counted in the emission history keeps the fee on its geometric trajectory, so the cost of new entry scales with total protocol activity rather than with circulating supply. Over time the combined burn rate surpasses the decaying emission, producing net deflation of circulating supply.

5.5.1 SPL Token-2022 Transfer Tax

The HASHISH mint is a SPL Token-2022 mint with the `TransferFeeConfig` and `TransferHook` extensions enabled. On every transfer:

1. The Token-2022 runtime withholds 1 basis point of the transferred amount into the recipient account as an `epoch-scoped fee`.

2. A separate `collect_transfer_fees` instruction harvests these withheld fees from all holder accounts into the protocol fee vaults, where they are split 50/50 between a permanent burn and the miner reward queue (`pending_reward_tokens`), recorded symmetrically across both pools.

3. The `transfer-hook` program enforces that no on-transfer bookkeeping can be bypassed and exposes a cranker incentive so that any caller can permissionlessly trigger fee harvesting.

6 Privacy-Preserving Mining with Arcium MPC

6.1 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.

6.2 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.

6.3 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 8: Encrypted fields and their encoding.

6.4 MPC Computation Circuits

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

6.4.1 `store_claim`

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

6.4.2 `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.

6.4.3 `deposit_fee`

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

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

6.4.4 `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:

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

6.4.5 `withdraw_fee`

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

6.4.6 `check_miner_balance`

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

6.5 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: $\mathcal{E}_s(\text{dest}) \rightarrow [\texttt{u64}; 4]$
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

6.6 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 9: Privacy guarantees of the Arcium integration.

6.7 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.

7 On-Chain Verification

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

1. **Hash Verification**: Recomputes SHA-256(`challenge`$\|$`miner`$\|n\|$`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.

8 Security Considerations

8.1 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.

8.2 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.

8.3 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.

8.4 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.

9 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	100
Pools	Protocol-level pools	2 (normal + seeker)
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	At 0 % emission	0.001 SOL
Fee Curve	Geometric in $E/S_{\max}$	$0.001 \times 1000^{E/S_{\max}}$
Fee Cap	At 100 % cumulative emission	1 SOL
Team Allocation	Of total fees	5 %
Treasury Allocation	Of total fees	95 %
Buyback Burn Ratio	Of tokens bought back	50 %
LP SOL Ratio	SOL wrapped into LP per cycle	2/3
Blocks per Cycle	Cycle gate trigger	10
Transfer Tax (Token-2022)	On every HASHISH transfer	0.01 % (1 bp)
Transfer Tax Split	Of collected tax	50 % burn / 50 % to miners
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 10: Complete protocol parameter table.

10 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 by reserving a full share of emissions for attested mobile devices alongside the open compute pool. 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, an emission-indexed geometric fee curve, automated buyback-and-burn via `pow-treasury`, and a perpetual SPL Token-2022 transfer tax — 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.