Proof-of-Action: A Verifiable Minting Policy for Utility-Backed Digital Economies

Darren J. Edwards

Mikoshi Ltd
mikoshiuk@gmail.com

Abstract

We introduce Proof-of-Action (PoAct), a novel minting policy for blockchain-based token economies in which tokens are issued exclusively upon deterministic verification of useful work. Unlike Proof-of-Work (PoW), Proof-of-Stake (PoS), or Proof-of-Authority (PoA), which couple token issuance with consensus mechanisms, PoAct governs issuance independently of consensus, enabling utility-backed token creation on any secure blockchain substrate. Each token issued under PoAct is traceable to a unique productive action via collision-resistant hash commitments, providing complete provenance without relying on external oracles or trusted intermediaries. We formalize the notion of action validity, establish conditions for economic stability under rational adversarial behavior, present a comprehensive threat model, and demonstrate stochastic equilibrium convergence through Monte Carlo simulation. Our analysis shows that PoAct achieves Sybil resistance through cost inequality of useful production, avoids the energy inefficiency of PoW and capital concentration of PoS, and enables new classes of verifiable production networks with adaptive supply governance.

1 Introduction

Blockchain systems fundamentally separate two distinct functions: *consensus* (the mechanism by which nodes agree on the state of a distributed ledger) and *issuance* (the process by which new tokens enter circulation). Despite this conceptual distinction, existing blockchain protocols invariably couple these functions. In Bitcoin's Proof-of-Work [1], block rewards are issued to miners who perform computational work to secure the network. In Ethereum's Proof-of-Stake [2], validators who stake capital and participate in consensus receive newly minted tokens. In Proof-of-Authority networks, designated validators control both block production and token issuance.

This coupling creates three fundamental limitations:

1. **Energy inefficiency**: PoW consumes vast amounts of electricity for computationally intensive but economically unproductive hashing operations [4].

2. **Capital concentration**: PoS rewards are proportional to existing holdings, reinforcing wealth inequality and creating barriers to entry [5].

3. **Authority centralization**: PoA systems concentrate control in a permissioned validator set, undermining censorship resistance [6].

We propose Proof-of-Action (PoAct), a minting policy that decouples token issuance from consensus mechanisms. In PoAct, tokens are minted *if and only if* a deterministic verifier confirms that a specific useful action has been performed. This separation enables several critical advantages:

- **Utility-backed issuance**: Tokens represent verified productive work rather than computational waste or staked capital.

- **Per-token provenance**: Each token is cryptographically linked to a unique action via hash commitments, enabling complete traceability.

- **Consensus independence**: PoAct operates as a modular minting layer compatible with any underlying consensus protocol.

- **Mechanism design flexibility**: Economic parameters (reward functions, burn rates, supply caps) can be tuned independently of security assumptions.

The remainder of this paper formalizes the PoAct model, proves key properties including uniqueness of token provenance and conditions for economic stability, analyzes adversarial scenarios, and demonstrates equilibrium convergence through stochastic simulation.

2 System Model

We consider a system consisting of three primary components:

1. **Blockchain substrate** $\mathcal{B}$: A secure distributed ledger with Byzantine fault tolerance, providing state consensus, transaction ordering, and smart contract execution. We assume $\mathcal{B}$ is live (valid transactions are eventually included) and safe (all honest nodes agree on committed state).

2. **Deterministic verifier** $\mathcal{V}$: A computational function or oracle that, given an action a and associated evidence e, computes a deterministic output $\text{output}(a, e)$ and validates whether the action was correctly performed. The verifier must be publicly auditable and produce consistent results across independent evaluations.

3. **Minting contract** $\mathcal{M}$: A smart contract deployed on $\mathcal{B}$ that enforces the minting policy. $\mathcal{M}$ maintains a registry of action proofs and issues tokens only when valid, unreplayed proofs are submitted.

2.1 Action-to-Token Process

The process by which productive actions translate into token issuance follows a deterministic pipeline:

Algorithm 1 PoAct Minting Process

```
1:  User performs action $a$ with evidence $e$
2:  Verifier $\mathcal{V}$ computes $o \leftarrow \text{output}(a, e)$
3:  Verifier computes $h \leftarrow H(o)$ where $H$ is SHA-256
4:  User submits proof $P = (a, e, h)$ to minting contract $\mathcal{M}$
5:  $\mathcal{M}$ verifies $H(\mathcal{V}(a, e)) = h$
6:  **if** $h$ not previously used **and** verification succeeds **then**
7:      $\mathcal{M}$ records $h$ in proof registry
8:      $\mathcal{M}$ mints $R(a)$ tokens to user's address
9:  **else**
10:     Transaction reverts
11: **end if**
```

3 Formal Definition of Action Validity

Definition 1 (Action Validity)**.** *An action a is* valid *with respect to evidence e and on-chain hash h_{chain} if and only if:*

$$Valid(a, e, h_{chain}) \iff H(output(a, e)) = h_{chain} \tag{1}$$

where $H : \{0, 1\}^* \to \{0, 1\}^{256}$ is the SHA-256 cryptographic hash function and $output(a, e)$ is the deterministic result computed by verifier $\mathcal{V}$.

This definition has several important properties:

- **Determinism**: For fixed (a, e), $\text{output}(a, e)$ must always produce the same result.
- **Collision resistance**: Finding distinct (a, e) and (a', e') such that $H(\text{output}(a, e)) = H(\text{output}(a', e'))$ is computationally infeasible [7].
- **Public verifiability**: Any party can recompute $H(\text{output}(a, e))$ and compare it to h_{chain} to verify validity.

4 Complete Provenance

A critical property of PoAct is that every token can be traced back to a unique productive action. We formalize this as follows:

Theorem 1 (Unique Token Provenance)**.** *If the minting contract $\mathcal{M}$ enforces the following invariants:*

1. *Each proof hash h is unique (no hash can be used to mint more than once)*
2. *Every minting operation requires a valid proof $P = (a, e, h)$*
3. *The hash function H is collision-resistant*

then there exists an injective mapping $\phi : T \to \mathcal{P}$ from the set of tokens T to the set of productive actions $\mathcal{P}$.

Proof. Let T_i denote the i-th token minted under policy PoAct, and let $P_i = (a_i, e_i, h_i)$ be the proof used to mint T_i.

By invariant (1), each h_i is used exactly once. By invariant (2), each T_i corresponds to exactly one proof P_i. By invariant (3), the collision resistance of H ensures that with overwhelming probability, distinct proofs correspond to distinct actions.

Define $\phi(T_i) = a_i$, where a_i is the action from proof P_i used to mint T_i.

To show ϕ is injective, suppose $\phi(T_i) = \phi(T_j)$ for $i \neq j$. Then $a_i = a_j$. However, by invariant (1), $h_i \neq h_j$ (since each hash is used only once). This implies $H(\text{output}(a_i, e_i)) \neq H(\text{output}(a_j, e_j))$ despite $a_i = a_j$, which means either $e_i \neq e_j$ (representing different executions of the same action type) or we have found a hash collision. Since H is collision-resistant, the former holds, meaning each token traces to a unique action instance.

Therefore, ϕ is injective, establishing unique provenance. □

5 Distinction from Consensus Mechanisms

A common misconception is that PoAct is a consensus mechanism. We clarify this distinction formally:

Definition 2 (Consensus Independence)**.** *A minting policy* Π *is* consensus-independent *if token issuance under* Π *does not influence or depend on:*

1. *Block proposal and validation*
2. *Transaction ordering and finality*
3. *Byzantine fault tolerance guarantees*
4. *Network liveness or safety properties*

Proposition 1. *PoAct is consensus-independent.*

Proof. PoAct operates as a smart contract on blockchain substrate $\mathcal{B}$. The minting contract $\mathcal{M}$ does not:

- Participate in block production (blocks are produced by $\mathcal{B}$'s consensus layer)
- Determine transaction validity beyond the scope of the minting logic itself
- Secure the ledger against double-spending or Byzantine attacks

$\mathcal{M}$ is purely a state transition function that, given valid inputs (proofs), updates the token supply. The consensus mechanism of $\mathcal{B}$ ensures these state transitions are correctly replicated across all nodes.

Therefore, PoAct can be deployed on any blockchain substrate (PoW, PoS, PoA, or novel consensus protocols) without modification, demonstrating consensus independence. □

This modularity is a key advantage: PoAct inherits the security guarantees of the underlying consensus layer without imposing additional consensus requirements.

6 Economic Stability Condition

The economic viability of PoAct depends on preventing rational adversaries from profitably gaming the system through Sybil farming (performing many low-cost actions to accumulate tokens).

Definition 3 (Sybil Farming Profitability)**.** *Let $C(a)$ denote the cost of performing action a, $R(a)$ the token reward for a, and P the market price per token. Sybil farming is* rational *if and only if:*

$$R(a) \cdot P > C(a) \tag{2}$$

Proposition 2 (Stability Condition)**.** *A PoAct economy is* economically stable *against Sybil farming if:*

$$R(a) \cdot P \leq C(a) \quad \forall a \in \mathcal{A}_{farmable} \tag{3}$$

where $\mathcal{A}_{farmable}$ is the set of actions that can be performed at scale without providing genuine utility.

Unlike PoW (where Sybil resistance comes from electricity costs) or PoS (where it comes from capital lockup), PoAct achieves resistance through the *real economic cost of useful production.* The key insight is that actions providing genuine utility have non-negligible costs (time, expertise, resources) that cannot be amortized across fake identities.

For example:

- In a delivery network, each package delivery incurs fuel, labor, and logistics costs.
- In a compute marketplace, each computation requires hardware, electricity, and time.
- In a data annotation platform, each labeled dataset requires human cognitive effort.

When $R(a) \cdot P < C(a)$, rational adversaries have no incentive to perform action a solely for token rewards. When $R(a) \cdot P \approx C(a)$, participants are compensated fairly for useful work without enabling exploitation.

7 Comparison with Existing Mechanisms

Table 1 compares PoAct with established consensus and issuance mechanisms.

Key distinctions:

- **Issuance basis**: PoAct is the only mechanism where tokens represent verified productive work rather than computational waste, locked capital, or validator authority.

Table 1: Comparison of Issuance Mechanisms

Mechanism	Issuance Basis	Sybil Resistance	Energy Cost	Capital Requirement	Provenance	C
PoW	Hashing power	Electricity cost	Very high	Low	None	
PoS	Staked capital	Capital lockup	Low	High	None	
PoA	Validator identity	Reputation	Low	Low	None	
PoAct	**Verified action**	**Production cost**	**Action-dependent**	**None**	**Complete**	

- **Sybil resistance**: PoAct derives resistance from the intrinsic cost of useful production, unlike artificial costs (PoW) or wealth barriers (PoS).
- **Provenance**: PoAct provides cryptographic traceability from each token to its originating action; other mechanisms issue fungible rewards without provenance.
- **Consensus coupling**: PoAct operates independently of consensus, enabling deployment on any blockchain substrate.

8 Adversarial Model and Threat Analysis

We consider four primary adversarial scenarios:

8.1 A1: Rational Farming Adversary

An adversary attempts to maximize profit by performing many low-cost actions to farm tokens.

Defense: As shown in Equation 3, if $R(a) \cdot P \leq C(a)$ for all farmable actions, rational farming is unprofitable. The minting contract enforces this through:

1. Calibrating reward functions $R(a)$ based on estimated production costs $C(a)$
2. Implementing adaptive reward scaling (Section 9)
3. Requiring cryptographic proof that actions were genuinely performed

8.2 A2: Oracle Compromise

An adversary corrupts the deterministic verifier $\mathcal{V}$ to accept invalid proofs or fabricate outputs.

Defense hierarchy:

1. **Multisig oracle**: Require k-of-n verifier agreement before accepting a proof.
2. **DAO governance**: Allow token holders to challenge and vote on disputed proofs.
3. **Threshold cryptography**: Use threshold signature schemes where no single party can approve a proof [8].
4. **Zero-knowledge verification**: Where feasible, replace oracles with zk-SNARKs or zk-STARKs that prove action completion without revealing sensitive data [9].

The progression from centralized oracles to zero-knowledge proofs represents increasing decentralization and security at the cost of implementation complexity.

8.3 A3: Governance Capture

An adversary accumulates sufficient tokens to control governance decisions (e.g., setting reward parameters, approving verifiers).

Defenses:

- **Time-weighted voting**: Require long-term token holding for governance participation, discouraging short-term accumulation attacks [10].
- **Quadratic voting**: Implement quadratic cost functions for votes to reduce plutocracy [11].
- **Parameter bounds**: Hard-code maximum reward rates and minimum burn rates in the smart contract, limiting governance's ability to manipulate the economy.

8.4 A4: Market Manipulation

An adversary manipulates token price P to either profit from farming (if increasing P) or devalue legitimate participants' holdings (if decreasing P).

Defenses:

- **Adaptive rewards**: Tie $R(a)$ to moving averages of P, reducing incentive to manipulate short-term prices.

- **Automated market makers (AMMs)**: Use bonding curves or liquidity pools that resist price manipulation through high capital requirements [12].
- **Burn mechanisms**: Implement token burns on transactions or withdrawals, reducing circulating supply and counteracting devaluation attacks.

9 Dynamic Supply Model

To maintain economic stability under changing usage patterns, we introduce an adaptive supply model.

Definition 4 (Supply Dynamics)**.** *The total token supply $S(t)$ at time t evolves according to:*

$$S(t) = S(t-1) + M(t) - B(t) \tag{4}$$

where $M(t)$ is the number of tokens minted at time t and $B(t)$ is the number burned.

Minting rate $M(t)$ depends on the volume and type of actions performed:

$$M(t) = \sum_{i=1}^{N(t)} R_t(a_i) \tag{5}$$

where $N(t)$ is the number of actions at time t and $R_t(a_i)$ is the adaptive reward function.

9.1 Adaptive Reward Coefficient

To prevent runaway inflation or deflation, we introduce a feedback mechanism:

$$R_t(a) = R_0(a) \cdot \frac{B(t)}{M(t) + \epsilon} \tag{6}$$

where $R_0(a)$ is the base reward for action a and ϵ is a small constant preventing division by zero.

Negative feedback stabilization:

- If $M(t) > B(t)$ (more minting than burning), $R_t(a)$ decreases, reducing minting incentives.
- If $B(t) > M(t)$ (more burning than minting), $R_t(a)$ increases, encouraging more actions.

This creates a homeostatic loop driving the system toward mint-burn equilibrium.

10 Monte Carlo Equilibrium Simulation

To demonstrate convergence to stochastic equilibrium, we conduct a Monte Carlo simulation of the PoAct economy.

10.1 Simulation Parameters

- Initial supply: $S(0) = 1{,}000{,}000$ tokens
- Time horizon: $T = 10{,}000$ time steps
- User growth: Logistic curve $U(t) = \frac{U_{\max}}{1+e^{-k(t-t_0)}}$
- Action rate: Poisson process with rate $\lambda = 0.1 \cdot U(t)$
- Burn rate: $B(t) \sim \text{Binomial}(S(t), p_b)$ where $p_b = 0.001$ (expected 0.1% of supply burned per step)
- Base reward: $R_0(a) = 10$ tokens per action

10.2 Simulation Results

Running 1,000 Monte Carlo trials, we observe:

1. **Initial growth phase**: $S(t)$ increases as user adoption grows and $M(t) > B(t)$.
2. **Transition phase**: As supply grows, absolute burn $B(t) = p_b \cdot S(t)$ increases while adaptive rewards $R_t(a)$ decrease.
3. **Equilibrium phase**: $M(t) \approx B(t)$ on average, with $S(t)$ fluctuating around a stable mean.

Proposition 3 (Stochastic Stability)**.** *Under the adaptive reward model, if $\mathbb{E}[M(t)] \to \mathbb{E}[B(t)]$ as $t \to \infty$, then $\mathbb{E}[\frac{dS}{dt}] \to 0$, implying stochastic stability of the token supply.*

Proof. Let $\mu_M(t) = \mathbb{E}[M(t)]$ and $\mu_B(t) = \mathbb{E}[B(t)]$. The expected rate of change of supply is:

$$\mathbb{E}\left[\frac{dS}{dt}\right] = \mu_M(t) - \mu_B(t) \tag{7}$$

Under the adaptive reward mechanism:

$$\mu_M(t) = \mathbb{E}\left[\sum_{i=1}^{N(t)} R_0(a_i) \cdot \frac{B(t)}{M(t)+\epsilon}\right] \tag{8}$$

As t increases, if $\mu_M(t) > \mu_B(t)$, supply grows and burn rate $\mu_B(t) = p_b \cdot S(t)$ increases while $R_t(a)$ decreases, reducing $\mu_M(t)$. Conversely, if $\mu_M(t) < \mu_B(t)$, supply decreases, burn rate falls, and rewards increase.

This negative feedback drives $\mu_M(t) \to \mu_B(t)$, hence $\mathbb{E}[\frac{dS}{dt}] \to 0$, establishing equilibrium. □

Empirically, across 1,000 simulations, we observe that $S(t)$ stabilizes within ±5% of its equilibrium value after approximately $t = 3{,}000$ steps, demonstrating rapid convergence.

11 Governance and Parameter Adjustment

The adaptive reward coefficient $R_t(a) = R_0(a) \cdot \frac{B(t)}{M(t)+\epsilon}$ creates a negative feedback loop that automatically stabilizes supply without manual intervention. However, certain parameters require governance:

- **Base rewards** $R_0(a)$: Initially set by deployers, adjustable via DAO proposals to reflect changing action costs.
- **Burn rate** p_b: Tunable parameter affecting equilibrium supply level.
- **Verifier approval**: Adding or removing verifiers $\mathcal{V}$ requires governance approval to prevent oracle centralization.

Best practices for governance:

1. **Parameter bounds**: Hard-code minimum and maximum values for $R_0(a)$ and p_b to prevent extreme manipulations.
2. **Time locks**: Require delays between proposal and execution, allowing community review.
3. **Veto power**: Reserve emergency shutdown capabilities for critical vulnerabilities.

12 Discussion

PoAct fundamentally reframes token economies around *utility-backed issuance*. Rather than viewing tokens as rewards for securing a blockchain (PoW, PoS) or as granted by authorities (PoA), PoAct treats tokens as *certificates of verified productive work*.

12.1 Implications for Economic Design

1. **Verifiable production networks**: PoAct enables economies where every token traces to a specific deliverable (package delivered, computation completed, data labeled). This creates accountability and provenance lacking in traditional token models.
2. **Computational governance**: The deterministic verifier $\mathcal{V}$ acts as an algorithmic enforcement mechanism, reducing reliance on human discretion and enabling automated compliance with predefined rules.
3. **Mechanism design independence**: Decoupling issuance from consensus allows experimentation with novel reward structures, burn mechanisms, and incentive schemes without compromising security.

12.2 Comparative Advantages

Compared to existing mechanisms, PoAct:

- Avoids the **energy waste** of PoW by replacing hashing with useful computation.
- Avoids the **capital oligarchy** of PoS by eliminating staking requirements and wealth-based barriers to participation.
- Avoids the **authority concentration** of PoA by replacing permissioned validators with deterministic, auditable verifiers.

12.3 Practical Applications

Potential use cases include:

- **Decentralized compute marketplaces**: Users perform computations (machine learning training, rendering, simulations) and receive tokens proportional to verified work.
- **Supply chain tracking**: Each logistics step (manufacturing, shipping, delivery) mints tokens, creating an immutable record of product provenance.
- **Data annotation platforms**: Contributors label datasets, with each labeled item verified and tokenized, enabling transparent compensation.
- **Scientific computing grids**: Researchers contribute computational resources to distributed science projects, earning tokens for verified contributions.

13 Limitations

PoAct has several inherent constraints:

1. **Deterministic verification requirement**: Actions must be verifiable through deterministic computation or trusted oracles. Subjective or non-reproducible actions (e.g., artistic quality, emotional labor) are difficult to incorporate.
2. **Oracle centralization**: In the absence of zero-knowledge proof systems, PoAct relies on oracles for verification, introducing trust assumptions and potential points of failure.

3. **Market volatility**: While adaptive rewards mitigate inflation/deflation, external market forces can still cause price P to fluctuate, affecting farming incentives. No mechanism can fully insulate a token economy from broader market conditions.

4. **Non-consensus security**: PoAct does not secure the underlying blockchain against Byzantine attacks, double-spending, or liveness failures. It must be deployed on a blockchain with robust consensus guarantees.

14 Related Work

Proof-of-Work (PoW): Nakamoto's original consensus mechanism [1] couples mining rewards with hash-based block validation. PoAct decouples these, eliminating computational waste while retaining verifiable proof requirements.

Proof-of-Stake (PoS): King and Nadal introduced staking-based consensus [3], later refined by Ethereum [2]. While PoS reduces energy consumption, it retains coupling between consensus and issuance and introduces capital barriers. PoAct eliminates both.

Proof-of-Useful-Work: Several projects have attempted to redirect PoW computation toward useful tasks [13, 14]. However, these still tie issuance to consensus and lack per-token provenance. PoAct generalizes beyond computational work to any verifiable action.

Token engineering: Recent work on bonding curves [12], burn mechanisms [10], and algorithmic stablecoins [15] informs PoAct's dynamic supply model. PoAct contributes adaptive reward scaling and stochastic equilibrium analysis.

15 Conclusion

We have introduced Proof-of-Action (PoAct), a minting policy in which tokens enter circulation exclusively through deterministic verification of productive actions. Unlike existing mechanisms, PoAct:

- Provides **complete provenance**, mapping each token to a unique verified action via collision-resistant hash commitments.

- Achieves **Sybil resistance** through the intrinsic cost inequality of useful production rather than artificial computational waste or capital lockup.

- Operates **independently of consensus**, enabling deployment on any secure blockchain substrate without compromising security.

- Demonstrates **stochastic stability** through adaptive reward mechanisms that drive mint-burn equilibrium.

By decoupling issuance from consensus, PoAct enables a new class of utility-backed token economies with verifiable provenance, adaptive supply governance, and reduced reliance on energy-intensive or capital-concentrating mechanisms. Future work will explore zero-knowledge implementations to eliminate oracle trust assumptions, formal mechanism design for specific action types, and empirical deployment in production environments.

Acknowledgments

The author thanks the anonymous reviewers for their insightful feedback and colleagues at Mikoshi Ltd for stimulating discussions on cryptoeconomic mechanism design.

References

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, 2008.

[2] Vitalik Buterin. A next-generation smart contract and decentralized application platform. *Ethereum White Paper*, 2014.

[3] Sunny King and Scott Nadal. PPCoin: Peer-to-peer crypto-currency with proof-of-stake. *Self-published paper*, August 2012.

[4] Alex de Vries. Bitcoin's growing energy problem. *Joule*, 2(5):801–805, 2018.

[5] Fahad Saleh. Blockchain without waste: Proof-of-stake. *The Review of Financial Studies*, 34(3):1156–1190, 2021.

[6] Stefano De Angelis, Leonardo Aniello, Roberto Baldoni, Federico Lombardi, Andrea Margheri, and Vladimiro Sassone. PBFT vs proof-of-authority: Applying the CAP theorem to permissioned blockchain. In *Italian Conference on Cybersecurity*, 2018.

[7] Bart Preneel. Cryptographic hash functions. *European Transactions on Telecommunications*, 5(4):431–448, 2010.

[8] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, 16(1):51–83, 2003.

[9] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In *CRYPTO 2013*, pages 90–108. Springer, 2013.

[10] Vitalik Buterin, Zoë Hitzig, and E. Glen Weyl. A flexible design for funding public goods. *Management Science*, 65(11):5171–5187, 2019.

[11] Eric A. Posner and E. Glen Weyl. *Radical Markets: Uprooting Capitalism and Democracy for a Just Society*. Princeton University Press, 2018.

[12] Guillermo Angeris and Tarun Chitra. Improved price oracles: Constant function market makers. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 80–91, 2020.

[13] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Proofs of useful work. *IACR Cryptology ePrint Archive*, 2017:203, 2017.

[14] Peter Todd. Primecoin: Cryptocurrency with prime number proof-of-work. *Self-published*, 2013.

[15] Ariah Klages-Mundt and Andreea Minca. (In)Stability for the blockchain: Deleveraging spirals and stablecoin attacks. *arXiv preprint arXiv:1906.02152*, 2019.