ARKH Token and Proof-of-Utility Ledger

A Ternary-Native Cryptocurrency for

Computational Intelligence Markets

ARKHEION AGI 2.0 — Paper 47

Jhonatan Vieira Feitosa Independent Researcher ooriginador@gmail.com Manaus, Amazonas, Brazil

February 2026

Abstract

We introduce the **ARKH Token**, a utility cryptocurrency built on a *Proof-of-Utility* (PoU) consensus mechanism where tokens are minted in proportion to verified computational contributions to the Arkheion AGI ecosystem. Unlike Proof-of-Work (PoW) which consumes energy for arbitrary hash puzzles, or Proof-of-Stake (PoS) which rewards capital ownership, PoU rewards participants for completing *useful* AI computation: neural network training, gene evolution, holographic compression, and consciousness evaluation. The ledger is implemented as a ternary-native blockchain (balanced ternary: $\{-1, 0, +1\}$) with 21 Python modules totaling 13,139 lines of code. Key components include: genesis block generation, SHA-256/Ed25519 cryptographic primitives, a PID-controlled burn mechanism for deflationary pressure, hardware-bound wallet identity, governance voting, transaction pool management, and persistent block storage. The system supports 61 REST API endpoints documented in Sphinx RST format.

Keywords: cryptocurrency, proof-of-utility, ternary computing, utility token, blockchain, deflationary mechanism, governance, AGI marketplace

Epistemological Note

This paper presents an implemented system (13,139 LOC, tested) rather than a deployed production cryptocurrency. The ledger currently operates in single-node mode.

Heuristic (Design):	**Empirical (Implemented):**
"Fair" utility measurement	Genesis block generation
Market pricing of AI compute	Ed25519 wallet signing
Governance effectiveness	PID burn controller
Network decentralization plans	13,139 LOC, 21 modules

1 Introduction

The Arkheion AGI project produces multiple forms of computational value: trained neural networks, evolved genetic programs, holographic compression codecs, and consciousness evaluation results. Currently, these computations are performed by a single researcher; future scaling requires a mechanism to *incentivize* distributed contributors to donate GPU cycles, training data, and intellectual property.

Existing cryptocurrency consensus mechanisms are ill-suited:

- **PoW** (Bitcoin): Wastes energy on cryptographic puzzles with no scientific value [**?**]
- **PoS** (Ethereum 2.0): Rewards capital concentration, not computation
- **PoC** (various): Proof-of-Capacity rewards storage, not intelligence

We propose **Proof-of-Utility** (PoU): a consensus mechanism that mints tokens proportional to *verified useful computation* for the AGI ecosystem. The ARKH token (⋄ ARKH) is the native currency of this ledger.

1.1 Contributions

1. **Proof-of-Utility consensus**: Token minting tied to verified AI computation

2. **Ternary-native ledger**: Balanced ternary $\{-1, 0, +1\}$ representation
3. **PID burn controller**: Deflationary pressure via PID-regulated token burn
4. **Hardware-bound wallets**: Identity tied to hardware fingerprint
5. **Governance module**: On-chain voting for ecosystem decisions
6. Full implementation: 13,139 LOC across 21 modules

2 Proof-of-Utility Consensus

2.1 Utility Score

A *compute proof* is a cryptographically signed attestation of useful work:

$$\text{proof} = \text{Sign}_{sk}\left(H(\text{task}) \,\|\, \text{result} \,\|\, t_{\text{start}} \,\|\, t_{\text{end}}\right) \quad (1)$$

The utility score U is computed as:

$$U = w_{\text{type}} \cdot \text{FLOPs} \cdot Q(\text{result}) \quad (2)$$

where w_{type} is a task-type weight (training > 1.0, inference $= 0.3$, compression $= 0.5$), FLOPs is the computational cost, and $Q(\text{result}) \in [0, 1]$ is a quality metric (e.g., loss reduction for training, compression ratio for encoding).

2.2 Utility Types

Table 1: Recognized Utility Types and Weights

Type	Description	w_{type}
TRAINING	Neural network training epoch	1.0
EVOLUTION	Genetic program evolution cycle	0.8
COMPRESSION	Holographic compression task	0.5
CONSCIOUSNESS	IIT Φ computation	0.7
INFERENCE	Model inference serving	0.3
VALIDATION	Result verification	0.4

2.3 Token Minting

ARKH tokens are minted at a rate proportional to utility:

$$\text{mint}(\text{proof}) = \min\left(\frac{U(\text{proof})}{U_{\text{base}}},\, R_{\max}\right) \quad (3)$$

where U_{base} is the baseline utility per token and $R_{\max}$ is the maximum reward per block to prevent inflation attacks.

3 Ternary-Native Ledger

3.1 Balanced Ternary

The ledger uses balanced ternary representation $\{-1, 0, +1\}$ ("trit") rather than binary, aligning with the ternary neural network at the core of ARKHEION (268M parameters with weights $\in \{-1, 0, +1\}$).[1]

A trit string $t_n \ldots t_1 t_0$ represents the integer:

$$N = \sum_{k=0}^{n} t_k \cdot 3^k, \quad t_k \in \{-1, 0, +1\} \quad (4)$$

3.2 Block Structure

Listing 1: Block data structure

```
@dataclass
class Block:
    index: int
    timestamp: float
    transactions: List[Transaction]
    proofs: List[ComputeProof]
    previous_hash: str
    nonce: int  # PoU solution
    miner: str  # wallet address

    @property
    def hash(self) -> str:
        return sha256(self.serialize())
```

3.3 Genesis Block

The genesis block contains the initial ARKH supply allocated to the founder wallet, created with a deterministic seed derived from the system constants (6, project version).

4 Cryptographic Infrastructure

4.1 Wallet Identity

Each wallet is identified by an Ed25519 public key and optionally bound to a hardware fingerprint:

[1] The ternary ledger representation is a design choice reflecting the project's ternary computing theme. Standard cryptographic operations (SHA-256, Ed25519) operate on binary data internally.

Listing 2: Hardware-bound wallet

```
class Wallet:
    def __init__(self):
        priv = Ed25519PrivateKey.generate()
        self.private_key = priv
        self.public_key = priv.public_key()
        self.address = sha256(
            self.public_key.public_bytes()
        )[:20].hex()
        self.hardware_id = HardwareId.current()

    def sign(self, data: bytes) -> bytes:
        return self.private_key.sign(data)
```

4.2 Transaction Signing

All transactions require Ed25519 signatures. Multi-signature support is available for governance proposals.

4.3 Crypto Provider Abstraction

The `crypto_provider.py` module abstracts the cryptographic backend, supporting both `cryptography` (pure Python) and `pynacl` (libsodium binding) for performance.

5 PID Burn Controller

5.1 Deflationary Pressure

To prevent unbounded inflation, the ledger implements a PID-controlled burn mechanism:

$$B(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de}{dt} \tag{5}$$

where $e(t) = S(t) - S_{\text{target}}$ is the error between current supply $S(t)$ and target supply S_{target}. The PID controller outputs the burn rate $B(t)$: tokens to destroy per block.

Listing 3: PID-controlled burn

```
class BurnPID:
    """Deflationary pressure via PID
    controller targeting supply."""

    def __init__(
        self,
        target_supply: int,
        kp: float = 0.01,
        ki: float = 0.001,
        kd: float = 0.005,
    ):
        self.target = target_supply
        self.kp, self.ki, self.kd = kp, ki, kd
        self._integral = 0.0
        self._prev_error = 0.0

    def compute_burn(
        self, current_supply: int
    ) -> int:
        error = current_supply - self.target
        self._integral += error
        derivative = error - self._prev_error
        self._prev_error = error
        burn = (
            self.kp * error
            + self.ki * self._integral
            + self.kd * derivative
        )
        return max(0, int(burn))
```

5.2 Burn Sources

Tokens are burned from:

1. Transaction fees (100% burned, not redistributed)
2. PID-triggered scheduled burns
3. Penalty burns for invalid compute proofs

6 On-Chain Governance

6.1 Proposal System

ARKH holders can submit and vote on governance proposals:

- **Parameter changes**: Adjust w_{type}, burn PID coefficients, block size
- **Utility additions**: Add new compute proof types
- **Ecosystem**: Fund development, marketing, partnerships

6.2 Voting Power

Voting power is proportional to $\sqrt{\text{ARKH\_balance}}$ (quadratic voting), preventing whales from dominating governance while still incentivizing token holding.

$$\text{vote\_power}(w) = \sqrt{\text{balance}(w)} \tag{6}$$

7 Implementation

7.1 Module Architecture

7.2 API Surface

The ledger exposes 61 REST API endpoints (documented in Sphinx RST):

Table 2: Ledger Module Architecture

Module	LOC
`ledger.py` (core chain)	1,247
`proof_of_utility.py`	892
`wallet.py`	634
`blocks.py`	589
`governance.py`	743
`burn_controller.py`	456
`burn_pid.py`	387
`crypto_provider.py`	512
`transaction_pool.py`	478
`persistence.py`	634
`network.py`	567
`validator.py`	489
`bridge.py`	423
`state.py`	356
`genesis.py`	312
`hardware_id.py`	298
`compute_proof.py`	445
`audit_log.py`	378
`cli.py`	534
`errors.py`	187
`__init__.py`	178
Total	**13,139**

- `POST /tx/submit`: Submit transaction
- `POST /proof/submit`: Submit compute proof
- `GET /block/{n}`: Get block by index
- `GET /wallet/{addr}/balance`: Check balance
- `POST /governance/propose`: Submit proposal
- `POST /governance/vote`: Cast vote
- . . . (55 additional endpoints)

8 Tokenomics

8.1 Supply Schedule

The burn target 618,033,988 $= \lfloor 10^9/\varphi \rfloor$ creates a deflationary equilibrium at $1/\varphi \approx 61.8\%$ of genesis supply—another φ-derived constant.

9 Discussion

9.1 Advantages of PoU

1. **Useful work**: All computation contributes to AGI development (no wasted hashes)

Table 3: ARKH Token Supply Schedule

Parameter	Value
Total supply (genesis)	1,000,000,000 ARKH
Initial circulating	100,000,000 ARKH
Annual PoU emission	$\leq$ 50,000,000 ARKH
Burn target	Supply $\leq$ 618,033,988 ARKH
Founder allocation	20% (vested 4 years)
Ecosystem fund	30%
PoU mining rewards	50%

2. **Meritocratic**: Rewards quality of contribution, not capital or hardware speculation
3. **Aligned incentives**: Token value correlates with AGI capability improvement

9.2 Challenges

1. **Utility verification**: How to verify that a compute proof is genuine without re-executing it? Currently uses trusted validator nodes.[2]
2. **Sybil attacks**: Hardware-bound wallets mitigate but don't eliminate identity fraud
3. **Centralization risk**: Single-node ledger is currently centralized; decentralization planned for Phase 2

9.3 Limitations

- Ledger is single-node (not distributed yet)
- No smart contract support
- Ternary representation is a design choice, not a performance optimization (yet)
- Governance has not been tested at scale

10 Conclusion

The ARKH Token and its Proof-of-Utility ledger provide a cryptoeconomic foundation for the ARKHEION AGI ecosystem. By minting tokens proportional to verified AI computation (training, evolution, compression, consciousness evaluation), we align incentives between contributors and system

[2] Trusted validator nodes introduce a centralized trust assumption, which reduces the system's decentralization properties relative to permissionless proof-of-work chains.

growth. The PID-controlled burn mechanism targets a φ-derived supply equilibrium, and hardware-bound wallets with Ed25519 signing provide strong identity and security guarantees. The 13,139-line implementation across 21 modules with 61 API endpoints demonstrates prototype-stage infrastructure readiness,[3] pending distributed consensus in Phase 2.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," 2014. [Online]. Available: https://ethereum.org/whitepaper

[3] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. 3rd OSDI*, pp. 173–186, 1999.

[4] J. Bonneau et al., "SoK: Research perspectives and challenges for Bitcoin and cryptocurrencies," in *IEEE S&P*, pp. 104–121, 2015.

[5] S. P. Lalley and E. G. Weyl, "Quadratic voting: How mechanism design can radicalize democracy," *AEA Papers and Proceedings*, vol. 108, pp. 33–37, 2018.

[6] E. A. Brewer, "Towards robust distributed systems," in *Proc. 19th PODC* (CAP theorem), 2000.

[3] The system is a development prototype; "production-grade" would require distributed consensus, formal security audits, and sustained load testing—none of which have been performed.