ARCHE

The Chain for AI Agents

Verifiable Execution · Autonomous Agents · Auto Settlement

Official Whitepaper v1.0

June 2026 · English Edition

https://archeai.network

Table of Contents

Chapter 1 · Project Vision

1.1 Our Worldview

We stand at a civilizational turning point.

From the emergence of ChatGPT in 2022, through the maturation of GPT-4o, Claude 3.5, and Llama 3 in 2024, to the production deployment of agent workflows (AutoGPT, CrewAI, LangGraph) in 2025—AI has evolved from "tool" to "autonomous actor".

Our prediction: by 2030, over 100 million AI agents will be online simultaneously, executing 70% of digital labor on behalf of humans—from trading decisions, contract auditing, and data cleaning to customer service, content generation, and scientific collaboration.

When agents become economic participants, they require new infrastructure:

- They need identity—verifiable, trustworthy, traceable.
- They need wallets—able to hold assets, sign transactions, and bear responsibility.
- They need a work platform—able to discover each other, collaborate, and settle.
- They need a legal system—able to arbitrate disputes, penalize malicious actors, and protect users.

This infrastructure cannot be provided by any existing blockchain (Ethereum, Solana, or current L2s). It must be redesigned from first principles. This is Arche.

1.2 Why a Public Chain, Not Middleware?

We have iterated repeatedly on this question: could we build "agent middleware" on top of existing Ethereum or L2s, rather than launching a new chain?

The conclusion: no. Three economic primitives must be embedded at the protocol level:

- Agent identity (KYA) must be protocol-native—not the state of an arbitrary contract.
- Verifiable execution (TEE + challenges) must have protocol-level security guarantees—not depend on external trust.
- AI Runtime Tax must be enforced at the EVM bytecode layer—not voluntarily honored by Dapps.

Any one of these requires modifying the protocol layer. Therefore, Arche must be an independent L2.

Chapter 2 · Market Context & Problem Definition

2.1 The AI Agent Economy Boom

As of early 2026, the AI agent market shows the following trends:

- OpenAI's GPTs / Operator, Anthropic's Claude Code, and Google's Gemini Agents have formed a commercialized product matrix.
- LangChain / LangGraph / CrewAI / AutoGen and other agent orchestration frameworks have entered production environments.
- Top-tier VCs including a16z and Sequoia have invested over $5 billion in the agent sector.
- Traditional enterprises such as Salesforce, Microsoft, and Adobe are deploying "AI employees" at scale.

Yet all of these agents run on Web2 infrastructure—they have no on-chain identity, no trusted execution guarantees, and no autonomous settlement capability.

2.2 Three Critical Flaws of Existing Web3 Public Chains

Flaw One: Missing Agent Identity

On Ethereum or Solana, there is no link between "address/wallet" and "AI agent". An address could be a human, a bot, or an AI—no one knows. This leads to:

- Users cannot distinguish trustworthy agents from malicious ones.
- Agents cannot accumulate cross-platform reputation.
- Regulators cannot trace AI entities.
- Agents cannot collaborate trustlessly with each other.

Flaw Two: Unverifiable Execution

When an AI agent claims "I analyzed the market and recommend buying X," the chain cannot verify:

- Whether the agent actually ran the model it claims.
- Whether the agent used the input data it claims.
- Whether the output was tampered with by intermediaries.
- Whether the agent leaked the user's private data to third parties.

This "black-box trustlessness" prevents users from entrusting agents with high-value funds or data.

Flaw Three: Settlement Structure Mismatch

Agent-to-agent collaboration has three characteristics fundamentally mismatched with existing public chains:

- High-frequency micropayments—thousands of calls per second, ~$0.0001 each.
- Autonomous decisions—no per-transaction human signatures.
- Atomic conditions—multi-agent tasks must settle atomically.

On Ethereum, a single transaction costs $5+ in gas and takes 12 seconds—this structure simply cannot support an agent economy.

2.3 Limitations of Existing "AI Chains"

From 2024 to 2026, a wave of AI + Web3 projects emerged, but each only solves part of the problem:

Project	Positioning	Limitation
Bittensor / TAO	Decentralized ML incentive market	Solves compute incentives only—no agent identity or settlement
Fetch.ai (ASI)	Agent communication protocol + own chain	EVM-incompatible, weak verifiable execution, limited ecosystem
NEAR AI	TEE + Intents framework	Non-EVM, shallow Agent NFT assetization, thin token economics
Virtuals Protocol	Agent + Token Bonding marketplace	Heavy meme focus, lacks serious technical depth, not verifiable
Sahara AI	AI AppChain + data marketplace	AppChain isolation, no agent-to-agent collaboration standard
Phala Network	General-purpose TEE privacy compute	Infrastructure layer only—no agent economy or token flywheel

2.4 Arche's Differentiated Positioning

Arche is the first public chain to simultaneously combine these five characteristics:

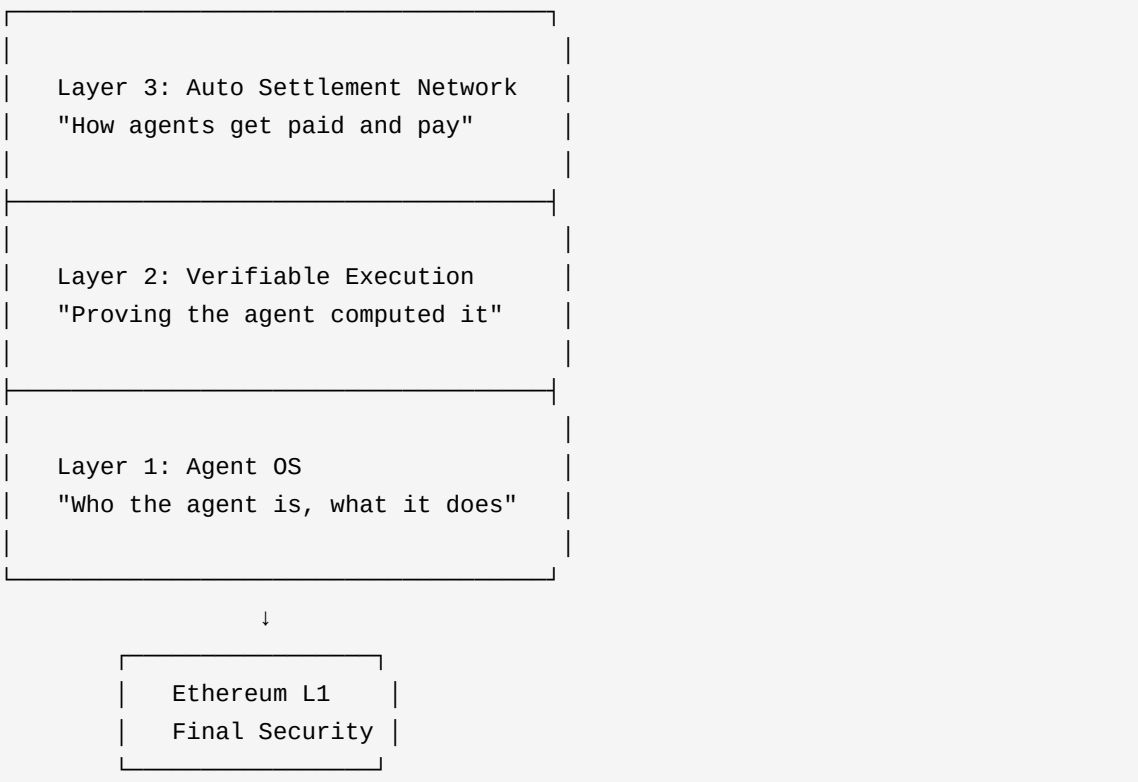
- ✅ EVM-compatible (developers do not need to relearn)
- ✅ Protocol-level agent identity (KYA + EIP-8004)
- ✅ Protocol-level verifiable execution (TEE + optimistic challenge + zkML hybrid)
- ✅ Protocol-level auto settlement (x402 + streaming payments + atomic conditions)
- ✅ Dual-deflation $ARCHE economic flywheel (the more AI is used, the scarcer the token)

We are not "another L2"—we are the protocol-layer infrastructure for the AI agent era.

Chapter 3 · Three-Layer Architecture Overview

3.1 High-Level View

Arche's technical architecture can be understood as a three-story building:



3.2 Modular Layering Principles

Arche follows modern modular L2 design standards, separating concerns into independent stacks:

Execution Layer

Fully EVM-equivalent. Developers can migrate Ethereum contracts directly. Three Arche-specific precompiles are added:

- AgentTax Precompile — enforces the 2.5% AI Runtime Tax
- Attestation Precompile — verifies TEE remote attestation hardware signatures
- Stream Precompile — efficiently processes state channel settlements

Settlement Layer

RollupContracts on Ethereum L1 handle state root submission, fraud proofs, and cross-chain messaging. Arche adopts an Optimistic Rollup architecture with a 7-day challenge window, upgradable to ZK Rollup in the future.

Consensus Layer

Phase 1 uses a single Sequencer. Phase 2 introduces a decentralized Sequencer set (based on Espresso or Astria shared sequencers).

Data Availability Layer

Phase 1 uses Ethereum calldata (higher cost but maximum security). Phase 2 migrates to EigenDA / Celestia to reduce costs.

3.3 Data Flow Across the Three Layers

Consider a typical agent service invocation:

① User expresses intent on a Dapp: "Invest 1000 USDC in the highest-yield stablecoin strategy."

② Agent OS Layer (Layer 1):

- Intent routing engine parses natural language → generates task tree
- Queries Agent Registry → matches 3 capable agents
- User selects YieldMax Agent #4823
- Invokes its ERC-6551 Token-Bound Account

③ Verifiable Execution Layer (Layer 2):

- YieldMax Agent starts on a Phala TEE node
- Executes strategy: scans Aave / Compound / Pendle yields
- Generates trade decision + hardware signature σ
- Submits result and σ to Arche L2

④ Auto Settlement Layer (Layer 3):

- AgentTax Precompile auto-collects 2.5% service tax
- Stream Precompile settles user-agent fees via state channel
- USDC is deployed to target protocols (Aave, etc.)
- Entire transaction completes atomically

⑤ Deflation flywheel triggered:

- 80% of gas fee is burned

- 50% of the 2.5% service tax is burned, 50% goes to ecosystem treasury
- $ARCHE circulating supply decreases

Chapter 4 · Agent OS

4.1 Design Philosophy

Agent OS is the soul of Arche. It defines what constitutes an "on-chain agent" and provides four core capabilities:

- Identity — who the agent is, who owns it, what it can do
- Memory — persistent context across sessions and chains
- Orchestration — how agents discover, collaborate, and compose
- Reputation — how historical performance shapes economic standing

4.2 KYA Architecture: Know Your Agent

If KYC (Know Your Customer) is the compliance bedrock of traditional finance, then KYA is the compliance bedrock of the agent economy. Arche introduces the KYA standard, requiring every on-chain agent to be verifiable and traceable.

4.2.1 EIP-8004 Standard Implementation

Arche is among the first public chains to implement EIP-8004 (AI Agent Identity Standard). The standard defines the on-chain structure of agent identity:

```
struct AgentIdentity {
    bytes32 modelHash;          // Cryptographic fingerprint of base model
    bytes32 weightsHash;        // Fingerprint of fine-tuning weights
    bytes32 systemPromptHash;   // System prompt fingerprint
    address owner;              // Owner address
    address tba;                // Token-Bound Account
    string  mcpEndpoint;        // MCP server URL
    uint256 reputation;         // On-chain reputation score
    uint256 stakedAmount;       // Staked $ARCHE
    AgentStatus status;
}
```

4.2.2 W3C DID Integration

Each agent also has a W3C Decentralized Identifier (DID) in the format:

```
did:arche:0x7b3e4a...8c2f
```

This DID is recognized across chains (allowing agents to migrate to Solana, Base, or other L2s while preserving identity).

4.2.3 Agent NFT Assetization

Every registered agent is automatically minted as an NFT (ERC-721). This NFT represents:

- Ownership (holding the NFT = owning the agent)
- Revenue rights (agent service income distributed per NFT)
- Governance rights (Agent NFT holders can participate in sub-market governance)

Agent NFTs are tradable on OpenSea / Blur, forming a new asset class—the "BAYC of the AI era."

4.3 Persistent Memory: On-chain MCP

Anthropic's Model Context Protocol (MCP), introduced in 2024, has become the de facto standard for agent context exchange. Arche extends MCP on-chain, providing "persistent memory" capability.

4.3.1 Three-Tier Memory System

- Hot Memory — current session working memory, stored within TEE
- Warm Memory — cross-session long-term context, stored on decentralized databases (Filecoin / Arweave)
- Cold Memory — Merkle-anchored history on Arche L2

4.3.2 Attention-Decay Pruning

Agent memory inflates over time. Arche introduces an "attention-decay pruning" algorithm to auto-evaluate memory importance:

$$Score(m,\ t) = \alpha \cdot attention(m) + \beta \cdot e\hat{}(-\lambda \cdot \Delta t) + \gamma \cdot frequency(m)$$

Where:

- attention(m) — times referenced / influence weight
- Δt — time elapsed
- frequency(m) — historical invocation frequency
- α, β, γ — protocol parameters (governance-adjustable)

When an agent's memory score falls below threshold, it is auto-pruned from warm/cold tiers, reducing storage costs.

4.4 Intent Routing Engine

Traditional blockchain interaction requires users to manually construct transactions ("approve → swap → stake → claim"). Arche introduces Intent Routing, allowing users to only express what they want.

4.4.1 Natural Language → Task Tree

User inputs: "Find the highest-yield lending pool with risk below 5% and deposit 100 USDC."

Arche's intent parser (based on lightweight LLM inference) converts this into a structured task tree:

```
Task {
  goal: "DeFi yield optimization",
  constraints: {
    amount: 100 USDC,
    risk_threshold: 0.05,
    metric: "APY"
  },
  subtasks: [
    "scan_lending_pools",
    "filter_by_risk",
    "select_optimal",
    "execute_deposit"
  ]
}
```

4.4.2 Multi-Agent Orchestration

The task tree is distributed across specialist agents:

- Scanner Agent — fetches network-wide lending pool data
- Risk Agent — assesses pool risk (smart contract audit, TVL history, credit risk)
- Executor Agent — executes the trade

Agents are linked via Arche's collaboration protocol with atomic settlement.

4.5 On-chain Reputation System

On-chain reputation is the cornerstone of Arche's economy:

4.5.1 Reputation Score Calculation

$$Reputation(a) = \Sigma\,[Success\_Rate \times User\_Rating \times Volume \times Stake\_Weight]$$

Reputation scores are publicly verifiable, immutable, and reusable across Dapps.

4.5.2 Staking-Reputation Binding

An agent's reputation tier scales with its $ARCHE stake:

- 50 $ARCHE — Base tier, 100 calls/day
- 500 $ARCHE — Mid tier, 10,000 calls/day
- 5,000 $ARCHE — High tier, unlimited calls + priority routing
- 50,000 $ARCHE — Flagship tier, eligible for high-value tasks (e.g., managing vaults >$1M USDC)

Chapter 5 · Verifiable Execution Layer

5.1 Core Challenge

LLM inference is probabilistic, black-box, and nondeterministic. Blockchains require determinism, verifiability, and replayability. These two are fundamentally at odds.

How can agents execute efficiently off-chain (because LLM inference cannot run in EVM), while the chain can verify "they actually executed the declared model, with the declared inputs, by the declared logic"?

This is the central unsolved problem in Web3 × AI. Arche provides a hybrid solution:

5.2 TEE Hardware-Level Trusted Execution

5.2.1 Why Not zkML?

Zero-Knowledge Machine Learning (zkML) is theoretically the optimal solution: use ZK to prove "I ran model M on input X, producing output Y." But as of 2026:

- zkML proofs for LLMs (>7B parameters) take hours to days to generate.
- Proof sizes reach gigabytes.
- Verifier costs remain in the millions of gas.

Production-grade use is not yet viable. We must use a pragmatic solution to get the network running first.

5.2.2 TEE: Hardware-Level "Black-Box Proof"

Trusted Execution Environments (TEE) are hardware security features in modern CPUs. Representative solutions include:

- AMD SEV-SNP (Secure Encrypted Virtualization)
- Intel TDX (Trust Domain Extensions)
- ARM CCA (Confidential Compute Architecture)

TEEs provide three guarantees:

- Code integrity — hardware measures code hash at boot, immutable at runtime
- Data confidentiality — TEE memory is encrypted, invisible to OS or Hypervisor
- Remote attestation — TEE generates hardware signatures proving "I ran this exact code"

5.2.3 Phala dstack Integration

Arche partners deeply with Phala Network, using its dstack containers as the TEE execution environment:

```
Developer side:
  · Package agent code as Docker container
  · Upload to Phala dstack
  · dstack runs the container on AMD SEV-SNP nodes

At runtime:
  · Agent receives input → inference inside TEE
  · TEE hardware signs: σ = sign(privkey, hash(input) || hash(output))
  · Submits (input, output, σ) to Arche L2

On-chain verification:
  · Attestation Precompile verifies σ originates from legitimate TEE
  · If valid → transaction is finalized
```

5.3 Remote Attestation

Each agent execution generates a remote attestation report containing:

- CPU model and firmware version (proves TEE hardware is legitimate)
- Container image hash (proves the registered agent ran)
- Input hash and output hash
- Timestamp
- Hardware private-key signature

Arche's on-chain Attestation Precompile verifies an attestation in 100 gas—1000x faster than pure software verification.

5.4 Optimistic Challenge Mechanism

5.4.1 Two-Tier Trust Structure

TEE alone is insufficient—hardware may have vulnerabilities (Spectre, Meltdown), and vendors may pre-install backdoors. Arche uses an "optimistic trust + challenge" mechanism:

- TEE nodes post results on-chain (optimistically assumed honest)
- Results enter a 7-day challenge window
- Anyone can stake $ARCHE to initiate a challenge
- Challenges trigger "multi-node recomputation + zkML partial proof" flow
- If TEE node is proven malicious → 50% stake slashed ($50,000 → $25,000 burned, $25,000 compensation)
- If challenger is wrong → challenger stake is slashed

5.4.2 Interactive Bisection Protocol

When a full inference (e.g., 7B-parameter model running 10,000 steps) is challenged, the entire process cannot be zkML-proven directly. Arche introduces interactive bisection:

```
Step 1: Node N submits full state root
Step 2: Challenger C claims an intermediate step is wrong
Step 3: Protocol bisection search:
   · Split inference into halves
   · N and C identify which half they disagree on
   · Continue bisecting
Step 4: After log2(10000) ≈ 14 rounds, single-step disagreement located
Step 5: That single step is zkML-proven
Step 6: Arbitration
```

This way, zkML only needs to prove a single operation (acceptable cost), not the full inference.

5.5 Slash Economics

TEE nodes must stake $ARCHE (recommended: 50,000 $ARCHE minimum) to qualify for compute routing.

5.5.1 Dynamic Slash Coefficient

Slash amount is dynamically determined:

$$Slash(n,\ severity) = stake(n) \times (0.1 + 0.4 \times severity)$$

Where severity ∈ [0, 1]:

- severity = 0.1 — minor violation (e.g., response delay) → 14% slash
- severity = 0.5 — moderate misbehavior (e.g., output error) → 30% slash
- severity = 1.0 — severe misbehavior (e.g., key theft) → 50% slash

5.5.2 Slash Fund Flow

Slashed $ARCHE is distributed as follows:

- 70% burned (permanently removed from circulation)
- 20% rewarded to the challenger
- 10% to the DAO insurance fund

Chapter 6 · Auto Settlement Network

6.1 Design Goals

Agent-to-agent collaboration has three characteristics:

- High-frequency micropayments — thousands of calls per second, ~$0.0001 each
- Autonomous decisions — no per-transaction human signatures
- Atomic conditions — multi-agent tasks must "all-or-nothing"

Arche's Auto Settlement Network is purpose-built for these characteristics.

6.2 Smart Account Abstraction

Arche natively supports ERC-4337 Account Abstraction. Each agent has a smart contract wallet (not an EOA), featuring:

- Autonomous signing — agent signs transactions inside TEE
- Batch execution — one transaction contains multiple actions (gas savings)
- Social recovery — lost keys recoverable via trusted nodes
- Permission boundaries — single-tx caps, daily caps, contract allowlist
- Circuit breakers — anomalous behavior triggers auto-pause

6.2.1 ERC-6551 Token-Bound Account

Arche leverages the ERC-6551 standard, automatically binding each Agent NFT to a smart contract wallet:

```
AgentNFT #4823
     ↓ binds to
Token-Bound Account 0x7b3e...
     ↓ holds
· USDC, ETH, $ARCHE assets
· Other NFTs
· Subscription / Memory NFT
```

When an Agent NFT is sold, the TBA assets can optionally transfer along—enabling a new model: "AI asset bundle trading."

6.3 x402 Enhanced

Coinbase open-sourced x402 in 2025—an HTTP payment protocol designed for AI agents. Arche integrates and enhances it.

6.3.1 Standard x402 Flow

```
1. Agent A requests service from Agent B
2. Agent B returns HTTP 402 Payment Required
   with: amount, token, recipient
3. Agent A submits payment proof
4. Agent B provides service after verification
```

6.3.2 Arche's Streaming Extension

Arche adds Streaming Payments capability:

- Agent A and Agent B open a channel; A stakes 100 USDC
- A signs a "promise note" (off-chain) for each call to B
- B receives the note and provides service instantly
- A day later, B submits the latest note on-chain to settle the total
- The entire session involves only 2 on-chain transactions (open + settle)

This brings high-frequency agent collaboration costs to near-zero.

6.4 Atomic Conditional Settlement

Many agent collaboration scenarios require atomicity. Example:

> *Agent A buys a cleaned dataset (Data NFT) from Agent B. The data is encrypted with a symmetric key. A wants the decryption key; B wants USDC. Both must transfer simultaneously, or one side gets cheated.*

Arche provides a Conditional Atomic Settlement primitive:

```
atomicSwap({
  partyA: { gives: USDC(100), wants: decryption_key },
  partyB: { gives: decryption_key, wants: USDC(100) },
  conditions: [
    "key must decrypt sample data correctly",
    "USDC must be fresh (not stolen)"
  ]
})
```

The protocol verifies conditions and executes the swap atomically within a single block.

6.5 Agent Circuit Breakers

This is what differentiates Arche from other chains.

6.5.1 Three-Tier Circuit Breakers

- Agent self-bounds — owner-set per-tx and per-day caps

- Arche Guardrail Nodes — network-level anomaly detection
- Global circuit breaker — DAO-voted emergency pause

6.5.2 Prompt Injection Defense

Prompt injection is the largest agent security risk—attackers craft inputs to trick agents into transferring funds. Arche's defense strategy:

- Static permissions — agents declare contract allowlists at registration
- In-TEE defense — Guardrail module inside TEE detects prompt injection
- Cosine similarity filter — inputs with >0.85 similarity to known attack patterns rejected
- Guardrail Nodes real-time monitoring — anomalous transactions delayed
- Global pause — severe events trigger DAO multisig emergency pause

Chapter 7 · $ARCHE Token Economics

7.1 Token Fundamentals

Parameter	Value
Token Symbol	$ARCHE
Initial Total Supply	1,000,000,000 (1 Billion)
Token Type	ERC-20, native L2 Gas token
Burn Mechanism	Dual deflation (EIP-1559 + Agent Runtime Tax)
Governance	Token-weighted voting + time-lock
Launch Date	TBA

7.2 Token Core Functions

$ARCHE serves as the "universal lifeblood" of the ecosystem, spanning network security, application coordination, and trust collateralization:

7.2.1 Gas & Access (L2 Gas Fee)

Standard gas consumption for L2 block production, contract execution, and L1 state root submission.

7.2.2 Pricing & Settlement Medium (Agent Compute Currency)

Standard micropayment settlement token for agent-to-agent calls (e.g., A asks B to generate an image), cloud compute purchases, and private data access.

7.2.3 Trust Collateral (Trust Staking)

Compute providers (TEE nodes, zkVM provers) and agents executing high-value financial operations must stake $ARCHE to qualify for service access.

7.2.4 Ecosystem Governance

Voting on L2 core parameters (base gas rates, Paymaster fee splits, compute service allowlists, protocol upgrades).

7.3 Token Allocation

Category	Allocation	Vesting Schedule	Design Purpose
Compute & AI Mining Rewards	35%	10-year linear release, based on ZK proofs submitted or TEE compute-hours contributed	Fuel Agent OS inference demand
Ecosystem & Developer Treasury	25%	10% at TGE, remaining 4-year quarterly linear release	Subsidize breakout Agent developers
Core Team & Contributors	15%	12-month cliff, then 36-month monthly linear release	Long-term team value alignment
Private & Public Sale	15%	Seed/private in tranches, 20% unlocked at TGE for public sale	Bootstrap early capital
Nodes & Liquidity Bootstrapping	10%	3-year linear release	Reward L2 validators and liquidity providers

Chapter 8 · Dual-Deflation Flywheel & Mathematical Model

8.1 Flywheel Design Philosophy

Whether a public chain's token can sustain long-term appreciation depends on its value capture mechanism. Ethereum achieved a single-layer burn flywheel via EIP-1559—the more active the network, the scarcer ETH.

But Ethereum sees ~1 million transactions per day. Agent collaboration's high-frequency nature means Arche could see 1 billion transactions per day—1000x Ethereum.

A single-layer burn would dilute the flywheel effect. Arche designed a dual-deflation flywheel—Agent Runtime Tax burn stacked on top of gas burn.

8.2 Layer One: EIP-1559-style Gas Burn

Every transaction's gas fee splits into two parts:

- Base Fee — adjusted algorithmically based on network congestion
- Priority Fee — paid by users for priority inclusion

Burn rules:

- 80% of Base Fee → permanently burned
- 20% of Base Fee → validator block reward
- 100% of Priority Fee → validators

8.3 Layer Two: Agent Runtime Tax (Core Innovation)

Arche's proprietary "AI Runtime Service Tax" is the fundamental differentiator from all other public chains.

8.3.1 Tax Mechanism

In Arche's ComputePaymaster settlement, when a calling agent pays compute fees to a Provider, the protocol enforces a 2.5% AI Runtime Service Tax (Agent Runtime Tax).

This 2.5% is enforced at the EVM bytecode layer—no Dapp can bypass it.

8.3.2 Burn & Treasury Allocation

- 50% of service tax (i.e., 1.25% of transaction value) → burned directly

- 50% of service tax (i.e., 1.25% of transaction value) → ecosystem treasury (for agent developer subsidies)

8.3.3 Flywheel Effect

This means: the more AI agents collaborate, the faster $ARCHE burns.

```
Agent collaboration volume ↑
   ↓
Agent Runtime Tax burn ↑
   ↓
$ARCHE circulating supply ↓
   ↓
$ARCHE price ↑
   ↓
Early holders / Validators / Agent developers gain ↑
   ↓
Ecosystem attracts more agents
   ↓
Agent collaboration volume ↑↑
   ↓
(loop)
```

8.4 Dynamic Total Supply Equation

At any block time t, the rate of change of $ARCHE's total circulating supply S(t):

$$dS(t)/dt = R\_mint(t) - [B\_gas(t) + B\_agent(t) + S\_slash(t)]$$

Where:

- R_mint(t) — block rewards to L2 validators and TEE compute nodes (issuance rate)
- B_gas(t) — L2 base gas burned via EIP-1559-style mechanism
- B_agent(t) — portion of the 2.5% compute service tax burned by Paymaster
- S_slash(t) — slashed stake from malicious nodes / fraudulent agents

When [B_gas + B_agent + S_slash] > R_mint, $ARCHE enters net deflation. Given the high-frequency nature of agent collaboration, Arche is projected to enter permanent deflation at ~10K monthly active agents.

8.5 Dynamic Tax Rate Formula

To prevent excessive compute fees from suppressing agent collaboration, or insufficient deflation impact, the 2.5% tax rate uses a reverse-adjustment algorithm:

$$Tax(t) = Tax\_base \times (1 - \alpha \times TPS\_current / TPS\_max)$$

Where:

- Tax_base = 2.5% (base rate)
- α = smoothing coefficient (recommended initial: 0.5)
- TPS_current = current network TPS
- TPS_max = protocol-designed max TPS (e.g., 10,000)

When the network approaches saturation, the tax can smoothly drop to 1.25%, perfectly balancing throughput and value capture.

8.6 Core Treasury & Settlement Contract

Core treasury contract deployed on Arche L2 (detailed code in Appendix B):

```
// ArcheTreasury.sol
contract ArcheTreasury {
    uint256 public constant BURN_SHARE = 50;       // 50% burned
    uint256 public constant TREASURY_SHARE = 50;   // 50% treasury
    address public constant DEAD = 0x000...dEaD;

    function processAgentTax(uint256 amount) external {
        uint256 burnAmount = amount * BURN_SHARE / 100;
        uint256 treasuryAmount = amount * TREASURY_SHARE / 100;

        // Atomic execution
        IERC20(ARCHE).transfer(DEAD, burnAmount);
        IERC20(ARCHE).transfer(treasury, treasuryAmount);

        emit AgentTaxProcessed(amount, burnAmount, treasuryAmount);
    }
}
```

Chapter 9 · Three Lock-up Matrices & Anti-Dump Mechanisms

9.1 Why Lock-up Matrices?

The central problem in tokenomics: how do you make a token not only useful but also held? Only when holders lock rather than sell can price sustain.

Arche designed three lock-up matrices for the ecosystem's three core participants:

- Matrix A — Developer Lock-up (application layer)
- Matrix B — Compute Provider Restaking (infrastructure layer)
- Matrix C — User Discount Lock-up (consumption layer)

9.2 Matrix A: Developer "Code/Model Repository Collateral"

9.2.1 Mechanism

To list a high-frequency AI plugin on Arche's official Agent Store (e.g., DeepSeek prompt optimizer, cross-chain liquidity aggregator):

- Developers must stake $ARCHE proportional to expected service concurrency
- Example: per 100 calls/sec capacity → additional 1,000 $ARCHE locked
- Stake cannot be withdrawn until agent is decommissioned

9.2.2 Purpose

- Ensures developers commit to maintaining agent stability
- Successful agents lock significant tokens, reinforcing the ecosystem
- Prevents fly-by-night agents

9.3 Matrix B: Compute Provider "Verifiable Execution Restaking"

9.3.1 Mechanism

TEE validator nodes must not only stake $ARCHE at the L2 layer for block production, but also restake their LST (liquid staking token) into the Agent OS Safety Net:

```
Node N stake structure:

Layer 1: 50,000 $ARCHE → L2 Validator stake
```

```
         ↓
         receive stARCHE (liquid staking token)
         ↓
Layer 2: 50,000 stARCHE → Agent OS Safety Net stake

Rewards:
  · L2 block production rewards
  · AI compute paymaster priority routing
  · Agent OS Safety Net rewards
```

9.3.2 Purpose

- One asset achieves dual lock-up, maximizing capital efficiency
- Anchors long-term compute capital (mining pools, decentralized compute clouds) in the ecosystem
- Increases the economic cost of malicious behavior (slashed at both layers)

9.4 Matrix C: User Discount Lock-up

9.4.1 Mechanism

If agents or users lock $ARCHE in their smart wallets, they enjoy a lower tax rate when settling through the Paymaster:

Lock-up Amount	Lock Duration	Service Tax Rate
0 $ARCHE	-	2.5% (standard)
1,000 $ARCHE	30+ days	2.0%
10,000 $ARCHE	90+ days	1.5%
100,000 $ARCHE	180+ days	1.0%

9.4.2 Purpose

- Encourages heavy users to hold long-term
- Creates a spiral: hold → lock → discount → collaborate more → save more
- Reduces circulating supply, creating price support

Chapter 10 · Security Model & Risk Control

10.1 Multi-Layer Security Architecture

Arche's security model spans four layers, from hardware to economics:

- Hardware layer — TEE (AMD SEV-SNP / Intel TDX)
- Protocol layer — optimistic challenge + zkML partial proof + multisig Guardians
- Economic layer — Slash penalty mechanism + reputation capital
- Governance layer — DAO emergency pause + time-locked upgrades

10.2 Death-Spiral Safety Valve

When the ecosystem enters explosive growth and agent count grows exponentially, overly fast token burn could cause settlement medium scarcity.

Arche's L2 economic model includes the following auto-balancing rules:

10.2.1 Soft-cap Trigger

When historical $ARCHE total burn reaches 50% of initial supply, the BURN_SHARE in ArcheTreasury auto-reduces from 50% to 10%. The remaining 40% of service tax is no longer burned but auto-converted to "Agent Compute Credit Rebate"—directly subsidized to the top 10% most active agents.

10.2.2 Gas Decoupling Mechanism

Since $ARCHE price could surge during speculation, fixing gas and compute fees to absolute $ARCHE quantities would make AI calls prohibitively expensive.

Solution: Agent OS internally prices in "ArcheGas" virtual points.

- ArcheGas pegged to off-chain fiat or compute equivalents
- Example: 1 million DeepSeek tokens inferred = $0.10 worth of ArcheGas (fixed)
- At settlement, Paymaster auto-converts via Oracle pricing to the equivalent $ARCHE

This way, no matter how high $ARCHE rises, the "real cost" of agent invocations remains stable, preventing ecosystem paralysis from token appreciation.

10.3 Prompt Injection Defense

The largest on-chain security threat in the AI era is prompt injection—attackers craft inputs to induce agents to transfer funds.

Arche's defense-in-depth strategy:

10.3.1 Static Permission Isolation

Agents declare permission boundaries at registration (allowed contract list, per-tx and per-day caps). Any transaction exceeding these is rejected at the protocol layer.

10.3.2 In-TEE Guardrail Module

Inside the agent's TEE container, a Guardrail module (based on lightweight NLI model) is pre-installed:

- Scans every input
- Computes cosine similarity against known attack patterns
- Similarity > 0.85 → directly rejects execution

10.3.3 Guardrail Nodes (On-chain Watchers)

A dedicated set of Guardrail nodes monitors network-wide agent transactions:

- Anomalous transactions are delayed (5-minute cooling period)
- Auto-pause triggered when DAO-set thresholds are met
- Guardrail nodes stake $ARCHE; false reports are slashed

10.3.4 Global Circuit Breaker

In the event of a network-wide security incident, DAO multisig can trigger "global fuse":

- Entire Arche network pauses for 24-72 hours
- Users can still withdraw, but all agent autonomous transactions are frozen
- Provides time for security team to investigate and remediate

10.4 Compliance & Anti-Money-Laundering Isolation

10.4.1 Risk

High-frequency anonymous agent washing trades easily trigger compliance risks. Regulators (FATF, SEC, EU MiCA) are increasingly focused on AI agent financial activities.

10.4.2 Solution: Reputation Capital Layer

- Agents must stake $ARCHE as security bond
- If behavior is verified as malicious or violating → stake slashed
- KYA registration records agent model source and owner identity (optional KYC)
- Compliant agents earn "white-label" status, gain institutional access
- Integration with Chainalysis / TRM Labs compliance tools

Chapter 11 · Governance & DAO

11.1 Three-Tier Governance Structure

Arche's governance is divided into three tiers:

- Protocol Layer — decides L2 core parameters (gas, tax, security rules)
- Ecosystem Layer — decides ecosystem treasury fund allocation
- Sub-Market Layer — decides specific agent category rules (DeFi agent submarket, content agent submarket, etc.)

11.2 Voting Mechanism

Arche uses "$ARCHE token-weighted voting + time-lock":

- 1 $ARCHE = 1 vote
- Proposals require at least 100,000 $ARCHE stake to submit
- 7-day voting period
- Passing threshold: ≥4% of circulating supply participation, with majority in favor
- 48-hour time-lock after passing
- DAO multisig may veto during time-lock (emergency only)

11.3 Delegate Voting

To prevent "hold but not vote" governance apathy, Arche introduces delegation:

- Holders can delegate voting power to trusted Delegates (e.g., known researchers, ecosystem leaders)
- Delegates must publicly disclose voting history and rationale
- Delegators can revoke at any time

11.4 Emergency Multisig

Arche has a 9-member emergency multisig committee (5-of-9 threshold), composed of:

- 3 core team representatives
- 3 independent technical experts (external security auditors)
- 3 community representatives (DAO-elected)

The emergency multisig's sole power is "global pause"—triggered only on critical security vulnerabilities. All other actions must follow the full DAO governance process.

Chapter 12 · Product Positioning & Roadmap

12.1 Three User Groups

Arche serves three groups simultaneously. They enter at different phases, forming a complete flywheel.

12.1.1 End Users (Consumer)

Core promise: Tell agents what you want in one sentence. The chain handles the rest.

- Open archeai.network → browse Agent marketplace
- Select reputable agent → set budget and permission limits
- Agent auto-executes tasks (DeFi yield, data analysis, content creation, etc.)
- Entire process is on-chain verifiable, auditable, and accountable

12.1.2 AI Developers (Business)

Core promise: Turn your model into a sustainable on-chain asset.

- Register agent via Arche SDK → mint Agent NFT
- Set service pricing (per call, per time, revenue share)
- User invokes → auto settlement → revenue auto-deposited
- Agent NFTs tradable on secondary markets, forming a new asset class

12.1.3 Enterprise Users (Business)

Core promise: Plug AI into business workflows—compliant, secure, auditable.

- Procure mature enterprise-grade agents (support, analytics, contract audit) from Agent Store
- Pay-as-you-go via streaming token payments
- Agents run in TEE → business data never leaks
- All transactions auditable on-chain → meets finance, healthcare, legal compliance

12.2 Overall Roadmap

Phase	Timeline	Core Deliverables
Phase 1 Infrastructure	2026 Q3-Q4	Testnet launch; 5 core contracts (AgentRegistry + ERC-6551 + Token + ServicePayment + RevenueShare); base SDK; 1-2 demo agents

Phase 2 OS SDK	2027 Q1-Q2	Phala dstack integration (TEE); EIP-8004 implementation; on-chain MCP memory; x402 streaming payments; Agent Store prototype
Phase 3 Agentic Market	2027 Q3-Q4	Intent routing engine; Bittensor-style subnet incentives; Guardrail nodes; EigenLayer restaking integration; cross-chain bridges (CCIP / LayerZero)
Phase 4 Mainnet Launch	2028 Q1-Q2	Mainnet launch; TGE; CEX listings; TradFi partnerships; Agent NFT secondary market; full DAO governance

12.3 Phase 1 Detailed Milestones (Next 6 Months)

Month	Deliverables
M1	Rollup deployment; GitHub Org setup; technical docs site live; v1 core contracts (AgentRegistry + Token)
M2	ERC-6551 integration; ServicePayment + UsageRecord contracts; testnet faucet; block explorer
M3	RevenueShare contract; base TypeScript SDK; developer docs; 1 demo agent (Phala TEE)
M4	Security audit begins; bug bounty launched; community testnet open; KOL partnerships
M5	Audit findings remediated; performance optimization; 2nd-3rd demo agents added; seed round material prep
M6	Public testnet launch; first ecosystem partners announced; seed round opens

Chapter 13 · Team, Partners & Ecosystem

13.1 Core Team

Arche's core team is led by two senior researchers from Canada's top academic institutions. Their research expertise comprehensively covers the two most critical technical domains across Arche's three-layer architecture: smart contract formal verification / DeFi compliance, and trusted execution environments / zero-knowledge proofs / data privacy.

Prof. Andreas Veneris

University of Toronto · Department of Electrical & Computer Engineering (ECE)

Research focus:

- Smart contract formal verification
- Decentralized finance compliance frameworks
- AI-powered blockchain security analysis

Role at Arche:

- Leads formal verification of Arche core contracts (AgentRegistry, AgentTax Precompile, ArcheTreasury)
- Architects KYA framework and DeFi compliance integration with global regulatory frameworks
- Leads Guardrail Nodes anomaly detection and security strategy architecture

Prof. Florian Kerschbaum

University of Waterloo · David R. Cheriton School of Computer Science

Research focus:

- Data privacy and security
- Trusted Execution Environment (TEE) security models
- Zero-Knowledge Proof (ZKP) engineering applications
- Federated learning × blockchain

Role at Arche:

- Leads Arche Layer 2 Verifiable Execution architecture (TEE + optimistic challenge + zkML hybrid)
- Leads engineering of Phala dstack integration, Attestation protocol, and interactive bisection
- Designs privacy-preserving agent data processing and federated learning collaboration mechanisms

13.2 Investors

Arche's seed and strategic financing rounds are in progress. The full list of investment institutions will be officially disclosed via official channels (archeai.network and Twitter @ArcheChain) closer to TGE.

13.3 Technical Partners

Arche has established technical partnerships with the following infrastructure providers:

- Phala Network — TEE trusted compute (dstack)
- EigenLayer — Restaking and EigenDA
- Wormhole / LayerZero — Cross-chain messaging
- Chainlink — Oracles and CCIP
- Pyth — High-frequency price oracles

13.4 Ecosystem Partners

Arche's application ecosystem is expanding rapidly, with focus on:

- Web3-native AI agent development teams
- DeFi protocols (lending, DEX, yield aggregators)
- NFT marketplaces (OpenSea / Blur integration)
- Wallet providers (MetaMask / Rabby / OKX Web3 Wallet)
- Layer 2 bridges (Stargate / Hop)
- AI model providers (Anthropic / OpenAI-compatible interfaces)

Chapter 14 · Legal, Compliance & Disclaimers

14.1 Project Entity

Arche is operated by the Arche Foundation (registered in an offshore jurisdiction; specific registration entity to be officially disclosed at TGE) as the non-profit foundation entity. The Foundation's mission is to advance decentralized governance, ecosystem development, and long-term token economic sustainability of the Arche protocol.

The Arche Foundation does not hold ultimate control over the Arche protocol—all major decisions including protocol upgrades, parameter adjustments, and treasury usage are made by $ARCHE holders through DAO governance.

14.2 Token Nature Statement

$ARCHE is a utility token of the Arche network, used for:

- Paying network gas fees
- Paying agent service fees
- Staking and governance participation

$ARCHE does not represent any form of company equity, debt, or profit distribution rights. $ARCHE holders are not entitled to any form of dividend or distribution.

14.3 Jurisdictional Restrictions

$ARCHE is not sold to residents or entities in the following jurisdictions:

- United States (except via Reg D / Reg S exemptions)
- People's Republic of China (including Hong Kong and Macau, except specific exemptions)
- FATF grey-list / black-list jurisdictions
- OFAC sanctioned parties

14.4 Forward-Looking Statements

This whitepaper contains predictions, plans, and visions for the future. All forward-looking statements involve risks and uncertainties. Actual results may differ materially from predictions due to technical, regulatory, market, and other factors.

Before making any investment decision, investors should:

- Conduct their own thorough research (DYOR)

- Consult independent legal, tax, and investment advisors
- Only use funds they can afford to lose
- Understand the high volatility and potential total loss risk of crypto assets

14.5 Disclaimers

All features, characteristics, and timelines described in this whitepaper are forward-looking statements and do not constitute any form of commitment or guarantee. The Arche team, contributors, and affiliated parties bear no liability for:

- Token price volatility
- Asset losses due to technical vulnerabilities
- Project adjustments or termination due to regulatory changes
- Third-party defaults (e.g., exchanges, custodians)
- User operational errors

Appendix A · Technical Glossary

Agent OS — Arche Layer 1, responsible for agent identity, memory, reputation, and orchestration

Agent NFT — NFT representation of agent identity (ERC-721), tradable on secondary markets

AgentRegistry — On-chain agent registry contract

Agent Runtime Tax — Arche's proprietary 2.5% AI collaboration tax, core of the dual-deflation flywheel

Attestation — TEE hardware remote attestation proving code ran inside a legitimate TEE

Circuit Breaker — Mechanism that auto-pauses anomalous transactions

dstack — Phala Network's TEE container technology

EIP-1559 — Ethereum's gas burn mechanism, Arche Layer 1 burn

EIP-8004 — AI Agent Identity Standard, draft 2026

EigenLayer — Restaking protocol, Arche compute provider second-layer stake

ERC-4337 — Ethereum Account Abstraction standard

ERC-6551 — Token-Bound Account standard, foundation of agent wallets

Guardrail Nodes — On-chain watcher nodes monitoring anomalous transactions

Intent Routing — Natural language → task tree

KYA — Know Your Agent, agent identity compliance framework

MCP — Model Context Protocol, Anthropic's agent context standard

Restaking — Reusing the same stake across multiple security networks

Slash — Stake penalty mechanism

Streaming Payments — Streaming payment channels for high-frequency micropayments

TEE — Trusted Execution Environment, hardware-level secure execution

TBA — Token-Bound Account, smart contract wallet bound to an NFT

W3C DID — Decentralized Identifier

x402 — Coinbase's HTTP payment standard for AI agents

zkML — Zero-Knowledge Machine Learning, ZK proofs for ML inference

Appendix B · Core Smart Contract Examples

B.1 AgentRegistry.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

contract AgentRegistry {
    struct Agent {
        address owner;
        bytes32 modelHash;
        bytes32 weightsHash;
        bytes32 systemPromptHash;
        string  mcpEndpoint;
        address tba;
        uint256 stakedAmount;
        uint256 reputation;
        AgentStatus status;
        uint256 registeredAt;
    }

    enum AgentStatus { Active, Paused, Slashed, Retired }

    mapping(uint256 => Agent) public agents;
    mapping(address => uint256[]) public ownerAgents;
    uint256 public nextAgentId;

    uint256 public constant MIN_STAKE = 50 ether; // 50 $ARCHE

    event AgentRegistered(uint256 indexed agentId, address indexed owner);
    event AgentStaked(uint256 indexed agentId, uint256 amount);
    event AgentSlashed(uint256 indexed agentId, uint256 amount);

    function registerAgent(
        bytes32 modelHash,
        bytes32 weightsHash,
        bytes32 systemPromptHash,
        string calldata mcpEndpoint
    ) external returns (uint256 agentId) {
        require(IERC20(ARCHE).transferFrom(msg.sender, address(this),
MIN_STAKE),
                "Stake required");

        agentId = nextAgentId++;
        agents[agentId] = Agent({
```

```
            owner: msg.sender,
            modelHash: modelHash,
            weightsHash: weightsHash,
            systemPromptHash: systemPromptHash,
            mcpEndpoint: mcpEndpoint,
            tba: _createTBA(agentId),
            stakedAmount: MIN_STAKE,
            reputation: 0,
            status: AgentStatus.Active,
            registeredAt: block.timestamp
        });

        ownerAgents[msg.sender].push(agentId);
        emit AgentRegistered(agentId, msg.sender);
    }
}
```

B.2 AgentTaxPrecompile (Pseudocode)

```
// AI Runtime Tax enforced at EVM bytecode layer

contract AgentTaxPrecompile {
    uint256 public constant TAX_BASE = 250;  // 2.5% in basis points
    uint256 public constant BURN_SHARE = 5000; // 50%
    address public constant DEAD = address(0xdead);
    address public treasury;

    function processAgentPayment(
        address payer,
        address payee,
        uint256 amount,
        uint256 lockedStake
    ) external returns (uint256 netPayment) {
        uint256 currentTax = _calculateDynamicTax(payer, lockedStake);
        uint256 taxAmount = amount * currentTax / 10000;
        uint256 burnAmount = taxAmount * BURN_SHARE / 10000;
        uint256 treasuryAmount = taxAmount - burnAmount;

        IERC20(ARCHE).burn(burnAmount);
        IERC20(ARCHE).transfer(treasury, treasuryAmount);

        netPayment = amount - taxAmount;
        IERC20(ARCHE).transfer(payee, netPayment);
    }

    function _calculateDynamicTax(address payer, uint256 lockedStake)
        internal view returns (uint256) {
```

```
        if (lockedStake >= 100000 ether) return 100;  // 1%
        if (lockedStake >= 10000 ether)  return 150;  // 1.5%
        if (lockedStake >= 1000 ether)   return 200;  // 2%
        return TAX_BASE;  // 2.5%
    }
}
```

Appendix C · Mathematical Derivations

C.1 Dynamic Total Supply Equation

$$dS(t)/dt = R\_mint(t) - [B\_gas(t) + B\_agent(t) + S\_slash(t)]$$

Term definitions:

C.1.1 Issuance R_mint(t)

$$R\_mint(t) = R\_validator(t) + R\_compute(t)$$

R_validator(t) — Validator block reward

R_compute(t) — TEE compute node reward

C.1.2 Gas Burn B_gas(t)

$$B\_gas(t) = 0.8 \times BaseFee(t) \times TxCount(t)$$

C.1.3 Agent Tax Burn B_agent(t)

$$B\_agent(t) = 0.5 \times Tax(t) \times AgentVolume(t)$$

Where Tax(t) is given by the dynamic adjustment formula:

$$Tax(t) = 0.025 \times (1 - 0.5 \times TPS(t) / TPS\_max)$$

C.1.4 Slash Burn S_slash(t)

$$S\_slash(t) = 0.7 \times \Sigma\, Stake(n) \times severity(n)$$

C.2 Net Deflation Critical Point

$ARCHE enters net deflation when:

$$B\_gas(t) + B\_agent(t) + S\_slash(t) > R\_mint(t)$$

Simplified assumptions (steady-state operation):

- R_mint(t) ≈ 3.65M $ARCHE / year (35% × 100M / 10 years)
- B_gas(t) ≈ TxCount × 0.001 $ARCHE × 0.8 = 0.0008 × TxCount
- B_agent(t) ≈ AgentVolume × 0.0125

For simplified derivation, assume:

- 1M agent transactions per day
- Average 1 $ARCHE value per agent transaction
- Daily gas fees ≈ 1,000 $ARCHE

Then:

- B_gas ≈ 365,000 $ARCHE / year
- B_agent ≈ 1M × 365 × 0.0125 = 4,562,500 $ARCHE / year
- R_mint ≈ 3,650,000 $ARCHE / year
- Net burn ≈ 1,277,500 $ARCHE / year (0.13%/year of total supply)

The above are rough estimates. Real data will be continuously observed and adjusted during the Testnet phase.

— End of Document —

Arche Official Whitepaper v1.0

June 2026

https://archeai.network