AzraCode

Whitepaper

Privacy by Architecture, Not by Promise.

A peer-to-peer sealed AI building protocol.
Settlement on Solana with $AZRA, under a Burn-to-Build economy.

Version 1.0 - Draft
June 2026

Table of Contents

1. Abstract

AzraCode is a peer-to-peer protocol for building software with AI without surrendering the data to do it. Computation executes inside sealed execution environments - hardware-enforced enclaves where input is decrypted, processed, and re-encrypted without the host being able to read, persist, or train on it.

Settlement runs on Solana with the native token $AZRA under a Burn-to-Build model: each build burns $AZRA, and each verified compute mints $AZRA to the provider that ran it. Privacy is a property of the architecture, not a clause in a policy.

2. The Problem - Cloud AI is a Surveillance Surface

Every mainstream AI tool asks you to ship your most sensitive material - source code, contracts, internal docs, private data - to a server you do not control. Once it arrives, it can be logged, retained, inspected by operators, and used to train future models. The privacy guarantee is a promise, enforced by policy and auditing rather than by structure.

For builders, that is an unacceptable trade. The work that benefits most from AI is exactly the work that is too sensitive to expose. Policy can change, logs can leak, and "we don’t train on your data" is unverifiable from the outside. The surveillance surface exists the moment plaintext leaves your machine.

3. The Solution - Privacy by Architecture

AzraCode removes the surveillance surface instead of promising not to use it. Data is encrypted on your device with a wallet-bound key and only ever decrypted inside a hardware-enforced enclave. The operator hosts the cage but holds no key to it - surveillance is structurally impossible, not merely prohibited.

The result is a builder, not a chatbot: AzraCode returns finished, deployable artifacts - code, audits, diagrams, tests - optimized for work that actually ships. No logs, no retention, no training surface.

4. Architecture - Sealed Execution Lifecycle

A single build moves through five stages. Encrypt on the device; route the ciphertext to an attested sealed node; execute inside the enclave where plaintext briefly exists; return the artifact re-encrypted to the wallet key; then destroy the environment - memory zeroed, state wiped, $AZRA burned at settlement.

Trust is established by proof-of-compute. Each enclave emits a signed attestation of its exact code and configuration. Verifiers check that quote against the expected measurement and the hardware root of trust before any reward settles - proving the work happened without revealing what the work was.

ENCRYPT -> ROUTE -> SEALED EXECUTE -> RETURN -> DESTROY

5. Token Model - $AZRA and Burn-to-Build

$AZRA is an SPL token on Solana with a fixed total supply of 1,000,000,000. It is the unit of settlement for sealed compute. The economy is Burn-to-Build: demand, not an emission calendar, drives supply.

Each build burns $AZRA. Each verified compute mints $AZRA to the provider that ran it. Heavier real usage burns more on the build side while minting proportionally to honest providers - so the token tracks actual network work.

Supply distribution

Compute Rewards	**45%**
Community & Ecosystem	**25%**
Treasury	**15%**
Core Contributors	**10%**
Liquidity	**5%**

Total supply: 1,000,000,000 $AZRA (SPL, Solana).

Contract address: TBA at launch.

6. Roadmap

Phase 0 - Sealed Genesis [done]

Sealed runtime, attestation spec, testnet settlement.

Phase 1 - Public Beta [active]

Open network, CLI + Studio + Web App, $AZRA on Solana.

Phase 2 - Provider Network [next]

Permissionless sealed nodes, reward minting at scale.

Phase 3 - Composable Seals [next]

Multi-node sealed pipelines, on-chain proof registry.