ArxCode: A Peer-to-Peer Sealed AI Building Protocol

Solana Labs Research Collective

contact@arxcode.xyz · arxcode.xyz

Abstract

A purely private form of AI computation would allow builders to generate software, documents, and analysis without any intermediary observing, storing, or training on their inputs. Existing AI tools offer privacy as policy—a contractual promise enforced by the very party that profits from breaking it. We propose a protocol where privacy is enforced by architecture rather than by promise. Work submitted to the network runs inside a sealed execution environment; inputs enter and finished output returns, but nothing in between is observable by the operator, the network, or any third party. Payment is settled per unit of compute using $ARX, an SPL token on Solana, eliminating accounts, subscriptions, and the data-for-service exchange that underlies conventional AI products. Tokens are removed from circulation as work is performed and minted only when verified compute is delivered, binding supply directly to real usage. As long as honest node operators control a majority of compute capacity, the network produces correct output without exposing the data that produced it.

1. Introduction

Software development, technical writing, and analysis have come to rely on AI tools hosted by centralized providers. These services are convenient, but their economics depend on a quiet exchange: the user supplies prompts, code, and proprietary data, and the provider supplies computation in return for the right to read, store, and train on everything submitted. What is presented as a free or low-cost tool is in practice a data-collection apparatus. The user is not the customer; the user is the input.

For casual use this trade is acceptable. For anyone building something of value—a startup, a trading strategy, a protocol, an unreleased product—it is disqualifying. A privacy policy does not solve the problem, because a policy is a promise made by the party with the strongest incentive to break it and the exclusive ability to do so unobserved. There is no way for the user to verify that deleted data was deleted, that unlogged prompts were unlogged, or that a model was not trained on their work.

What is needed is an AI building system based not on the operator's good behavior but on an architecture in which surveillance is impossible rather than merely prohibited. In such a system the operator *cannot* read user data because the system exposes none; *cannot* leak logs because none are kept; and *cannot* train on submissions because they were never retained. We propose such a system. It combines sealed execution with on-chain settlement, replacing the trust placed in a provider with the guarantees of a protocol.

2. What Makes ArxCode Different

ArxCode is not an incrementally more private chatbot. Three properties, taken together, distinguish it from every conventional AI tool and from prior privacy-oriented alternatives.

It is a builder, not an assistant. The unit of value is a finished artifact—a working feature, a deployable interface, a complete test suite, a security audit—rather than a fragment of advice. The user describes an outcome; the protocol returns the outcome. Most AI products optimize for conversation; ArxCode optimizes for shipped work.

Privacy is enforced by the architecture, not the operator. Competing tools can read everything submitted and rely on a policy not to misuse it. ArxCode cannot read what it processes. The distinction is

structural and is developed in Sections 3 and 4: there is no administrative session to inspect, no log to leak, and no retained corpus to train on.

The token is the meter, not a coupon. $ARX is not a governance token with speculative future utility bolted onto a free product. It is the metering unit consumed by every build. Usage burns supply; verified compute mints it. The economy is driven by work performed, not by emissions or promises.

No prior system combines all three. Conventional assistants are readable and account-bound. Privacy chains protect transactions but not computation. Confidential-compute offerings seal execution but bill through ordinary accounts that reintroduce a surveillance surface. ArxCode closes the loop: sealed building, paid anonymously, settled on-chain.

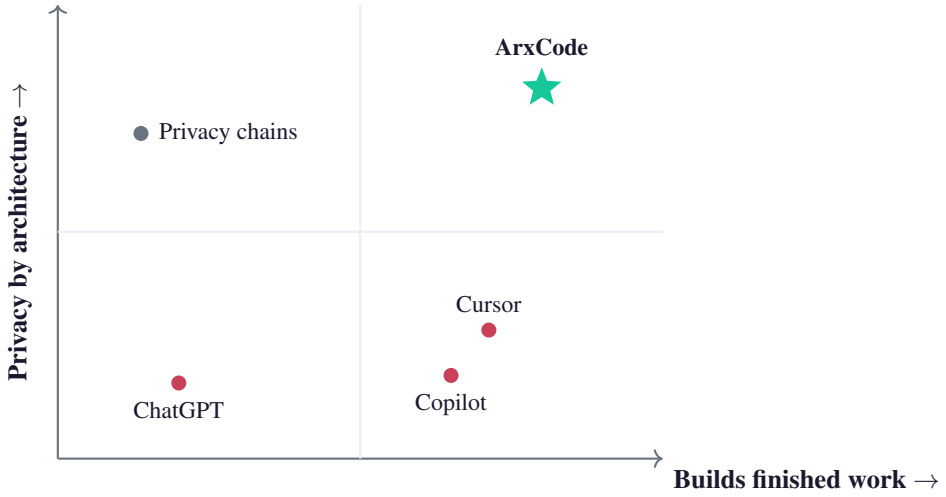

Figure 1: ArxCode occupies the quadrant no incumbent reaches—finishing real work while remaining private by construction.

3. Privacy as Policy versus Privacy as Protocol

The conventional model treats privacy as a contractual layer placed on top of an architecture that is fully capable of surveillance. The provider retains an administrative interface that can read any session, a logging system that records inputs for debugging, and a training pipeline that ingests submissions to improve future models. Privacy, in this model, is the provider's decision to refrain from using capabilities it possesses.

This arrangement fails in three independent ways. It fails to *subpoena*, where a third party compels disclosure of data the provider was able to retain. It fails to *breach*, where an attacker extracts logs the provider chose to keep. And it fails to *drift*, where a future change in terms, ownership, or incentives quietly repurposes data the user assumed was private.

Privacy as protocol removes the capability rather than restraining its use. If the architecture exposes no readable session, there is nothing to subpoena. If no logs are written, there is nothing to breach. If submissions are never retained, no future policy can repurpose them. The distinction is the difference between a promise and a property. We build on the latter.

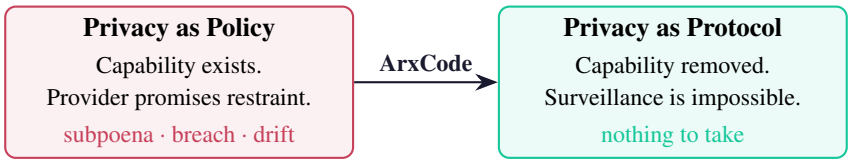

Figure 2: The shift from a promise the operator can break to a property the operator cannot violate.

4. Sealed Execution

The core of the protocol is the sealed execution environment. A unit of work—which we call a *build*—consists of an encrypted input submitted by a user and a finished output returned to that user. Between submission and return, the work is processed inside an environment that is opaque to everyone, including the node operator running it.

4.1 Definition

A sealed execution environment guarantees the following properties for any build b with input x and output y:

(i) **Input confidentiality.** x is decrypted only inside the environment and is never accessible to the operator, the host, or the network.

(ii) **Non-retention.** Neither x nor any intermediate state survives the build. When the environment terminates, the data is gone.

(iii) **Output integrity.** y is returned only to the holder of the originating wallet key, and its correctness can be attested without revealing x.

(iv) **No training surface.** Because x is never persisted, it cannot enter any training corpus.

4.2 Lifecycle of a Build

A build proceeds in five stages. The user encrypts the input client-side under a key bound to their wallet. The encrypted payload is routed to a node whose environment can be attested as sealed. The environment decrypts the input internally, performs the computation, and produces the output. The output is returned to the user, encrypted to their key. Finally the environment is destroyed, taking all intermediate state with it, and the consumed $ARX is burned on settlement. At no point does plaintext exist outside the sealed boundary, and at no point is a copy retained.

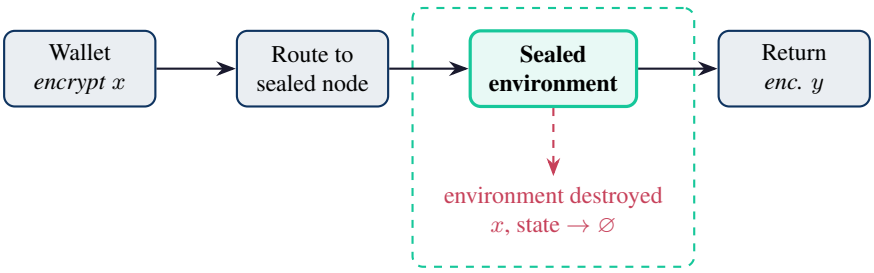

Figure 3: A build's lifecycle. Plaintext exists only inside the opaque boundary and is destroyed on completion.

5. Threat Model

We state explicitly what the protocol defends against and what it assumes. The adversary may be the node operator, the host infrastructure, a network observer, a legal authority compelling disclosure, or a

future owner of the protocol with changed incentives. The adversary's goal is to read, copy, or train on a user's input.

Against the operator and host. The plaintext input is decrypted only inside the sealed boundary, which the operator cannot inspect without invalidating its attestation. An operator that breaks the seal to read data forfeits the ability to produce valid attestations and earns no reward (Section 8).

Against the network observer. Inputs and outputs cross the network encrypted to wallet-bound keys. An observer sees ciphertext and timing, not content.

Against subpoena and breach. The protocol retains nothing after a build completes. There is no store to compel and no log to exfiltrate; a request for past data returns nothing because nothing was kept.

Against drift. Because the architecture forecloses retention, a future change of policy cannot reach data that was never collected.

We do *not* claim to defend against a user who voluntarily discloses their own output, against endpoint compromise on the user's own device, or against an adversary controlling a majority of attested compute capacity—the last being the boundary condition analyzed in Section 8.

6. Payment Without Accounts

A sealed system cannot rely on conventional accounts, because accounts are themselves a surveillance surface: an email, a billing record, and a usage history tied to an identity. The protocol replaces the account with a wallet and the subscription with a balance.

A user interacts with the network through a Solana wallet address and a balance of $ARX. There is no registration, no email, no know-your-customer step, and no payment card on file. To use the network, a user holds $ARX; to perform a build, the network consumes $ARX proportional to the compute expended. This is the entire transaction. No behavioral data is collected because none is required to bill, and none can be sold because none exists.

Settlement occurs on Solana. The chain's sub-second finality and sub-cent fees make per-build settlement economical at the granularity of individual operations, which a slower or more expensive chain could not support.

7. The Token and the Incentive

$ARX is an SPL token that functions as the metering unit of the network. Its design ties token supply to real compute rather than to speculation or emissions.

7.1 Burn-to-build

Every build permanently removes a quantity of $ARX from circulation proportional to the compute consumed. Usage is therefore deflationary by construction: the act of using the network for its intended purpose reduces supply. Demand for builds translates directly into demand for the token, and sustained usage translates into sustained contraction of supply.

7.2 Mint-on-compute

New $ARX enters circulation through exactly one channel: a node operator delivers verified compute and is rewarded for it. There is no liquidity mining, no farming airdrop, and no emission to bootstrap

mercenary capital. Tokens are created only when real work is performed, so circulating supply tracks the productive capacity of the network rather than a predetermined inflation schedule.

7.3 Hold-for-discount

The per-build rate a user pays declines as the user's $ARX balance increases. The same compute costs less to a larger holder. This aligns the economic incentive toward accumulation rather than disposal, and rewards committed users with a lower effective price for identical service.

7.4 Supply dynamics

Let $B(t)$ denote the cumulative $ARX burned through builds up to time t, and $M(t)$ the cumulative $ARX minted for verified compute over the same interval. The circulating supply $S(t)$ evolves as

$$S(t) = S_0 + M(t) - B(t),$$

where S_0 is the genesis supply. Because mint events require delivered compute and burn events accompany consumed compute, both terms are driven by the same underlying activity. As adoption grows, aggregate burns from many users tend to outpace mint events tied to marginal capacity additions, producing net contraction. The result is a supply curve governed by usage rather than by schedule.

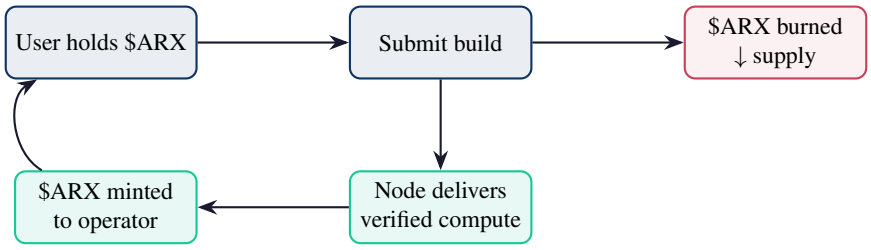

Figure 4: Tokens burn as work is consumed and mint only as verified compute is delivered—supply tracks usage.

8. The Node Network

The network is sustained by independent node operators who contribute sealed compute capacity. An operator stakes resources, attests that its environment satisfies the sealing properties of Section 4, and accepts builds routed to it. In return, the operator earns newly minted $ARX for compute it verifiably delivers.

Correctness is enforced by attestation and redundancy rather than by inspection. An operator cannot earn rewards by claiming compute it did not perform, because rewards are conditioned on a verifiable attestation that the work was carried out inside a properly sealed environment. An operator that tampers with its environment to expose user data forfeits its ability to produce valid attestations and is excluded from reward. So long as honest operators control a majority of attested capacity, the network delivers correct, private output.

8.1 Reorganizing trust

Consider the trust an attacker must overcome to read a user's data under each model. Under the policy model, an attacker need compromise a single point: the provider's administrative access, its logs, or its

training store. Under the protocol model, the data is never assembled in a place an attacker can reach; to expose a build, an attacker must defeat the sealing of the specific environment in which it ran, during the brief interval it existed, without producing a detectable attestation failure.

We frame this as a race analogous to distributed consensus. Let p be the share of attested capacity controlled by honest operators and $q = 1 - p$ the share controlled by an adversary attempting to subvert sealing. The probability that the adversary compromises a given build falls as honest capacity grows; for $p > q$ the expected fraction of compromised builds approaches zero as the network scales. Privacy, like consensus, becomes more robust as honest participation increases.

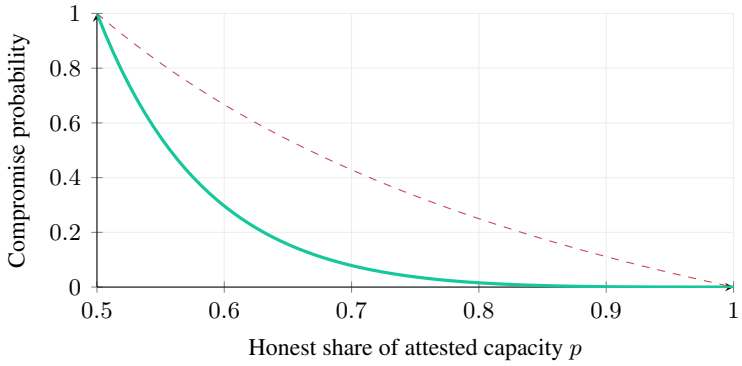

Figure 5: As honest capacity p grows past a majority, the probability of compromising a build collapses toward zero (solid: higher redundancy; dashed: lower).

9. Applications

The protocol is a general engine for private knowledge work. A single balance of $ARX purchases any of the following, with no input ever leaving the sealed boundary:

- **Code.** Writing, repairing, and extending features across a full stack rather than isolated fragments.
- **Applications and interfaces.** Working front-end and back-end software with live preview.
- **Diagrams.** Flowcharts, system architecture, data models, and charts.
- **Documentation and writing.** Guides, references, and technical explanation.
- **Review and security.** Detecting defects and vulnerabilities and proposing corrections.
- **Data.** Constructing and explaining database queries.
- **Tests.** Generating suites that validate the behavior of delivered code.
- **Private assistance.** Research and reasoning conducted entirely within the sealed boundary.

The addressable population is not limited to those already interested in cryptocurrency. It is everyone whose work touches sensitive material—engineers protecting unreleased products, crypto teams protecting on-chain logic, and professionals in law, medicine, finance, and research who currently restrict their use of AI because they assume, correctly, that conventional tools are reading. The protocol removes the reason for that restraint.

10. Distribution

Because the protocol exposes no user data, it can serve as a private execution layer beneath existing infrastructure providers. A cloud platform with a large enterprise base can offer sealed AI building to customers whose legal constraints forbid sending proprietary data to a conventional provider. The customer obtains AI capability without surrendering confidentiality; the platform gains a settlement layer for which it can earn a share; and the network gains distribution. Every build performed through such a channel consumes $ARX, binding third-party distribution to the same usage-driven token economy described above.

11. Comparison

	ArxCode	**Hosted AI**	**Confidential compute**
Can read your data	No (sealed)	Yes	Partial
Trains on your work	No	Yes	Varies
Account / email	Wallet only	Required	Required
Payment	Pay-per-build	Subscription	Subscription
Finishes real work	Yes	Partial	No
Settlement	On-chain	Off-chain	Off-chain

Table 1: ArxCode versus the closest categories of alternative.

12. Conclusion

We have proposed a protocol for AI building in which privacy is a property of the architecture rather than a clause in a contract. By processing every build inside a sealed execution environment, the protocol removes the operator's ability to read, retain, or train on user data, converting privacy from a promise into a guarantee. By settling payment in $ARX on Solana, it eliminates accounts and subscriptions and the data-for-service exchange they enable. By burning tokens as work is consumed and minting them only as verified compute is delivered, it ties supply to genuine usage rather than to emission schedules or speculation. The network requires no trust in any single operator: as long as honest participants control a majority of attested capacity, it produces correct output while exposing nothing of the data that produced it. This is the difference between privacy as policy and privacy as protocol—a pinky-swear replaced by mathematics.

ArxCode · Chain: Solana (SPL) · Token: $ARX

AI Infrastructure / Privacy / DePIN · `arxcode.xyz`