ArxCode: A Private AI Code Studio on Solana

ArxCode Research

contact@arxcode.xyz · arxcode.xyz

Abstract

Every AI tool you use today reads what you type. Your code, your ideas, your unreleased product—it all goes into a system designed to store it, learn from it, and sell access to the intelligence it produces. ArxCode is built differently: what you submit is processed inside a sealed environment and returned to you as finished work, with nothing observed, stored, or retained in between. Privacy is not a policy we promise. It is an architectural property we cannot violate even if we wanted to. Payment is made per build using $ARX, an SPL token on Solana—no account, no email, no subscription. The token burns as work is consumed and is earned only when compute is delivered, tying its supply directly to real usage.

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.

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 code studio based not on the operator's good behavior but on an architecture in which surveillance is impossible rather than merely prohibited. We propose such a system.

2. What Makes ArxCode Different

ArxCode is not an incrementally more private chatbot. Three properties, taken together, distinguish it from every conventional AI tool.

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. There is no admin 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. It is the metering unit consumed by every build. Usage burns supply; completed builds create it. The economy is driven by work performed, not by emissions or promises.

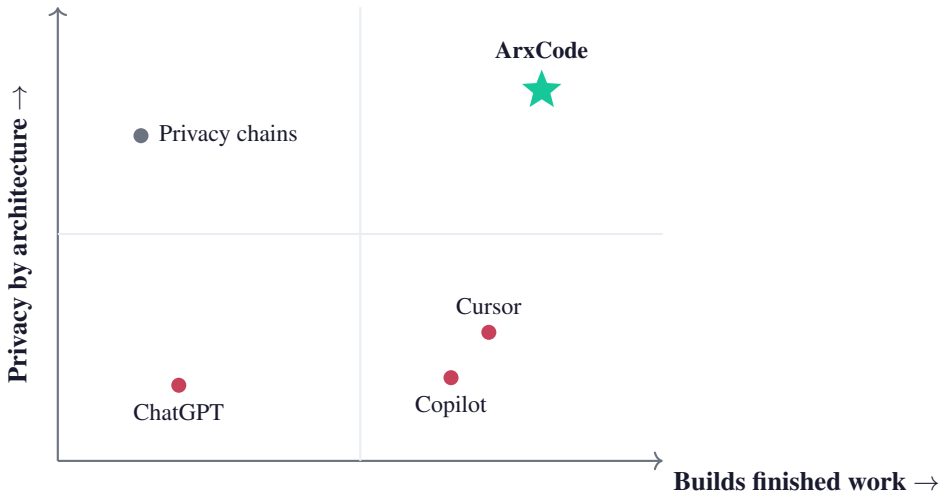

Figure 1: ArxCode occupies the quadrant no incumbent reaches—finishing real work while remaining private by construction.

3. Privacy as Policy versus Privacy as Protocol

The conventional model treats privacy as a contractual layer placed on top of an architecture 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 pinky-swear and a property. We build on the latter.

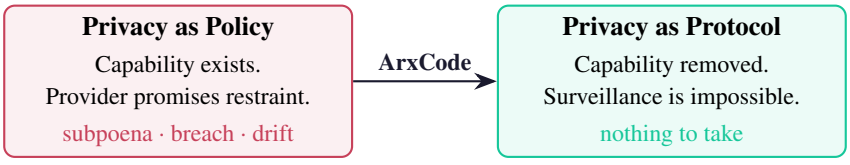

Figure 2: The shift from a promise the operator can break to a property the operator cannot violate.

4. Sealed Execution

The core of the studio 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 opaque to everyone outside it.

4.1 Guarantees

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

(i) **Input confidentiality.** x is decrypted only inside the environment and is never accessible to any outside party.

(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.

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

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 sealed environment. The environment decrypts the input internally, performs the computation, and produces the output. The output is returned to the user, encrypted to their wallet key. The environment is then 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. At no point is a copy retained.

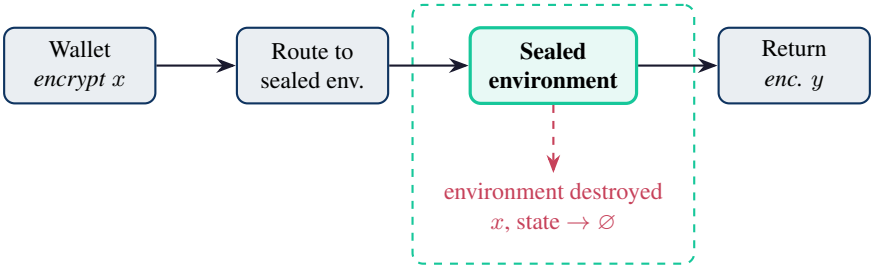

Figure 3: A build's lifecycle. Plaintext exists only inside the sealed boundary and is destroyed on completion.

5. Payment Without Accounts

A sealed system cannot rely on conventional accounts, because accounts are themselves a surveillance surface: an email address, a billing record, 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 ArxCode through a Solana wallet address and a balance of $ARX. There is no registration, no email, no KYC step, and no payment card on file. To use the studio, a user holds $ARX; to perform a build, the protocol consumes $ARX proportional to the compute expended. This is the entire transaction. No behavioral data is collected because none is required, 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—something a slower or more expensive chain could not support.

6. The Token and the Incentive

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

6.1 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 studio 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.

6.2 Earn-on-compute

New $ARX enters circulation through one channel: verified compute is delivered and rewarded. 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 genuine activity rather than a predetermined inflation schedule.

6.3 Hold-for-discount

The per-build rate a user pays declines as their $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.

6.4 Supply dynamics

Let $B(t)$ denote the cumulative $ARX burned through builds up to time t, and $M(t)$ the cumulative $ARX earned 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 earn 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 new supply, producing net contraction. The result is a supply curve governed by usage rather than by schedule.

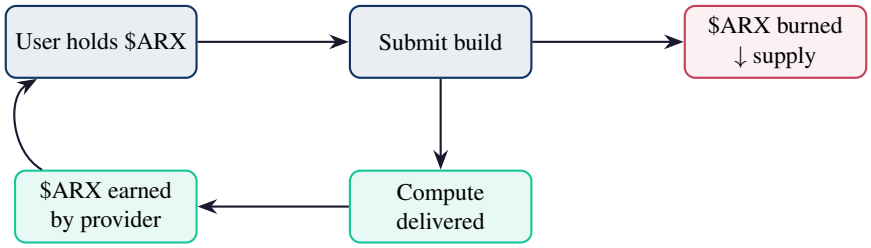

Figure 4: Tokens burn as work is consumed and are earned only when compute is delivered—supply tracks usage.

7. What ArxCode Builds

The studio 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—not isolated snippets.
- **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, READMEs, 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.

8. 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 studio gains distribution. Every build performed through such a channel consumes $ARX, binding third-party distribution to the same usage-driven token economy described above.

9. Comparison

	ArxCode	**ChatGPT**	**Copilot**	**Cursor**
Can read your data	No (sealed)	Yes	Yes	Yes
Trains on your work	No	Yes	Yes	Yes
Account / email	Wallet only	Required	Required	Required
Payment	Pay-per-build	$20/mo	$10/mo	$20/mo
Ships finished work	Yes	Partial	Partial	Partial
Settlement	On-chain	Off-chain	Off-chain	Off-chain

Table 1: ArxCode versus the closest AI alternatives.

10. Conclusion

We have described a private AI code studio 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 studio 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, subscriptions, and the data-for-service exchange they enable. By burning tokens as work is consumed and earning them only as compute is delivered, it ties supply to genuine usage rather than to emission schedules or speculation.

You use AI to build. What you build stays yours. That is the entire proposition—and it is enforced by math, not by a policy.

ArxCode · Chain: Solana (SPL) · Token: $ARX
AI Code Studio / Privacy / DePIN · `arxcode.xyz`