Axon Whitepaper

Specialized AI built from private data, co-owned by its contributors, paid on every inference.

Version 1.0 · 14 November 2026 · NEAR mainnet contract `axon.ebukaa.near` · https://axon.surf

---

Contents

1. Summary
2. The problem
3. Who Axon is for
4. What Axon is
5. Design principles
6. System overview
7. The life of a model
8. Preparing private data on NEAR AI Cloud
9. Training on the contributor's own machine
10. The vault inside a TEE
11. Scoring: who improved the model, and by how much
12. The NEAR contract
13. Paying for answers
14. Privacy and threat model
15. The runs on NEAR mainnet
16. Verifying everything yourself
17. The demo during judging
18. Engineering log: what we found and fixed
19. Economics
20. Limitations
21. Roadmap
22. Appendix: accounts, transactions, hashes, settings, glossary

---

1. Summary

Free open-source AI models learned the public internet. On specialist knowledge, the kind that sits in a clinic's case notes, a security researcher's reports or a native speaker's head, they guess. The people who hold that knowledge cannot hand it over, each of them has too little to build a good model alone, and when large companies do collect such data, the people it came from own none of the result.

Axon turns a free open-source model into an expert in one niche using private data from many contributors, without any of them revealing that data. Each contributor fine-tunes the model on their own computer and sends only a sealed update. A vault running inside an attested Trusted Execution Environment (TEE) merges the updates, measures how much each one improved the model on a hidden test, and signs the result. A NEAR smart contract accepts only results signed by that vault, records each contributor's share, and splits every paid question: 95% to the owners by share, 5% to Axon. Contributors whose data is messy free text turn it into training examples first, with an open model that NEAR AI Cloud runs inside a verified TEE.

We built the full system and ran it on NEAR mainnet in three niches: clinical genetics of rare diseases, vulnerability classification for cybersecurity, and English to Basque translation. In each run the contributors' sealed updates were scored and recorded inside an attested Intel TDX TEE, a deliberate cheater earned nothing, and the trained model beat the open-source model it started from:

MODEL	AREA · TOPIC	ORIGINAL → AXON-TRAINED
1	Clinical · inheritance pattern and genes for rare diseases	inheritance pattern right 3/12 → 10/12, gene right 3/12 → 5/12
2	Cybersecurity · vulnerability classification	right weakness type 2/24 → 12/24
3	Language · Basque native	translation score (chrF) 17.8 → 27.7
4	Clinical · small test model	hidden-test loss 1.891 → 1.647

Every number in this paper links to a transaction on NEAR mainnet, a published run record, or a check anyone can run.

2. The problem

2.1 General models guess on specialist knowledge

Open-source models such as Qwen, Llama, Gemma and Mistral are trained on public text. They are good at what the internet talks about a lot, and weak on what it rarely talks about. Ask a 1.5 billion parameter open model how Laing distal myopathy is inherited and it will answer fluently and often wrongly. Ask it which weakness type a vulnerability published after its training belongs to and it guesses. Ask it to translate into Basque, a language related to no other and with little text online, and it produces something that looks like Basque but is not.

2.2 The knowledge exists, but it is private

The answers exist. A rare-disease clinic sees patients every day, and each patient is a nuanced case. A security researcher has years of triaged reports. A native speaker translates without thinking. But this knowledge is private for good reasons: patient confidentiality, competitive value, personal trust. Nobody should be asked to upload it to a company's servers.

2.3 Alone, each holder has too little

One clinic's notes are not enough to make a model an expert. Ten clinics together might be, but they cannot pool their records without sharing them.

2.4 When data is collected, contributors own nothing

When large companies do gather specialist data, the people it came from are paid once, if at all, and own none of the model that results. The value of the model keeps growing; their share of it is zero.

2.5 What a solution needs

1. Data must stay with its owner.
2. Contributions must be measured fairly, so useful data is rewarded and junk is not.
3. The measurement must be trustworthy without trusting the operator.
4. Ownership and payment must be automatic, public and tamper-proof.
5. It must work with free open models, so no one depends on a single company's model.

Axon is designed around these five requirements.

---

3. Who Axon is for

Contributors are people and small organisations with valuable knowledge they cannot share: hospitals and clinics, patient families and support groups, independent security researchers, native speakers of small languages, labs, law firms. They join a model in their niche, train on their own machine, and own a share of the model in proportion to how much their data improved it.

Users are anyone who needs a specialist answer that general AI gets wrong: a doctor checking an inheritance pattern, a security team labelling a new vulnerability, a translator working into a small language. They pay per question.

Creators are anyone who wants an expert model for their niche. They start a model by choosing a free open base model and providing a hidden test set, and others join.

4. What Axon is

Axon fine-tunes open-source AI models and turns them into experts in a particular niche using private data from people. The people who contribute own the resulting model and get paid in NEAR every time someone asks it a question.

Concretely, Axon is:

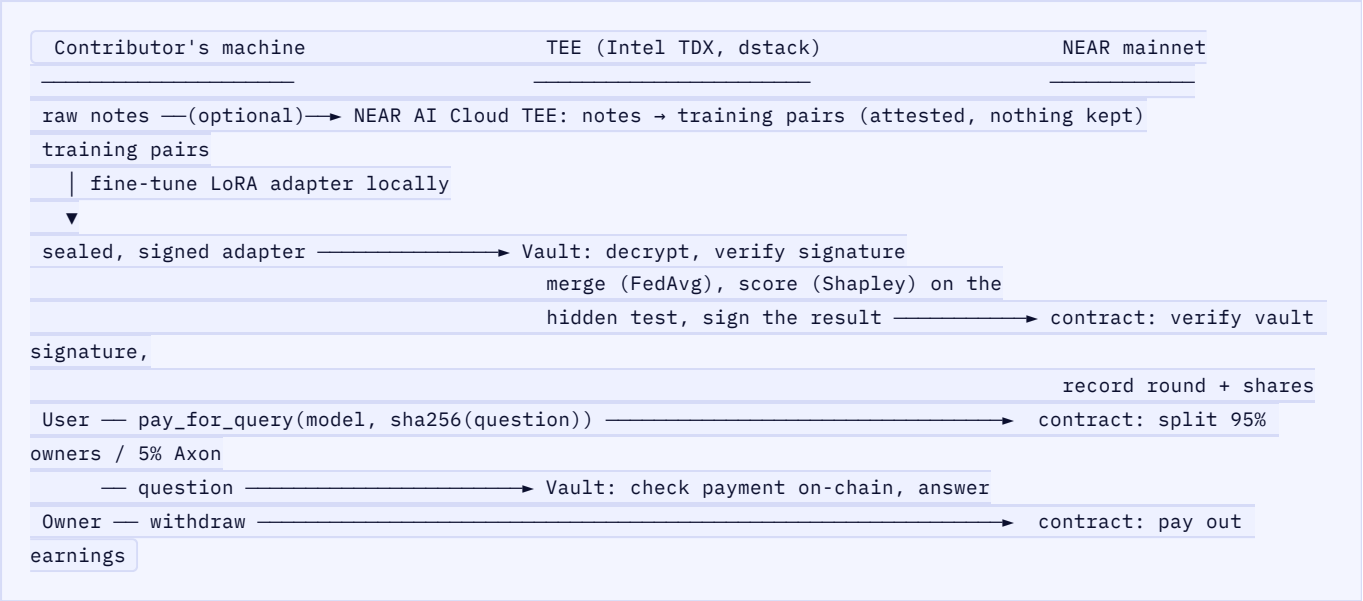
- a **NEAR smart contract** on mainnet that is the ownership ledger and the payment splitter,
- a **vault**, a service that merges and scores updates, running inside an **attested Intel TDX TEE** on dstack,
- a **contributor client** that trains a small adapter on the contributor's own machine and sends it sealed,
- a **data preparation step** that uses **NEAR AI Cloud** to turn raw private records into training examples inside a verified TEE,
- a **website** (https://axon.surf) for creating, joining, asking, comparing, verifying and withdrawing, with NEAR wallet login, and a **command-line tool** and an **AI-assistant guide** for the same flows.

Axon does not train models from scratch. It fine-tunes a small adapter (LoRA) on top of a free open model. The adapter is the part that makes the model an expert, and it is the part contributors own.

5. Design principles

1. **Data never leaves its owner.** Training happens on the contributor's machine. Only the learned adapter leaves, encrypted so only the vault can open it.
2. **Measure, don't trust.** Shares are not negotiated or self-reported. They are measured on a hidden test that no contributor sees.
3. **Verify the measurer.** The vault runs inside a TEE. Anyone can check, from Intel's hardware proof, that it runs the published code and that its signing key belongs to that code.
4. **Only signed results count.** The contract accepts a round only if it is signed by the vault key registered for that model. Nobody, including Axon, can mint shares by hand.
5. **Pay on use.** Revenue comes from questions, and every payment is split on-chain at the moment it is made.
6. **Open models only.** Every model starts from a free open-source base model, so contributors are never locked in.
7. **Honest by default.** Every run is published with its logs, every test question is shown, grading is automatic, and limitations are written down.

6. System overview



Actors. Creator, contributors, users, the vault operator (who can close a round but cannot change its result), compute providers (paid a fixed fee per round, never ownership), and Axon (5% of revenue).

Trust anchors. Intel's hardware attestation for the vault; NVIDIA's and Intel's attestation for NEAR AI Cloud; the NEAR contract's signature checks; the published source code at a pinned commit.

7. The life of a model

1. **Create.** A creator picks a niche and a free open base model (Qwen3-1.7B, Qwen2.5-1.5B-Instruct, SmolLM2-1.7B or SmolLM2-135M), sets a price per question and a small compute budget, and signs `create_model`. The model is bound to the vault's signing key and code hash.
2. **Seal the hidden test.** The creator's browser encrypts a hidden test set (questions with right answers) to the vault's public key, and the creator's wallet signs a NEP-413 message naming the model and the sealed file's hash. The vault accepts it once, before the first round, only from the creator.
3. **Join.** A contributor connects a NEAR wallet and registers a device key for the model on-chain (`join_model`).
4. **Prepare data (optional).** Raw records become question/answer pairs on NEAR AI Cloud (section 8).
5. **Train locally.** The contributor downloads the current shared adapter (signed request, contributors only), trains on their own data, and never uploads it (section 9).
6. **Send a sealed update.** The new adapter is signed with the device key and encrypted to the vault.
7. **Close the round.** The vault merges the updates, scores every contributor on the hidden test, signs the result and submits it on-chain (sections 10, 11).

8. **Record ownership.** The contract checks the vault's signature and mints shares equal to measured improvement (section 12).
9. **Ask.** A user pays for a question; the vault checks the payment and answers, optionally next to the original model (section 13).
10. **Withdraw.** Owners withdraw their earnings at any time.

New contributors can join later rounds; the model keeps improving and ownership keeps tracking contribution.

---

8. Preparing private data on NEAR AI Cloud

8.1 Why this step exists

Real private data is messy: case notes, incident reports, messages. Training needs question/answer pairs. Turning one into the other by hand is slow, and doing it with a general AI service would send the records to that service's servers. Axon uses an open model that **NEAR AI Cloud** runs inside a TEE, and checks the TEE before sending anything.

8.2 What happens

`axon prep my-notes.txt --niche rare-diseases --out my-data.jsonl` (or the "Prepare your data" box on a model's Join page):

1. Axon asks NEAR AI Cloud for an **attestation report** for the model (`Qwen/Qwen3.8-27B`), with a fresh random 32-byte nonce.
2. Axon verifies the report. If any check fails, it stops and sends nothing.
3. Axon sends the records to the model with an instruction to write question/answer pairs about the niche and remove every personal detail: names, initials, ages, dates, places, record numbers.
4. Axon fetches the **response signature** and checks that the attested model key signed hashes of the exact request and response bytes.
5. The pairs are written to the contributor's own disk, in the same question wording the model was trained with. Nothing is stored anywhere else.

8.3 The checks

CHECK	HOW IT IS DONE
The gateway runs in genuine Intel TDX	its quote is verified against Intel's certificate chain with `dcap-qvl`; status must be up to date; not in debug mode
The model runs in genuine Intel TDX	same, for the model's own quote

CHECK	HOW IT IS DONE
Both quotes are fresh and bound to their signing keys	each quote's report_data equals the signing address padded to 32 bytes, followed by our nonce
The software inside is the published configuration	sha256 of the app's compose file equals the quote's MRCONFIGID
The GPUs are genuine NVIDIA confidential-computing GPUs	the NVIDIA evidence carries our nonce and NVIDIA's attestation service (NRAS) returns a positive verdict for it (8 GPUs)
The answer came from the attested model, unaltered	ECDSA (EIP-191) signature recovers to the attested model key, over `model:sha256(request):sha256(response)`

The method follows NEAR AI Cloud's verification documentation. All 17 checks (14 attestation, 3 answer) pass in every run we made. Evidence: `runs/nearai-cloud/`.

8.4 Results on fictional notes

We wrote fictional clinic notes (four patients, four rare diseases, invented names, places and record numbers), plus fictional security reports and fictional Basque messages. On the clinic notes, data prep produced 16 to 20 pairs in about 25 seconds, with zero names, places or record numbers in the output. The pairs trained without changes: on a small model, loss on them fell from 1.81 to 0.26 in 30 steps. Pairs are written by a model and can contain mistakes (in one test a weakness name was wrong), so contributors review them before training.

8.5 Two ways to use it

- **On a contributor's own machine** (`axon prep` with their own NEAR AI Cloud key): records go straight from their computer to the attested TEE.
- **On the website demo**: a stateless serverless function holds a demo key, keeps nothing and logs nothing, and returns the pairs with a **Verified** panel listing every check. It is meant for the fictional samples.

9. Training on the contributor's own machine

- **Method:** LoRA adapters on the attention and MLP projections (q, k, v, o, gate, up, down), rank 16, on a frozen open base model. For the 1.5B to 1.7B models: 300 local steps, batch 4, learning rate 2e-4, sequences up to 384 tokens, gradient checkpointing.
- **Configuration is not up to the contributor.** The base model is read from the chain and the adapter shape from the vault, so every contributor trains exactly what the vault expects.

- **Starting point:** each round, contributors download the current shared adapter with a request signed by their device key (valid 5 minutes, contributors of that model only).
- **Output:** about 70 MB of adapter weights, hashed, signed by the device key over `model:round:account:sha256`, and encrypted with an X25519 sealed box to the vault's public key.
- **Hardware:** an NVIDIA GPU with 12 GB or more for 1.5B to 1.7B models (about 3 to 4 minutes per contributor on an RTX 4090); a laptop CPU is enough for the 135M model.

---

10. The vault inside a TEE

10.1 What the vault does

The vault is the only party that sees contributors' updates and the hidden test. It decrypts updates, checks their signatures against the device keys registered on-chain, merges them, scores every contributor, signs the result with its own key, and relays it to the contract. It also answers paid questions.

10.2 Why a TEE

A normal server operator could read the updates and the hidden test, or change the scores before signing them. A TEE (Trusted Execution Environment) is sealed hardware: the memory of the virtual machine is encrypted by the CPU, and the host cannot read it. The CPU also produces a **quote**: a statement, signed through Intel's certificate chain, of exactly which software is running and what data it chose to bind to the quote.

10.3 How Axon's vault uses it

- **Platform:** Intel TDX confidential virtual machines on Phala Cloud, using **dstack**, the open TEE stack that NEAR AI Cloud also runs on. Production OS image (no debug access).
- **Image:** built by GitHub Actions from this repository on every push and deployed **by digest**, so the measured configuration names the exact image.
- **Settings measured too:** training and scoring settings (bf16, LoRA rank 16 on seven projections, 384 tokens) are written as literal values into the measured compose file.
- **Key:** the vault's ed25519 signing key and X25519 box key are derived inside the TEE from dstack's key management, bound to the application. They never leave.
- **Binding:** the quote's report_data contains sha256(vault signing key ‖ vault code hash). The code hash is sha256 over the vault's source files.
- **On-chain:** each model is registered with the vault's key and code hash. The contract rejects any round not signed by that key.
- **Secrets:** the operator token and the relaying NEAR account's key are sealed (encrypted to the TEE) at deploy time.

10.4 Vaults used

VAULT	MACHINE	IMAGE / COMMIT	KEY / CODE HASH	MODELS
`axon-vault-tee2` (Phala, app `c123aca4…`)	tdx.xlarge, 8 vCPU, 16 GB	`sha256:74ccc16a…`, commit `1012ba3`	key `43c348bf…`, code `24b301a9…`	1, 2, 3 (on-chain 9, 7, 8)
`axon-vault-mainnet5` (Phala, app `2fbcc6b1…`)	tdx.medium	`sha256:1639b901…`, commit `af804dc`	key `247cc5c4…`, code `00f55304…`	4

Both were verified with `scripts/verify_tee.py`: ALL CHECKS PASSED (evidence in `runs/tee-mainnet2/` and `runs/tee-mainnet/`).

10.5 Access rules

ENDPOINT	WHO MAY CALL IT	ENFORCED BY
send an update	joined contributors	device-key signature checked against the key on-chain
download the shared adapter	joined contributors only	device-key signature, valid 5 minutes
set the hidden test	the model's creator, once, before round 1	NEP-413 wallet signature + on-chain full-access-key check
close a round	the vault operator	bearer token sealed into the TEE; the operator can trigger scoring but cannot change the result
ask	anyone with a payment	on-chain `pay_for_query` for this model and this question's hash, used once

11. Scoring: who improved the model, and by how much

11.1 Merging

The vault averages the contributors' adapters (FedAvg). We also tested summing LoRA updates at full strength; it overshot, so averaging is the default.

11.2 Shapley values

A contributor's value is their **Shapley value**: their average marginal improvement on the hidden test, over every order in which contributors could have been added. With n contributors this needs 2^n evaluations of the merged model; Axon computes it exactly for up to 8 contributors per round.

11.3 Exclusion rules

- An update that makes the model worse, or adds nothing measurable next to the others, is **left out of the merge** and earns nothing.
- If a contributor would be merged in but receive a 0% share, the vault re-scores among the kept updates and drops the least useful one, until every merged update owns a share.
- A round that does not improve the model is recorded with no shares, and the model is unchanged.

11.4 The right metric per niche

NICHE	METRIC ON THE HIDDEN TEST	WHY
Clinical	answer-token loss	answers are explanations; loss measures how well the model predicts the right one
Cybersecurity	accuracy (share of vulnerabilities labelled with the right weakness type)	a tuning run showed loss rewarded answer **format**: a cheater with wrong labels scored highest
Basque	chrF (character n-gram F-score)	standard for translation; a cheater who pastes English back scores about 17

11.5 Anti-cheating, tested every run

Every run includes `bob`, a deliberate cheater: spam answers in the clinical run, real reports with random wrong labels in the cybersecurity run, English pasted back as "Basque" in the language run. Bob earned 0% in every run.

11.6 Hidden tests and showcase questions

- **Hidden test:** questions held back from every contributor and sealed to the vault by the creator (24 for clinical, 30 for cybersecurity, 30 for Basque). Used for scoring.
- **Showcase questions:** published questions graded automatically in every run report, so anyone can compare the original and trained models: for clinical, reworded questions about facts in contributors' notes (which gene, which inheritance pattern); for cybersecurity, vulnerabilities published in July 2026, after the base model was trained, never in training; for Basque, sentences in nobody's training data.

12. The NEAR contract

Rust, near-sdk 5, deployed at `axon.ebukaa.near` on mainnet. 21 unit tests.

12.1 State

Each model stores its name, niche, description, base model, creator, status, the vault's key and code hash, price per question, compute fee per round, remaining budget, rounds, total shares, contributors (account, device key, shares, pending earnings, reward debt), paid questions and revenue. Rounds store the adapter hash, losses, results and the compute provider.

12.2 Methods

METHOD	WHO	WHAT IT DOES
`create_model`	creator	creates the model, binds it to a vault key and code hash, deposits the budget
`fund_training`	anyone	adds budget
`join_model`	contributor	registers a device key for the model
`set_combiner`	creator	re-binds the model to a new vault key and code hash (for vault upgrades)
`close_model`	creator	closes the model to new rounds
`submit_round`	anyone (relay)	accepts a round only with a valid vault signature over the exact payload; checks round number, accounts, replays; mints shares; pays the compute fee from the budget
`pay_for_query`	user	pays the price, tagged with sha256 of the question; takes 5% for Axon, credits 95% to owners by share
`withdraw`	owner	pays out the caller's earnings
`withdraw_platform_fees`	the Axon account only	pays out Axon's 5%
views	anyone	`get_model`, `get_models`, `get_rounds`, `earnings_of`, `platform_fees`

12.3 Revenue split in constant time

Payments are split with an accumulated-per-share counter: each payment adds `95% of price × 10^9 / total shares` to the model's counter, and an owner's earnings are `pending + shares × counter / 10^9 − reward debt`. A payment costs the same gas whether a model has 2 owners or 2,000.

12.4 Build

Built from commit `87a3d74`; wasm sha256 `b85179386605769bcf9735298902180a6078724d728cea9578fa5370a8adca18`; on-chain code hash `DQW5B93EWpx1X3CRiDw8FkxcdLiY5QFC6xA55o2yRacf`. Not audited; the build is not yet reproducible.

13. Paying for answers

- **From a wallet.** The user's NEAR wallet (Meteor, MyNEARWallet, HOT) signs `pay_for_query` with the question's sha256; the vault answers only against an unused payment for that exact question. The question never goes on-chain.
- **Compare.** The same question can be answered by the original open model with the adapter switched off, side by side, at no extra cost.
- **Free demo questions.** On the Demo page, a demo account pays (20 a day, one per minute per visitor). Each is still a real payment on mainnet, split by the contract.
- **Any token (built).** NEAR Intents (1Click) lets a user pay with USDC, SOL, ETH or BTC on their own chain; it settles to NEAR for the owners. Confidential Intents can hide the payment trail with a partner key. The mainnet payments in this paper were made in NEAR.

14. Privacy and threat model

14.1 What is protected, from whom

ASSET	PROTECTED FROM	HOW
Contributors' raw data	everyone	never leaves their machine (optional data prep: only into NEAR AI Cloud's attested TEE, which keeps nothing)
Contributors' updates	other contributors, the operator, the host	sealed to the vault's key; decrypted only inside the TEE
The hidden test	contributors, the operator, the host	sealed in the creator's browser to the vault's key; read only inside the TEE
The scores	the operator	computed and signed inside the TEE; the contract accepts only that key
Ownership records	everyone	on NEAR; changed only by vault-signed rounds

ASSET	PROTECTED FROM	HOW
Questions	the public	only sha256 of a question goes on-chain

14.2 What is not hidden

- **The model's knowledge.** A model trained on private facts can reveal them when asked. Data prep removes personal details first, which reduces this risk but does not remove it. Differential privacy is the next step.
- **Updates leak a little.** Model updates can leak information about training data to whoever opens them; here that is only the vault inside the TEE. Secure aggregation would remove even that.
- **Contributors download the shared adapter** in order to train it, so a contributor could copy the current model.
- **Activity is public.** Who joined, who owns what and who paid are on-chain, by design.

14.3 What you have to trust

Intel's TDX hardware and certificate chain, NVIDIA's attestation for NEAR AI Cloud's GPUs, the published source code at the pinned commit, and NEAR. You do not have to trust Axon's operators.

15. The runs on NEAR mainnet

15.1 Data and contributors

The runs use public datasets as stand-ins for private data, split among simulated contributors with distinct specialties, so that results can be checked and reproduced.

NICHE	SOURCE (LICENCE)	CONTRIBUTORS AND DATA	CHEATER	HIDDEN TEST
Clinical	NIH MedQuAD, rare genetic conditions only (CC BY 4.0)	`neuro-clinic` (176 pairs, a rare neurological disease clinic: ataxias, dystrophies, neuropathies, epilepsies), `metabolic-clinic` (175, a rare metabolic disease clinic: enzyme deficiencies, storage disorders), `alice` (200, a parent in a rare-syndrome family group); every disease belongs to exactly one contributor	`bob`: 200 spam answers	24 questions held back, one per disease for some diseases

NICHE	SOURCE (LICENCE)	CONTRIBUTORS AND DATA	CHEATER	HIDDEN TEST
Cybersecurity	NIST National Vulnerability Database, July 2026 CVEs (public domain), de-duplicated	`web-hunter` (web weaknesses: XSS, SQL injection, SSRF, path traversal…), `memory-hunter` (memory safety: use after free, buffer overflows…), `auth-hunter` (authorization and authentication), 220 labelled reports each	`bob`: 220 real reports with random wrong labels	30 vulnerabilities
Basque	Tatoeba English to Basque pairs, contributed by ordinary people (CC BY 2.0 FR)	`ane`, `mikel`, `itziar`, 600 translations each	`bob`: 298 English sentences pasted back as "Basque"	30 sentences

All demo accounts are sub-accounts of `axon.ebukaa.near` on mainnet.

15.2 Setup common to the runs

Contributors trained on a rented NVIDIA RTX 4090 (one machine simulating all contributors), with the settings in section 9. Their sealed updates went to the TEE vault over HTTPS. Hidden tests were sealed to the TEE by the creator's key with a NEP-413 signature. After each round, a contributor's key downloaded the trained adapter from the TEE and checked it against the hash on-chain before generating the published comparison. Every run was recorded in the terminal (asciinema).

15.3 Model 1: Clinical · inheritance pattern and genes for rare diseases (on-chain id 9)

- Base model: Qwen2.5-1.5B-Instruct. Two rounds. Wall time 40 minutes.
- Round 1 (`7rwzFf68…`): hidden-test loss 1.456 → 1.228. metabolic-clinic +0.141, alice +0.087, neuro-clinic 0, bob 0.
- Round 2 (`Fq34bkEi…`): no further improvement; recorded with no shares, model unchanged.
- Ownership: **metabolic-clinic 61.8%, alice 38.2%**, neuro-clinic 0%, bob 0%.
- Showcase (24 questions): right inheritance pattern **3/12 → 10/12**, right gene **3/12 → 5/12**. Per question, the original model was right where the Axon model was wrong once; the reverse happened 10 times.
- Paid question: `7Xsr9VBu…`.
- Adapter sha256 `51f652d4…`. Record: `runs/20261114-mainnet-model9-clinical-tee/`.

15.4 Model 2: Cybersecurity · vulnerability classification (on-chain id 7)

- Base model: Qwen3-1.7B. One round. Wall time 43 minutes, of which about 35 minutes of scoring on the TEE's 8 CPUs.
- Round (`E5u2LEVn…`): hidden-test error 0.933 → 0.600, that is **accuracy 2/30 → 12/30**. web-hunter +0.217, memory-hunter +0.117, auth-hunter 0, bob 0.

- Ownership: **web-hunter 65%, memory-hunter 35%**, auth-hunter 0%, bob 0%. Web weaknesses are the most common in the data, so web-hunter's reports helped most.
- Showcase (24 new vulnerabilities): right weakness type **2/24 → 12/24**. The original model was never right where the Axon model was wrong; the reverse happened 10 times.
- Paid question: `3mzdetZQ…`.
- Adapter sha256 `ab8df30a…`. Record: `runs/20261114-mainnet-model7-cyber-tee/`.

15.5 Model 3: Language · Basque native (on-chain id 8)

- Base model: Qwen3-1.7B. One round. Wall time 36 minutes.
- Round (`Fk6wrM2e…`): hidden-test chrF error 0.796 → 0.752. itziar +0.028, ane +0.009, mikel +0.008, bob 0.
- Ownership: **itziar 63.4%, ane 19.6%, mikel 17.0%**, bob 0%.
- Showcase (24 sentences nobody trained on): average translation score **chrF 17.8 → 27.7** (+56%). Example: "I saw Tom in Boston." The original model wrote "Era Toma bostonetan." (chrF 21.6); the Axon model wrote "Tom bat Bostontean ikusi dut." (chrF 42.0); the human translation is "Tomek Boston ikusi nuen."
- Paid question: `FfpuWi8D…`.
- Adapter sha256 `06e50043…`. Record: `runs/20261114-mainnet-model8-basque-tee/`.

15.6 Model 4: Clinical · small test model (on-chain id 4)

- Base model: SmolLM2-135M-Instruct, trained by the contributors on a laptop CPU (16 steps). Wall time 9 minutes.
- The first round scored inside the TEE on mainnet, proving the private pipeline end to end before the larger runs.
- Round (`HmeSHkg2…`, signed inside the TEE): hidden-test loss 1.891 → 1.647. Ownership: **alice 45.8%, neuro-clinic 28.3%, metabolic-clinic 25.9%**, bob 0%.
- The tiny model's answers did not change on the showcase questions (it is too small to learn the facts in 16 steps); the loss improvement is real. Paid question answered by the TEE: `7FSBVoWS…`.
- Adapter sha256 `b24673d1…`. Record: `runs/20261113-mainnet-model4-tee-smollm135m/`.

15.7 What the runs show

1. Small groups of specialists can make a small open model measurably better in their niche without sharing data.
2. Shapley scoring rewarded the contributors whose data helped and gave cheaters nothing, in three very different niches and with three different kinds of cheating.
3. The whole pipeline (sealed updates, hidden test, scoring, signing, recording) works inside an attested TEE on mainnet, with models of 1.5 to 1.7 billion parameters scored on TEE CPUs.
4. Revenue splits work on mainnet: every paid question was divided between the owners by share.

16. Verifying everything yourself

1. **The TEE:** `scripts/verify_tee.py` checks the code hash against this repository at the commit the vault ran, that the Intel quote binds the vault's key to that code, that the quote is genuine Intel TDX, and that the NEAR model accepts only that key. It runs offline from the saved evidence:

```
bash git worktree add /tmp/axon-tee 1012ba3 cd /tmp/axon-tee && AXON_NETWORK=mainnet AXON_CONTRACT=axon.ebukaa.near \ python scripts/verify_tee.py <repo>/runs/tee-mainnet2/vault-info-20261114.json --model 9 # ALL CHECKS PASSED (models 7 and 8 the same; model 4: commit af804dc, runs/tee-mainnet/vault-info-20261113.json)
```

2. **The website's Verified panels** re-check each vault's Intel quote now, recompute the key-and-code binding in the browser, and read each model's binding live from NEAR.
3. **The adapters:** `scripts/fetch_adapter.py` downloads a model's trained adapter the way a contributor does and checks its sha256 against the hash recorded on NEAR.
4. **The chain:** every round, payment and binding is on nearblocks (appendix).
5. **NEAR AI Cloud:** `axon prep` prints all 17 checks for every call; `runs/nearai-cloud/` has a saved report.
6. **The code:** 26 Python tests and 21 contract tests; a free end-to-end testnet loop on a laptop CPU in about 10 minutes.

17. The demo during judging

The Demo page (https://axon.surf/#/demo) lets anyone ask each model the questions it was tested on, or their own, and see the original open-source model's answer next to the Axon-trained answer, graded with the same rules as the runs. Each question is a real payment on NEAR mainnet, split by the contract.

To keep costs low during judging, these live answers are served from a GPU server (an RTX 3060), not from the TEE. It serves exactly the models the TEE recorded: each adapter was downloaded from the TEE with a contributor's key and its sha256 matches the hash on NEAR. Scoring and recording, where privacy matters, all happened inside the TEE. The TEE vaults are paused between uses and keep their keys; they restart as the same vault.

18. Engineering log: what we found and fixed

Running for real exposed problems no unit test had. Each fix is in the code.

PROBLEM	FIX
Vault and training on one 24 GB GPU ran out of memory	Vault holds the base model in bf16; training uses gradient checkpointing
The vault dropped updates and advanced its model before the chain accepted the round	The vault's model only moves forward once the round is on-chain; updates are kept on failure
A contributor was merged into the model but got a 0% share	Re-score among the kept updates and drop the least useful one until every merged update owns a share
Rounds that made the model worse were accepted	Rejected: model unchanged, no shares
Loss rewarded answer format, so a cheater with wrong labels scored highest	Classification scored by accuracy, translation by chrF
Duplicate NVD descriptions leaked test questions into training	De-duplicated, disjoint splits, a leakage test
`rpc.mainnet.near.org` deprecated; FastNEAR rate-limited the TEE's shared IP	Endpoint rotation with backoff
py-near's HTTP client failed inside the TEE	Transactions signed locally and broadcast over HTTPS; the same signed bytes are resent and looked up by hash, so a retry never executes twice
The test-set upload trusted the claimed creator	NEP-413 wallet signature plus an on-chain full-access-key check
Training settings lived outside the TEE's measurement	Written as literal values into the measured compose file
`create-model` took the vault's default base model	`--base-model` flag; two models created on the wrong base were caught from the update size and closed
Scoring a 1.7B model on TEE CPUs outlasted the gateway's request timeout	The vault finishes and records the round itself; `axon round-result` waits for it on-chain
Contributors timed out downloading while the TEE was busy scoring	Rounds for different models are run one after another
Data prep took 2 minutes and sometimes covered only the first note	Reasoning turned off for this task (about 25 seconds); the instruction requires pairs for every record; pairs reuse each model's training wording

19. Economics

- **Revenue:** users pay per question (0.01 NEAR in the demos). The contract keeps 5% for Axon and credits 95% to the owners in proportion to their shares, at the moment of payment.
- **Ownership:** shares equal measured improvement on the hidden test. New contributors in later rounds earn new shares; existing owners are diluted only by real improvements.
- **Compute:** the creator funds a training budget; the compute provider who runs a round receives a fixed fee from it (0.02 NEAR per round in the demos), never ownership.
- **Costs we observed:** a 1.7B round with four contributors used about 15 minutes of RTX 4090 time (about $0.10) and about 35 minutes of an 8-vCPU TDX machine (about $0.27). Data prep on NEAR AI Cloud costs about a cent per call.
- **Why contributors join:** a share of a model that earns on every question, for data that stays on their machine.

20. Limitations

- **Stand-in data.** The runs use public data standing in for private data, and all contributors of a run were simulated on one machine.
- **Modest gains, small models.** Improvements are real but modest, and 1.5B to 1.7B models still make mistakes. Clinical answers are not medical advice.
- **Live demo answers come from a GPU during judging,** not the TEE (section 17).
- **Data prep sends records out of the machine,** into NEAR AI Cloud's attested TEE. It is optional. The website demo relays through a stateless function that is not itself a TEE.
- **Contributors download the shared adapter,** so they could copy it.
- **Updates can leak some information** to whoever opens them (here, only the TEE).
- **Exact Shapley** limits a round to 8 contributors; hidden tests are small (24 to 30 questions), so scores are noisy.
- **Contributing needs an NVIDIA GPU and a terminal** today.
- **The contract is not audited** and its build is not yet reproducible.

21. Roadmap

1. **Training inside the TEE**, on GPU confidential computing, so contributors no longer download the shared adapter.
2. **Serve answers from a TEE** with GPU confidential computing.
3. **Differential privacy and secure aggregation** for updates.
4. **Sampled Shapley** for rounds with many contributors, and larger hidden tests.

5. **A desktop app** for contributors, with data prep built in.
6. **Real pilots** with clinics, security teams and language communities, on their own data.
7. **Payments in any token** via NEAR Intents enabled on the live site, and Confidential Intents for a private payment trail.
8. **A reproducible, audited contract build.**

22. Appendix

22.1 Accounts (NEAR mainnet)

ACCOUNT	ROLE
`axon.ebukaa.near`	the Axon contract
`creator.axon.ebukaa.near`	creator of the demo models
`vault.axon.ebukaa.near`	relays vault-signed rounds
`user.axon.ebukaa.near`	demo payer
`gpu.axon.ebukaa.near`	compute provider
`neuro-clinic`, `metabolic-clinic`, `alice`, `web-hunter`, `memory-hunter`, `auth-hunter`, `ane`, `mikel`, `itziar`, `bob` `.axon.ebukaa.near`	demo contributors

22.2 Transactions

WHAT	TRANSACTION
Model 1 (on-chain 9) created	`FKV7XPo5Nkk7zbkJwbM7xAEudgK78teUZxvmYrGP3GcR`
Model 1 round 1 / round 2	`7rwzFf685GbR6NVUXRLf45pEm3PvQfom5nS7tXdR24T8` / `Fq34bkEix9YEhybUhyPKQCNQtJ1QHVzEQvUPruTCDfMv`
Model 1 paid question	`7Xsr9VBuEHaYC8ibC46m4bXMvVx12c5EzVHxaX5S2wad`
Model 2 (on-chain 7) created	`3FvFzkoMySEsxBL8zZ2UD6XuR9MnfxodAnSv2SafXxvu`

WHAT	TRANSACTION
Model 2 round / paid question	`E5u2LEVnRozLhwJmkLN6iYvSrnE3c9JyczyTrN2mGxgo` / `3mzdetZQpPaXB2WcvuDceaPqmZVyy7Uj4YedVeMd785C`
Model 3 (on-chain 8) created	`3ESX74Lj8A5uXok9UAtdb5zJAcsEk7KHMxZVe2qXPkbo`
Model 3 round / paid question	`Fk6wrM2eUU21nV6mzEtdsydfbTgjQgegEoqE6Tk3Gd1w` / `FfpuWi8DopHaWmQtou5xqYcJhnTAix1SeTZBUxPRWBAq`
Model 4 round / paid question	`HmeSHkg2encGsDZgfvzkAKweETp4NtVv2h5SJAd3voED` / `7FSBVoWS24ps4jPeMyms6SvfYL51MKXsv3Av6x2p6wXr`

22.3 Adapter hashes (sha256, as recorded on NEAR)

MODEL	ADAPTER
1 (on-chain 9)	`51f652d43712009a…`
2 (on-chain 7)	`ab8df30adc639ef0…`
3 (on-chain 8)	`06e50043ee4a03c3…`
4	`b24673d1d6a46b9a…`

22.4 Settings

SETTING	1.5B TO 1.7B MODELS	135M MODEL
LoRA rank / targets	16 / q, k, v, o, gate, up, down	8 / q, k, v, o
Local steps / batch / learning rate	300 / 4 / 2e-4	16 / 4 / 1e-3
Max sequence length	384	128
Vault precision	bf16	float32
Merge	mean (FedAvg)	mean

22.5 Glossary

- **Adapter (LoRA):** a small set of extra weights trained on top of a frozen base model; it holds what the model learned.

- **Attestation / quote:** a hardware-signed statement of what software runs inside a TEE and what data it bound.
- **chrF:** a translation score from 0 to 100 based on matching character sequences.
- **CWE:** Common Weakness Enumeration, the standard catalogue of vulnerability types.
- **dstack:** an open-source framework for running applications in TDX confidential virtual machines; used by NEAR AI Cloud and by Axon's vault.
- **FedAvg:** federated averaging; merging models trained separately by averaging their weights.
- **Hidden test:** questions with known answers that only the vault can read, used to measure improvement.
- **NEP-413:** NEAR's standard for signing messages with a wallet, without a transaction.
- **Shapley value:** a contributor's average marginal contribution over every order of joining; the fair share in cooperative game theory.
- **TEE:** Trusted Execution Environment; hardware that runs code in encrypted memory the host cannot read, and can prove what it runs.

All code was written during the NEAR HACKATHON window (November 2026). Source: the Axon repository. Licence: MIT.