



Sealroom litepaper

Per-call inference on attested, bonded confidential-GPU operators:
how a call is served, how the weights stay sealed, and how every call
is paid and split.

Edition 1 · October 2026

1. What Sealroom is
2. How a call is served
3. What keeps the weights sealed
4. Payments and splits
5. Bonds and disputes
6. \$SEALROOM
7. For owners, operators and callers
8. Parameters
9. Why Solana

1. What Sealroom is

Sealroom is a registry and payment rail on Solana for confidential inference. Model owners list encrypted models and sell them per call. Vetted datacenter operators serve them on NVIDIA H100 and H200 GPUs in confidential computing mode, inside Intel TDX confidential virtual machines (CVMs). Callers, from applications to AI agents, pay for each call in USDC over x402 without an account. The key to a model's weights reaches a machine only after that machine has shown, on-chain, that it is a registered server of a bonded operator running an approved runtime.

Solana holds the state that has to be public and enforced by code: the registry of operators, servers, GPUs and runtimes, the attestation records, each model's key-release policy, price and split, the vaults that collect payments, and the operators' bonds. Weights, keys, prompts and GPUs stay off-chain. One Anchor program, `sealroom`, holds the on-chain state, and a five-member verifier committee, each owner's key broker, an x402 facilitator, an indexer and a keeper operate around it.

2. How a call is served



Figure: The seven steps from listing a model to settling a dispute, with the instruction or service behind each one.

List. The owner encrypts the weights in chunks with AES-256-GCM under a random 256-bit data key and publishes the ciphertext on any storage. It then signs `create_model` with the manifest hash, its key broker's public key and URL, up to 8 allowed runtime measurements, the operator constraints (minimum tier, minimum bond, an allowlist of up to 16 operators), the price and the split. The Model account is the public statement of which machines may receive the key and who is paid for each call.

Onboard. Sealroom vets each operator off-chain: legal entity, datacenter certifications and a signed Operator Agreement, whose hash goes on the Operator account. The admin multisig registers the operator at a tier, and the operator posts a USDC bond of at least that tier's minimum. With the admin co-signing, the operator registers each server's platform identity (a hash of the Intel PPID/PCK) and its GPUs' UEID hashes, then registers each node: one CVM with its endpoint and the exact GPUs it claims.

Attest. The node boots the measured Sealroom runtime, which generates its enclave keys inside the CVM and binds them, with a recent Solana slot hash, into the TDX quote and the GPU evidence. Verifiers check the quote against Intel collateral, the GPU evidence against NVIDIA reference measurements and OCSP, and the platform, GPUs and runtime against the registry. Three of the five sign, and `record_attestation` stores the result for at most 24 hours.

Release. The node asks the owner's key broker for the data key and sends its evidence. The broker reads the node's accounts from Solana, applies the model's policy, checks the evidence against the hash on-chain and verifies it again. It wraps the data key to the enclave's HPKE public key, so only that CVM can unwrap it, and the weights are decrypted in CVM memory and loaded into GPU protected memory.

Serve and pay. The caller reads the node's attestation on-chain and checks that the node's TLS key is the one committed there, so the prompt terminates inside the enclave. An unpaid request gets HTTP 402 with a price payable to the lane vault for this model and node. The caller's wallet signs one USDC transfer, a facilitator settles it on Solana, and the completion comes back from the enclave. The runtime stops serving when its own attestation lapses.

Distribute. Payments collect in the lane vault until anyone calls `distribute(lane)`. One atomic transaction pays the protocol fee into the buyback vault, the serving operator's share to its payout account and every other split recipient.

Dispute. With evidence of leaked weights or attestation fraud, the model owner opens a dispute against the operator, recording the evidence hash on-chain and posting a USDC deposit. The arbitration panel, a named multisig, rules under the Operator Agreement. A ruling for the owner can freeze the operator, revoke all of its attestations and slash its bond to the claimant.

3. What keeps the weights sealed

A data key leaves the owner's broker only when every link below holds, and each link is recorded where any party can read it.

Platform pinning

Only registered hardware counts as a node. `register_platform` needs the signatures of both the admin and the operator, and a bond that meets the operator's tier. It creates a global PlatformPin keyed by the platform identity hash and a GpuPin for each GPU UEID hash, so a physical server or GPU can belong to only one operator and one platform. `register_node` then claims specific GPUs of that platform, each for exactly one node. A quote from a genuine Intel platform that no vetted operator registered, or GPU evidence from a device outside the node's claim, fails verification. Keys are therefore released only to registered servers inside vetted facilities, each under a signed contract with a bond at stake.

Measured runtime

The runtime is a reproducibly built CVM image on Confidential Containers. It holds the attestation agent, the key client, vLLM, the x402 middleware, the voucher signer and TLS termination; its

egress is limited to the key broker, the Solana RPC and the facilitator, and it runs no owner-supplied code. A runtime's 48-byte measurement enters the protocol through `allow_runtime`, which records the image name, version and 40-character source commit and takes effect after 48 hours.

`revoke_runtime` takes effect immediately. `record_attestation` accepts only a measurement that is allowed and in effect, and each model pins its own list of up to 8 measurements on top, so a node receives a model's key only when its measurement is on both lists.

Evidence bound to fresh keys and a recent slot hash

At boot the runtime generates, inside the CVM, an HPKE key pair (X25519 + ML-KEM-768), a TLS key and an Ed25519 voucher key. Their public halves form a fixed transcript of 1,280 bytes: the hybrid public key (1,216 bytes), the TLS key (32) and the voucher key (32). The node computes:

$$H = \text{SHA-256}(\text{transcript} \parallel \text{recent_slot_hash})$$

H is written into TDX REPORTDATA and used as the nonce of every GPU's evidence. The transcript's hash is committed on-chain as the attestation's `enclave_key_hash`, and `record_attestation` requires the recent slot and its hash to appear together in Solana's SlotHashes sysvar. An accepted quote therefore proves that these keys were generated inside this CVM after that slot; evidence from another boot binds other keys and does not match.

A 3-of-5 verifier committee

Five independent verifiers, each with its own signing key, run Intel DCAP quote verification and the NVIDIA local verifier with reference measurements and OCSP. They also check the registry: platform and GPU pins, the node's GPU claims, runtime activation, the committee epoch, the operator epoch and the replay sequence. Each then signs the same 200-byte message:

offset	bytes	field
0	16	"sealroom.att.v0" + NUL
16	32	program id
48	32	genesis hash
80	32	node account
112	8	verifier epoch
120	8	replay sequence
128	32	SHA-256 of the attestation commitment
160	8	recent slot
168	32	recent slot hash

The 194-byte commitment holds the platform, runtime measurement, GPU RIM hash, evidence hash, enclave key hash, TCB status, expiry, method and operator epoch. A coordinator that holds no verifier key collects the signatures and submits them as native Ed25519 instructions ahead of `record_attestation`. The program reads them through the instructions sysvar, requires three distinct members of the active VerifierSet over exactly this message, and accepts only TCB status UpToDate. Raw quotes are archived off-chain by hash; the Attestation account stores the evidence hash.

24-hour attestations

An attestation expires at most 24 hours after it is recorded, and never after the operator's unbonding date, so every serving node re-attests at least daily under the next replay sequence. Any verifier or the admin can revoke an attestation at once. Revocation is sticky: a renewal signed before it cannot revive the record. Arbitration invalidates all of an operator's attestations in one step by advancing the operator epoch that every signed commitment includes.

The owner's key broker and HPKE release

The owner runs its own key broker, so the data key never passes through a new party. On each key request the broker:

1. Reads the Attestation, Node, Platform, Operator and Model accounts directly from Solana RPC.
2. Applies the release checks: an attestation that is unexpired, unrevoked and UpToDate; a registered platform and GPUs; an enabled node; a runtime on the model's list; an unfrozen operator whose tier, bond and allowlist entry meet the model's policy; an unpaused protocol.
3. Checks that the evidence hash and key transcript match the on-chain commitment, and verifies the Intel and NVIDIA evidence again itself.
4. Writes its decision to an audit log, then wraps the 256-bit data key to the enclave's HPKE public key (X25519 + ML-KEM-768, HKDF-SHA256, AES-256-GCM), with associated data binding the deployment, model, node, attestation revision, evidence hash and request.

Every release and denial is published through the public API with the attestation account and signed revision it relied on, so anyone can compare the broker's log with the chain.

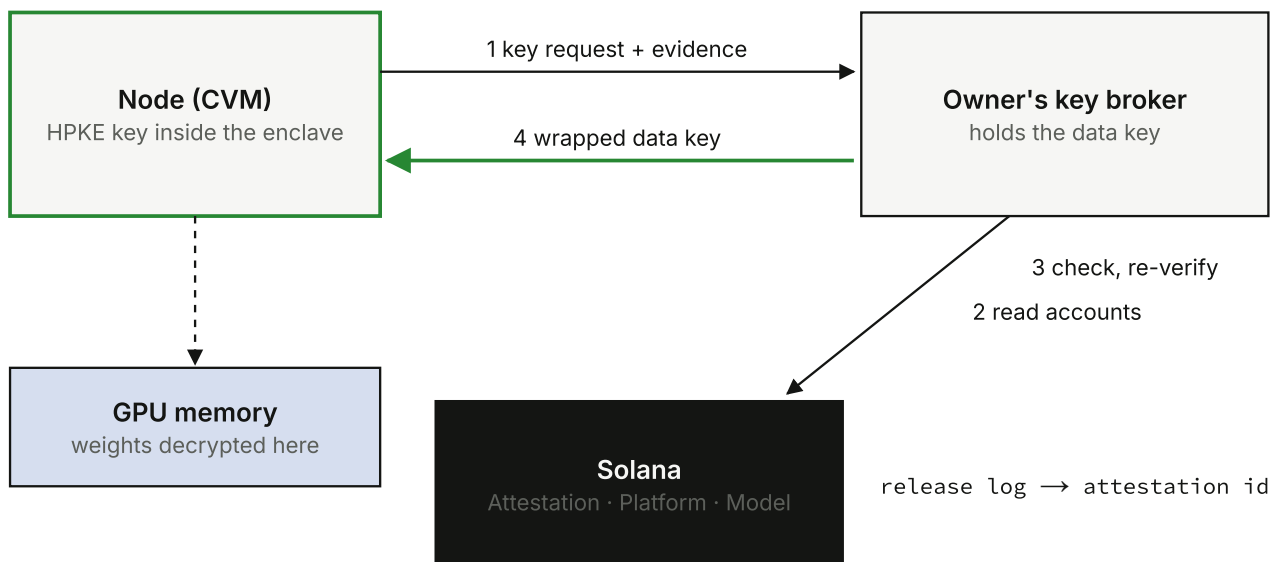


Figure: The key broker reads the chain, re-verifies the evidence and wraps the data key to the enclave's HPKE key, which only the attested CVM can open.

4. Payments and splits

x402 to the lane vault

Each model and node pair has a lane, which anyone can open with `open_lane`. The lane owns a USDC vault that receives every payment for calls on that pair. The node speaks x402 version 2: a request without payment receives HTTP 402 with a `PAYMENT-REQUIRED` header whose `payTo` is the lane vault's token account. The model's price selects the scheme.

- `exact` is a fixed price per call. The caller's wallet signs one checked USDC transfer to the vault and returns it in `PAYMENT-SIGNATURE`; the facilitator verifies it, co-signs only as fee payer and submits it. The node checks its own eligibility again before settlement, returns the completion once the transfer confirms, and never debits a repeated request twice.
- `upto` is a metered price per million input and output tokens, capped per call. The caller escrows the ceiling in the Solana Foundation payment-channels program. After the call, the enclave's voucher key signs the actual charge, binding channel, request, nonce, cumulative amount, expiry and attestation revision, and settlement pays the vault and refunds the unused ceiling in one transaction. The voucher key lives only in the enclave, so the operator cannot raise a charge.

Each call is one transfer to one recipient, the layout standard Solana x402 facilitators validate. Revenue is split later, at distribution.

One permissionless `distribute`

Anyone can call `distribute` on any lane. It allocates the vault's whole newly available balance in one transaction:

```
G = vault_balance - reserved_amount
fee = floor(G * protocol_fee_bps / 10000)
net = G - fee
share_i = floor(net * bps_i / 10000)
share_last = share_last + (net - sum(share_i))
```

A split has up to 8 entries whose weights sum to exactly 10,000 bps, and exactly one has the Operator role. The model declares the operator's weight, and the program pays it to whichever operator runs the lane's node, so a model serving on several operators pays each its own share. The other entries carry the roles Owner, BaseModel or Contributor and keep their declared USDC accounts. The rounding remainder goes to the last entry, so the fee and shares add up to G to the base unit. A recipient whose token account cannot receive has its exact share reserved in the lane as a claim for its beneficiary, while the other payouts proceed. A distribution paying the fee and eight split recipients fits in one 308-byte transaction.

The owner changes a split with `propose_split`, which can be applied after 7 days. A locked entry can be lowered only with its beneficiary's signature. A base model can set a `derivative_royalty_bps`, and every model built on it must then carry a locked BaseModel entry of at least that weight paying the base owner. Each lane switches to a new split only after all revenue

received under the old one has been allocated; a keeper service applies mature proposals and moves each lane across that boundary.

Worked example

Take the launch protocol fee of 500 bps, a lane vault holding 100 USDC (100,000,000 base units) of newly available revenue, and a split of Owner 7,000 bps, Operator 2,000 bps, a locked BaseModel royalty of 500 bps and a Contributor at 500 bps.

```
fee = floor(1000000000 * 500 / 10000) = 50000000
net = 1000000000 - 50000000 = 950000000
```

Recipient	Weight	Base units	USDC
Buyback vault (protocol fee)	500 bps of gross	5,000,000	5.00
Model owner	7,000 bps of net	66,500,000	66.50
Serving operator	2,000 bps of net	19,000,000	19.00
Base model owner (locked)	500 bps of net	4,750,000	4.75
Data contributor	500 bps of net	4,750,000	4.75
Total		100,000,000	100.00

One `distribute` transaction makes all five transfers. On a lane where the same model runs on another operator's node, the Operator share goes to that operator's payout account.

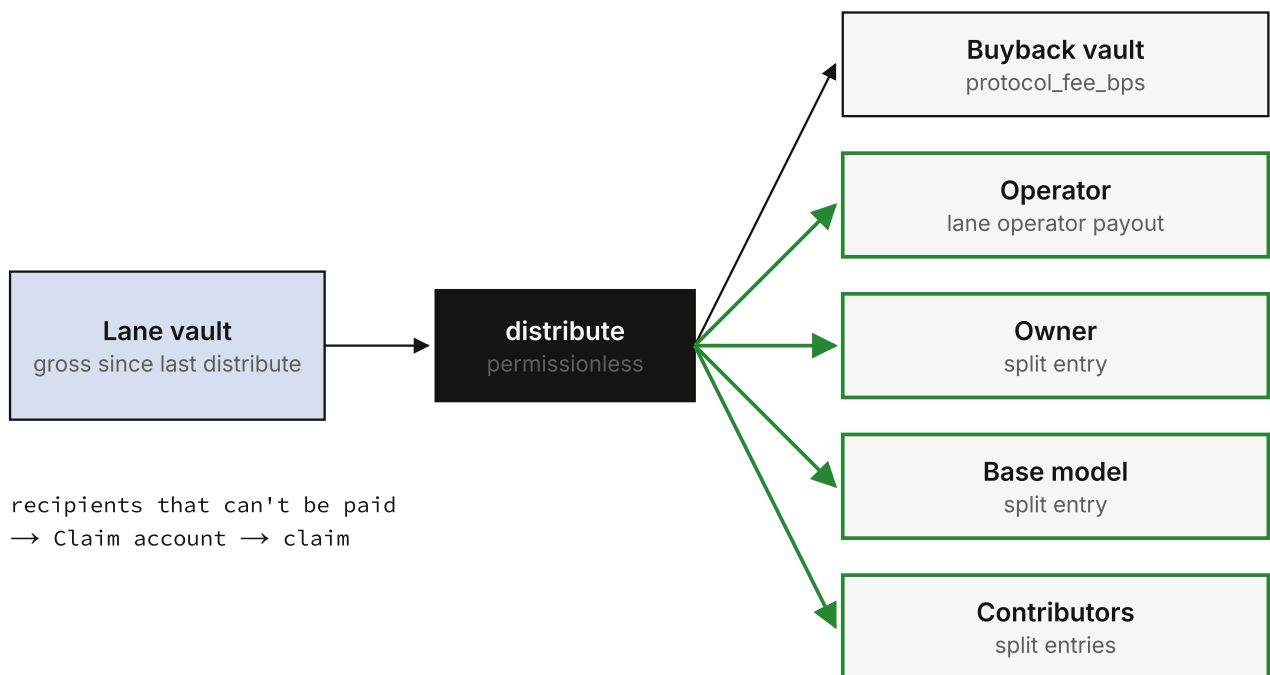


Figure: One distribute call splits a lane vault into the protocol fee, the serving operator's share and the model's other split recipients.

5. Bonds and disputes

USDC bonds per tier

Operators are admitted at one of three tiers, each with a USDC bond minimum in Config. An operator is healthy while it is not frozen and its bond meets its tier's minimum, and registering platforms, registering nodes and recording attestations all require a healthy operator. Each model adds its own minimum tier and bond, which the key broker checks before every release. Bonds and fees are in USDC, because collateral has to hold its value in the unit a slash pays out in; \$SEALROOM is never accepted as collateral.

30-day unbonding

`request_unbond` starts a 30-day wait and is refused while the operator is frozen or has an open dispute. No attestation can expire after the unbonding date, so the operator's nodes lose key eligibility before its bond can leave. `withdraw_bond` pays out the principal after the 30 days, and is refused while the operator is frozen or a dispute against it is open. A ruled slash stays reserved outside the withdrawable principal.

Admission by paid protocol fees

A model owner opens a dispute against an operator that serves one of its models. The owner's lane on that operator must have paid unconsumed protocol fees into the buyback vault of at least the dispute deposit. `open_dispute` consumes that credit, moves the USDC deposit into a dispute vault and records the evidence hash. Only fees that actually reached the buyback vault count; owner and operator shares cannot be recycled into credit. At the 500 bps launch fee, one dispute's credit takes lane revenue of 20 times the deposit. Two caps bound the arbitration queue: one per claimant and operator, and one per operator across all claimants, 8 by default.

Arbitration

The arbitration multisig signs `rule_dispute`, once per dispute. A ruling for the claimant can slash up to the operator's full bond, freeze the operator and revoke all of its attestations, and returns the deposit. A dismissal cannot slash, freeze or revoke, and forfeits the deposit to a beneficiary set in Config. The slash is reserved at the ruling; anyone can then call `settle_slash` to move it from the bond vault to the dispute vault, and the claimant collects deposit and slash with `claim_dispute`. Arbitration can also freeze or revoke an operator directly with `arbitrate_operator`, without a dispute slot. Changes to the arbitration key, deposit, forfeiture beneficiary or caps take a 48-hour proposal, and the admin and arbitration keys must differ.

6. \$SEALROOM

\$SEALROOM is Sealroom's token, and its design is deliberately small.

Supply	1,000,000,000, fixed
Launch	100% to one public launch on equal terms
Allocations	none: no presale, team, investor, airdrop or reserved allocation
Vesting	none
Protocol fee share	all of it buys \$SEALROOM on the market and burns it

Where the buyback comes from. Every paid call lands in a lane vault in USDC. Each `distribute` takes the protocol fee first, 500 bps at launch, and pays it into the buyback vault: the USDC account of a program address that no key controls. Its only way out is `buyback_swap_and_burn`, which in one instruction swaps the USDC for \$SEALROOM through Jupiter, signing for the vault itself, and burns everything the swap returns; if any step fails, nothing moves. The protocol's keeper starts it, but never holds the funds. The buyback therefore tracks paid inference directly: a lane that distributes 25 USDC at 500 bps sends 1.25 USDC into the vault, and all of it is spent on \$SEALROOM that is then burned.

What stays in USDC. Calls, bonds, dispute deposits and payouts are priced and settled in USDC, and key release depends only on the six release checks. No token, \$SEALROOM or any other, is accepted as bond collateral or used to gate a key release, so using Sealroom never requires holding \$SEALROOM.

7. For owners, operators and callers

Everyone uses the same app. Its registry pages (Models, Nodes, Operators, Disputes) read the indexed chain, and every write is a transaction the user's own wallet signs from a preview, through Wallet Standard wallets such as Phantom, Solflare or Backpack.

Role	Where	What they do
Model owner	Your models, Disputes	List a model with its manifest hash, key broker, policy, price and split. Open a lane on each serving node that meets the policy, distribute lanes, propose split changes, and open a dispute on one of its lanes.
Operator	Apply, Operate	Apply with the authority wallet and pass vetting. Deposit the bond, register servers (platform identity and GPU UEIDs, stored as hashes) and nodes, watch each node's attestation and eligibility, and unbond.
Caller	Call a model	Pick a model and a serving node, see the price from the node's 402, and sign one USDC transfer to the lane vault. The answer comes back from the enclave, and the payment shows in the vault on the model's page.

Role	Where	What they do
Anyone	Registry pages	Read operators, nodes, attestations, models, lanes and disputes, and call <code>distribute</code> on any lane.

8. Parameters

Parameter	Value	Set by
Protocol fee	500 bps of each distribution (launch parameter)	Config; <code>propose_fee</code> , <code>apply_fee</code> after 48 h
Attestation lifetime	at most 24 h (86,400 s)	program constant
Verifier committee	5 members, 3 signatures	VerifierSet (room for 7 keys)
Accepted TCB status	UpToDate	<code>record_attestation</code>
Runtime allowlist	48 h to allow, immediate revocation	<code>allow_runtime</code> , <code>revoke_runtime</code>
Governance delay	48 h for fee, arbitration and dispute settings	propose, then apply
Unbonding period	30 days (2,592,000 s)	<code>request_unbond</code>
Split delay	7 days (604,800 s)	<code>propose_split</code> , <code>apply_split</code>
Split entries	up to 8, summing to 10,000 bps, one Operator role	Model
Runtimes per model	up to 8 measurements of 48 bytes	Model
Operator allowlist per model	up to 16	Model
Operator tiers	3, each with a USDC bond minimum	Config
GPUs per platform and node	1 to 8	Platform, Node
Node endpoint, broker URL	HTTPS, up to 128 bytes	Node, Model
Open disputes per operator	8 by default	Config

9. Why Solana

x402 payments. Agent payments over x402 are concentrated on Solana. On January 20, 2026, x402 payments on Solana passed those on Base (518,400 against 505,000), and by mid-August 2026 Solana carried about 70% of monthly x402 volume and had processed more than 35 million x402 transactions. A Sealroom call is one transfer to a lane vault, the layout Solana x402 facilitators

already accept, so any agent that pays over x402 on Solana can call a Sealroom model. Metered calls use the Solana Foundation payment-channels program (CHNLxYvVA28MJP9PrFuDXccuoGXAx7jBacfLEkahyGsX), live on mainnet and audited by Cantina in July 2026, which settles actual usage against an escrowed ceiling with an Ed25519-signed voucher.

Cheap re-attestation and distribution. Every serving node re-attests at least once every 24 hours, and lanes are distributed as revenue arrives, so both transactions run constantly. Measured on the compiled program, a three-signature `record_attestation` uses 49,314 compute units in a 1,056-byte transaction (version 0 with an address lookup table), and a `distribute` paying the protocol fee and eight split recipients fits in 308 bytes. The attestation uses about 3.5% of the 1.4 million compute-unit transaction limit, at a fraction of a cent per transaction.

Native USDC and the Ed25519 precompile. Calls, bonds, dispute deposits and payouts all settle in native USDC through the SPL Token program, with no bridge in the path. Committee signatures are checked by Solana's native Ed25519 precompile: the program reads the verification instructions through the instructions sysvar and confirms a 3-of-5 quorum without running signature arithmetic itself, and the SlotHashes sysvar supplies the slot hash every attestation is bound to. Because the registry and each model's policy live on Solana, any owner, caller or auditor can read which machines are attested, which runtime they run, which operator is liable for them and which keys the broker released.