

SOVEREIGN INTELLIGENCE SYSTEMS

Huntress, Batgirl, & the Hulk

We took the whole system off the public web and put it on a private tailnet of our own devices, built on Tailscale. Public once. Tailnet forever after.

We took the whole system off the public web and put it on a private tailnet of our own devices, built on Tailscale. Public once. Tailnet forever after.

Kaleeg Hainsworth & Malaika

10 chapters · 2,407 words · 2026

<https://sovereignintelligence.systems/works/huntress-batgirl-hulk>

CONTENTS

01	The beginning	1 min
02	The problem, stated honestly	1 min
03	The inversion	1 min
04	Five things that make it different	3 min
05	The mechanics — receipts	1 min
06	The evidence	1 min
07	The commons	1 min
08	What is live today	1 min
09	The road ahead	1 min
10	References	1 min

The beginning

A man walked across Europe to hear one sentence. He had crossed a continent on foot to stand before a monk he would meet for five minutes, through a translator, and what the monk gave him in those five minutes changed the rest of his life: *the beginning is everything*.

We begin the same way he did — with a claim about where things start. Everything described in this paper began as one impertinent question: *what if the computer that knows you did not live on the internet at all?*

This paper is about a thing we built and the idea underneath it. It has taken a year to learn that they are the same idea.

The problem, stated honestly

Artificial intelligence made the most intimate things about a person — memory, taste, the private shape of a life — into *data*. And data, as we have built the world, flows to computers we do not own.

Every serious answer so far has been a variation on one move: *keep the cloud, but make the cloud polite*. Encrypt at rest. Add a compliance layer. Shrink a datacenter into a pod and put it in a cupboard, while the model's questions still go up and the answers still come back down. The topology never changes. The client reaches up; the cloud reaches back; and somewhere in between, someone else's computer holds the middle of the conversation.

The cloud is the wrong shape for a life. A house does not become yours because the landlord promised to be careful with the keys.

[FIGURE 1 — "Client to cloud": the conventional topology, drawn as a single tree reaching upward into a shared sky. Every leaf's data passes through the same trunk. Caption: *The shape everyone keeps building.*]

The inversion

Sovereign removes the cloud from the shape of the thing.

Imagine a forest. There is a mother tree, and there are the many younger trees she supports. Between them, under the soil, runs a web of roots and fungal threads — a mycorrhizal network — that carries water and signal from one tree to another without ever rising into the air. Above the canopy there is sky, but the life of the forest happens in the roots, in the soil, out of sight of anything passing overhead.

That is the architecture. The **root network** is a WireGuard overlay — a *tailnet* — connecting a person's devices to one another, device to device, encrypted end to end, invisible from the public internet above. The community node is the mother tree. The public internet shrinks to a thin canopy edge, used only where the outside world genuinely must reach in — to hand a newcomer a seed, to carry a single message up to the sky and back. Everything else happens below.

We have a phrase for this, and it is now a rule: **public once, tailnet forever after**.

The computer that knows you no longer lives on the internet. It lives on a private network you own, made of your own machines, that no stranger can reach and no datacenter routes.

[FIGURE 2 — "The forest and the canopy": the tailnet drawn as a mycorrhizal root web beneath the soil line, devices as trees, a thin public edge above. The mother tree — the community node — at the center. Caption: *Public once. Tailnet forever after.*]

Five things that make it different

Each of these is a consequence of changing the topology, and each is unusual enough to deserve its own page.

1. The network is the security boundary

On the internet, security is something you add: a password, a token, a firewall, an apology after the breach. On a tailnet, security is the default state of the wire. Our tailnet runs a policy file that denies everything first and grants only what is named — a file that is version-controlled, tested on every change, and applied the way software is applied: a pull request, a machine that checks the rules, a merge that refuses to land while a test fails.

We call the tests **deny-tests**, and they are the heart of the matter. You cannot prove a door is locked by asserting that it is locked. You prove it by writing a test that tries to open it and must fail. Our policy ships with tests that say: *a member cannot reach the broker's port; a member cannot open another member's device; the node can never push into a member's machine*. They run on every save, on every pull request, on every apply. Isolation is a failing test that stays green.

[FIGURE 3 — "The constitution": the policy file as a written constitution beside the tailnet, with red X's marking the deny-tests that must fail. Caption: *Security as a failing test*.]

2. Identity is the wire

In the old shape, you prove who you are by remembering something, and the system

stores what you remembered. On the tailnet, the connection itself carries who you are. Tailscale proves the identity of a connecting device with cryptography before a single request is read. When your own phone reaches your own node, the network already knows it is you — no second sign-in, no password to lose, no credential store to leak.

We treat that identity as the login. A person's device is their presence. And because the proof is cryptographic and lives in the wire, there is nothing extra for an attacker to phish.

3. Sharing is scoped to the thing

This is the one that changes what a community can be.

Under the cloud, sharing meant giving someone access to the machine, and then building ever more careful walls inside it. Here, you share an *item*. You mark one book as restricted and grant it to one person. For everyone else the book is gone from the shelf, from search, from the AI's answers, from the knowledge graph, from the library's constellations. It does not exist for anyone who was not granted it. Revoke the grant, and it stops existing again, immediately, everywhere at once.

Locked rooms can be picked. A room that was never built cannot.

[FIGURE 4 — "The book that isn't there": two members' views of the same library side by side — one sees the book, the other sees a gap where the book would be. Caption: *Restricted means absent.*]

4. The machine becomes unaddressable

Members never address the *machine*. They address a **named capability** — `svc:community-web`, `svc:community-models` — a name that floats above the physical host and can be drained, moved, or failed over without anyone's configuration

changing. The host itself has no address a member can aim at.

While building this we found that the hiding runs deeper than we planned: **a Tailscale service's name does not even resolve for a device that holds no grant to it.** The ungranted never learn the capability's address. DNS is a second boundary. A machine you cannot name is hard to attack.

5. Public exposure is a decision

Nothing is public unless someone deliberately says so — and the decision is reversible, tag-scoped, and visible in the policy. Our community node once had a public door; we closed it, and proved it closed from outside the network. What remains of the public face is a single path-scoped slot for a Telegram message — a mail-slot. If the decision ever stops being worth it, it is one rule away from being undone.

The mechanics — receipts

A few decisions deserve their own lines, because a reader should be able to take them and use them.

Three layers, none sufficient alone. Network perimeter (default-deny grants), identity (the wire), and application grants (per-item tables) — each covers what the other two cannot. A grant at the network layer cannot tell one book from another; a grant at the application layer cannot stop a device from reaching the host. Together they hold.

Union semantics. Tailscale's grants can only *add* access. This looks like a weakness until it turns the security model into arithmetic: isolation comes from the *absence* of grants, and absence is exactly what the deny-tests prove. You cannot accidentally carve a hole; you can only fail to close one, and the tests catch that.

The allow-all footgun. A fresh tailnet, before any policy file exists, allows everything. We ship a policy file at first boot. An appliance handed to a non-technical person must never spend one second in the default state.

The pull doctrine. Members pull their credentials from the node; the node never holds a way into a member's machine. Enforced in the policy by a deny-test. The direction credentials travel decides the shape of the network.

Capability tokens for things that travel. A book can be handed to a phone or a Telegram chat by a link that is hashed at rest, expires by default, and is revocable in one stroke. The token names one item and nothing else.

The lineage. The per-item grant did not appear from nothing. It is the same idea as a legacy `UserBookEntitlement` table that once carried a `LOANED_TO_COMMUNITY`

status, reborn on rails that actually enforce it. Sovereignty is old ideas finally given teeth.

The evidence

We did not invent the claim that this is possible. We measured it against the field, and the field agrees.

It is viable. OpenJarvis (Stanford, 2026) showed that a decomposed local stack — intelligence, engine, agents, tools, memory — can reach within 3.2 points of a frontier cloud model on personal-AI benchmarks, at roughly 800× lower marginal cost, with inference on the device. Minions (ICML 2025) showed the raw context can stay local while a frontier model only orchestrates, at ~98% of the quality for 5–30× less. The gap between "local" and "cloud" is a systems problem, and it closes.

It is necessary. PrivacyBench (2025) measured retrieval-augmented assistants leaking a user's secrets in 16% of conversations by default. Opal (2026) showed that even the *pattern of retrieval* leaks. Scoping is the distance between a home and a sieve.

It is open ground. Multi-tenant *community* machines — friends and family sharing one computer and one AI — are almost unstudied; the literature is all enterprise Kubernetes. Cloudflare's own Mesh announcement (April 2026) admits that per-agent identity at the network layer remains unsolved industry-wide. The UK's ARIA is funding millions to verify the control plane of a WireGuard overlay. We are standing in a space the field has named and left empty.

[FIGURE 5 — "The white space": a map of what is studied (enterprise tenancy, cloud privacy) and what is empty (community tenancy on an overlay). Sovereign's mark in the empty space.]

The commons

A shared machine is a commons, and commons have known physics. We borrowed from the people who learned this before the cloud existed.

Guifi.net — the world's largest community network — runs on Elinor Ostrom's design principles: clear boundaries, monitoring, graduated sanctions, and a conflict-resolution procedure for the day someone sees something they should not have. NYC Mesh isolates its members from one another by default: *members connect to the network, to each other they are strangers*. Both lessons are now ours. A community node is a shared uplink and a shared library, never a shared hard drive; and "Alice saw Bob's corpus" is a governance event.

What is live today

Most of this is already running.

The tailnet is **deny-by-default**, governed by a version-controlled policy, applied through a CI pipeline that tests on every change and blocks a bad apply. The community node carries a server identity.

The node's **public door is closed**, proven from outside the network. The only public surface left is the Telegram mail-slot.

The **per-item grant system is complete**: one book, one person, enforced on read, images, export, edit, shelf, search, the AI's tool lane, the graph, and the collections — hidden everywhere, in every case, for everyone without the grant.

Capabilities have names — `svc:community-web`, `svc:community-models` — and members address the name, never the machine. The machine's own address is already invisible to them.

A member's device is their identity, and a **share link** can carry a book to a phone without a session.

Three migrations are applied to the substrate; the whole thing is pinned by more than sixteen hundred tests; and the settings panel shows a member their place in the forest.

We caught a real bug on the way — an off-by-one in the search floor — because the discipline here is to never trust the last step, to smoke-test the live thing. That, as much as any feature, is the point.

The road ahead

Three things remain before this paper can stop being a draft.

The last cloud dependency. The node's data plane still leans on a cloud project for identity and membership. The next build brings that layer home — a fully local stack, so a node depends on nothing it does not own.

The grant panel. The machinery works through the API today; the human-facing button ships when its lane lands. A grant should be a click.

Federation. Sibling nodes are modeled but not yet wired — the point where one forest holds hands with another without merging their soil. That is the next horizon, and it is the one that turns "a community" into "communities."

When the last dependency comes home and the first federation link is made, we will write this paper again — and this time it will read like a map of ground already walked.

[FIGURE 6 — "The grove": many mother trees, each with its own root network, connected by a single shared root between them. Caption: *The next horizon.*]

References

NIST SP 800-207, *Zero Trust Architecture* (2020). <https://csrc.nist.gov/pubs/sp/800/207/final>

NIST SP 800-207A, *A Zero Trust Architecture Model for Access Control in Cloud-Native Applications* (2023). <https://csrc.nist.gov/pubs/sp/800/207/a/final>

NIST SP 1800-35, *Implementing a Zero Trust Architecture* (2025). <https://csrc.nist.gov/pubs/sp/1800/35/final>

Cloudflare, *Mesh* (2026-04-14). <https://blog.cloudflare.com/mesh/>

Tailscale, *Grants are generally available* (2025-05-29). <https://tailscale.com/blog/grants-ga>

Tailscale, *Tailscale Services (beta)* (2025-10-28). <https://tailscale.com/blog/services-beta>

Tailscale, *Tailnet Lock* white paper. <https://tailscale.com/docs/concepts/tailnet-lock-whitepaper>

Tailscale, *Five smart ways to share Tailscale with friends and family* (2025-09-15). <https://tailscale.com/blog/tailscale-sharing-friends-family>

Tailscale, *Application capabilities* (2025-10-30). <https://tailscale.com/blog/app-capabilities>

Tailscale, *Aperture* (GA 2026-08-26). <https://tailscale.com/blog/aperture-ga>

Saad-Falcon, Narayan, Ré, Mirhoseini et al., *OpenJarvis: Personal AI, On Personal Devices* (2026). <https://arxiv.org/abs/2605.17172>

Biderman, Narayan, Ré, *Minions: Fast, Private Local-to-Cloud LLM Chat* (ICML 2025). <https://hazyresearch.stanford.edu/blog/2025-05-12-security>

Mukhopadhyay et al., *PrivacyBench* (2025). <https://arxiv.org/abs/2512.24848>

Opal: Private Memory for Personal AI (2026). <https://arxiv.org/abs/2604.02522>

UK ARIA, *Safeguarded AI: Cybersecurity solicitation* (2026). https://aria.org.uk/media/1padxpaf/safeguarded-ai_cybersecurity_solicitation.pdf

Kleppmann, Wiggins, van Hardenberg et al., *Local-First Software: You Own Your Data, in Spite of the Cloud* (2019). <https://www.inkandswitch.com/essay/local-first/>

Guifi.net — Ostrom's commons principles applied to a community network. <https://guifi.net>

NYC Mesh — isolation by default. <https://wiki.nycmesh.net>