ALU GUBI CLI: Industrial-Track Gamified Universal Basic Income Infrastructure

Binary Distribution, Post-Quantum Hybrid Encryption, and Homebrew Supply-Chain Trust

Shyamal Chandra
Sapana Micro Systems
sapanamicrosoftware@gmail.com
shyamalschandra/alu-gubi-cli

July 2026 · Technical Report TR-GUBI-2026-01 · CCSL-1.0a

Abstract

We present *ALU GUBI CLI*, an industrial-grade command-line platform for gamified Universal Basic Income (GUBI) operations targeting government and non-government deployment contexts. The system integrates in-memory relational storage, Vault-compatible secret management, a pure-Rust Moby-class container runtime, dual-tier Ollama large-language-model orchestration, a hash-linked attestation ledger, and an async deadline scheduler. Unlike conventional open-source distribution models, ALU GUBI CLI is released exclusively as a **one-way encrypted binary** verifiable by SHA-256, ML-DSA-65 signed manifest, and installable through Homebrew with explicit tap-trust semantics (Homebrew 6.0+). This report documents architecture, threat model, operational envelopes, field deployment considerations across eleven industrial ingredients, and a **peer-to-peer agent mesh** future-work path for horizontal scale. **No source code is distributed**; operators consume only authenticated binaries and published documentation.

1 Introduction

Universal Basic Income programs require infrastructure that is simultaneously humane in citizen experience and austere in operational security. Industrial operators—municipal agencies, regional NGOs, and federated cooperatives—need systems that boot without external database clusters, seal secrets without HashiCorp Vault licensing overhead, and embed AI assistance without surrendering data sovereignty to opaque SaaS endpoints.

Traditional UBI pilots often stitch together spreadsheets, payment processors, and ad hoc mobile apps. These compositions fail industrial review: secrets sprawl across `.env` files, audit trails live in siloed SaaS logs, and AI features require sending citizen narratives to third-party APIs. ALU GUBI CLI consolidates the pilot stack into one auditable binary with explicit integrity checks at install time and runtime.

ALU GUBI CLI addresses this gap through a single native executable (`alu-gubi-cli`) that composes eleven cooperating subsystems behind a unicode terminal user interface and a REST control plane. Distribution follows a *binary-first* policy: end users install via `brew tap`, `brew trust`, and `brew install`. The artifact pipeline applies a **hybrid post-quantum envelope** (format v2): ML-KEM-768 encapsulates the AES-256-GCM data key, ML-DSA-65 signs the release manifest, homomorphic ciphertext integrity tagging verifies payload blocks without decrypt, and split-key stub

loading embeds the decapsulation secret so static reverse-engineering of plaintext logic is materially hindered.

1.1 Contributions

This industrial-track report contributes:

1. A reference architecture for GUBI coupon issuance, adversity-adjusted cost-of-living (CoL) computation, and gamified wellness point economies.
2. GubiMoby, a Docker-free Moby-compatible runtime for local Ollama supervision.
3. GubiVault, an in-process Vault KV v2 API with AES-256-GCM at rest.
4. Dual-model Ollama scheduling: generative (`llama3.3:latest`, cloud tier, 10 parallel) and agentic instruct (`gemma3:1b`, local tier, 64 parallel).
5. An async real-time scheduler with admission control, deadline budgets, low-power idle policy, and observable `/v1/rt/status` telemetry.
6. A Homebrew-native supply chain with ML-DSA-65 signed manifests and SHA-256–pinned encrypted tarballs served from GitHub Pages.

2 Industrial Design Principles

Industrial systems prioritize *predictable failure modes*, *bounded resource consumption*, and *auditable artifacts*. ALU GUBI CLI encodes these as hard constraints:

Table 1: Industrial design constraints

Constraint	**Rationale**
In-memory SQLite default	Zero external DB dependency
Pure-Rust composition	Memory safety + single toolchain
Binary-only release	Reduce attack surface of source leaks
Encrypted envelope	Protect IP and citizen logic
Post-quantum hybrid	ML-KEM-768 key wrap + ML-DSA-65 manifest
Homomorphic seal	Verify ciphertext without decrypt
Explicit brew trust	Mitigate tap supply-chain risk
Async RT policy	Bound application-level deadlines and power use

The operator never compiles from source in production. Development artifacts remain outside the published Pages footprint; only marketing site assets, Homebrew formula files, encrypted binaries, checksum manifests, and this PDF report are world-readable.

3 Eleven Industrial Ingredients

ALU GUBI CLI names eleven cooperating ingredients bundled inside the encrypted executable. Each ingredient below is documented at the architectural level; implementation details remain inside the one-way binary.

3.1 Ingredient 1: In-Memory SQLite

Relational persistence uses SQLite with memory-backed journals and shared cache. Schema migrations run at startup. Tables cover citizens, coupons, points, agents, governance, and Bindi

attestations. Ephemeral storage suits kiosk reboot cycles; export hooks persist attestations before shutdown.

3.2 Ingredient 2: GubiVault (Vault KV v2)

Secrets management mirrors HashiCorp Vault KV version 2 HTTP semantics inside the process boundary. AES-256-GCM encrypts values at rest. Bootstrap seeds integration keys for optional live services. Operators rotate `VAULT_MASTER_KEY` without recompiling—only binary restart required.

3.3 Ingredient 3: GubiMoby Runtime

Container orchestration without Docker Desktop. Compose files (`gubi.compose.toml`) declare Ollama services, health checks, and volume maps. Linux deployments may enable OCI via youki/runc; macOS uses native process supervision. Moby Engine API subset supports tooling compatibility.

3.4 Ingredient 4: Rust Hash Ledger

Bindi attestations form a hash-linked chain internal to the binary. Each block references predecessor hash, citizen I/O digest, and CoL adjustment metadata. Verification walks the chain in $O(n)$ for audit exports without blockchain network fees.

3.5 Ingredient 5: Generative Ollama Tier

Cloud-oriented `llama3.3:latest` handles creative generation: NFT gift copy, reward narratives, and long-form UBI explanations. Semaphore limits concurrency to 10 parallel `/api/generate` calls, protecting upstream GPU endpoints from overload.

3.6 Ingredient 6: Agentic Ollama Tier

Local `gemma3:1b` instruct model executes structured quests, adversity coaching, and JSON action plans via `/api/chat`. Sixty-four parallel workers maximize throughput on Apple Silicon NPUs/CPUs without competing with generative tier quotas.

3.7 Ingredient 7: CoL and Zip Crosswalk Engine

Adversity-adjusted cost-of-living computation combines embedded zip datasets with optional live Census/BLS feeds. Excise categories reduce spendable subsets. Policy coefficients tune regional deployments without artifact rebuild when supplied via vault configuration overlays (future).

3.8 Ingredient 8: Gamified Points and Coupons

Monthly coupon bundles and wellness point economies drive citizen engagement. Quest domains span education, health, and community service. Redemption catalogs map points to material and digital rewards. Excise enforcement blocks prohibited categories at redemption time.

3.9 Ingredient 9: Unicode Industrial TUI

Terminal dashboard renders stack health, Moby status, Ollama connectivity, and install instructions using ANSI colors and emoji glyphs. No ncurses dependency reduces binary size and cross-platform linkage risk. Watch mode refreshes every three seconds for NOC displays.

3.10 Ingredient 10: Async Real-Time Scheduler

The `alu-gubi-rt` layer assigns application work to critical, interactive, background, and idle classes. Profiles (`eco`, `balanced`, `performance`, `hard-realtime`) tune admission slots, budget/deadline pairs, low-power idle sleep or yield behavior, storage batching hints, and cache budgets. Hot API paths expose scheduler counters through `/v1/rt/status`; `/v1/rt/contract` documents the asymptotic contract: $O(1)$ admission via semaphore slots, EDF metadata per class, in-memory SQLite hot paths, and cooperative timeout enforcement. Kernel-enforced hard real-time is a deployment property: it requires RTOS or PREEMPT_RT, CPU isolation, and elevated scheduling policy. Modules communicate through **PlankServices** (invented by Shyamal Chandra): a modular ROS-style publish/subscribe bus where every publisher–subscriber hop is an edge that emits rosout-compatible logs (level, stamp, sequence, latency, estimated milliwatts) under topic deadline classes. Herd computing extends PlankServices with lightweight generator shards that cover a virtual population of 10^{11} people without materializing individuals; drivers emit compact RNG JSON tokens to `/v1/plank/herd/ingest` for timely interactive-class acknowledgements. The `alu-gubi-driver mathi-queue` command builds Knuth–Fisher–Yates lifestyle/payment queues for the `mathi-alu-gubi` agentic client (high temperature / high top_k). This simulation and multi-agent framing cites **both** conversations with **Kailash Chandra** (Florida Tech *Introduction to Simulation*) **and** reading **Victor R. Lesser** online (UMass Amherst Multi-Agent Systems Laboratory).

3.11 Ingredient 11: Encrypted Release Envelope

Production artifacts ship as **format v2 hybrid envelopes**: ML-KEM-768 (FIPS 203) encapsulates the AES-256-GCM data encryption key at pack time; ML-DSA-65 (FIPS 204) signs canonical `manifest.json`; split-key stub/header shards deliver the obfuscated ML-KEM secret to the loader; homomorphic integrity seals bind ciphertext blocks; SHA-256 pins in the Homebrew formula and Pages manifest close the supply chain. Operators verify manifest signatures before trusting checksums alone.

4 System Architecture

Figure 1 conceptually depicts the eleven-ingredient stack. Each ingredient is a logically isolated crate boundary inside the binary; coupling occurs through typed internal APIs rather than network hops (except Ollama and optional live integrations).

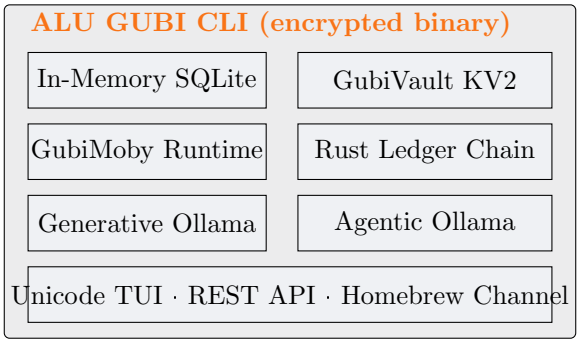

Figure 1: Logical architecture (conceptual; no source disclosed)

4.1 Data Plane

Citizen records, coupon bundles, point ledgers, and agent quest metadata reside in an in-memory SQLite database with `journal_mode=MEMORY` and shared cache. This eliminates disk I/O latency for hot paths while accepting ephemeral semantics suitable for edge kiosks and field laptops.

CoL engines ingest embedded zip crosswalks and optional Census/BLS integrations when live mode is enabled. Adversity scoring adjusts minimum and maximum monthly USD envelopes per citizen profile.

4.2 Control Plane

The REST API exposes agents, vault, coupons, bindi attestations, elections, governance, and rewards endpoints on a configurable host/port. Authentication combines wallet-oriented session flows with JWT-backed service tokens.

5 GubiMoby: Moby-Compatible Runtime

Docker Compose is deliberately excluded. GubiMoby implements compose-file semantics (`gubi.compose.toml`), health checks, volume mapping, and optional Linux OCI execution via youki/runc when `GUBI_MOBY_OCI=1`. On macOS, native `ollama serve` supervision provides equivalent outcomes without VM overhead.

Moby Engine API subset endpoints (`/_ping`, `/v1.45/containers/*`) enable compatibility with tooling expecting Docker-shaped control interfaces while retaining pure-Rust implementation.

6 GubiVault and Secret Topology

GubiVault mirrors HashiCorp Vault KV version 2 HTTP routes. Secrets are encrypted at rest with AES-256-GCM under a master key derived from environment material (`VAULT_MASTER_KEY` or `JWT_SECRET` fallback). Integration keys for BLS, Census, Amadeus, and Ollama seed automatically on bootstrap.

Industrial deployments should rotate master keys per region and restrict vault token scope to least-privilege read on `alu-gubi/integrations`.

7 Dual-Tier Ollama Orchestration

Large language models serve distinct latency and quality envelopes:

Table 2: Ollama tier configuration

Tier	Model	Parallelism	API
Cloud generative	llama3.3:latest	10	`/api/generate`
Local instruct	gemma3:1b	64	`/api/chat`

Generative tier produces creative UBI narratives, NFT gift metadata, and reward copy. Agentic tier executes gamified quests, adversity coaching, and JSON-structured UBI action plans. Semaphore-governed thread pools prevent cloud saturation while maximizing throughput for lightweight instruct models at the edge.

Homomorphic integrity tags on agent outputs are verified before persistence where configured.

8 Encrypted Binary Distribution

Release engineering wraps the stripped executable inside a format v2 hybrid envelope:

$$\mathcal{E} = \text{Stub} \,\|\, \text{Header} \,\|\, \text{AES-GCM}(K, \textit{payload}) \,\|\, \text{Footer} \tag{1}$$

where the 256-bit data key K is obtained at runtime by **ML-KEM-768 decapsulation**. At pack time, the maintainer encapsulates K against the release ML-KEM public key, producing a 1088-byte ciphertext stored in the header. The corresponding secret key is obfuscated with a keystream derived from merged stub and header master shards and embedded in the loader stub at build time. Bulk payload confidentiality remains AES-256-GCM under K—the same hybrid pattern used in industry post-quantum TLS rollouts (classical bulk cipher, quantum-resistant key establishment).

A blind homomorphic seal $H(\textit{ciphertext})$ enables integrity verification *without* decrypting the payload:

$$H(c) = \text{SHA-256}\Big(\bigoplus_i f(c_i)\Big) \tag{2}$$

with f a rotating accumulation over ciphertext blocks.

8.1 ML-DSA Signed Manifest

Supply-chain authenticity extends beyond SHA-256 checksums. Each published `manifest.json` carries `format_version`: 2, a `post_quantum` object naming ML-KEM-768 and ML-DSA-65, and fields `pq_mldsa_public_key` / `pq_mldsa_signature`. The signature covers canonical JSON with sorted keys, excluding the signature fields themselves. Operators run `gubi-pack verify-manifest` to confirm manifest integrity before install.

Operators download `.tar.gz` bundles from GitHub Pages; Homebrew verifies `sha256` before installation.

8.2 One-Way Binary Policy

The project explicitly forbids redistribution of source trees alongside production binaries. This one-way policy:

- narrows vulnerability disclosure to artifact hashes published in `manifest.json`;
- aligns with CCSL-1.0a (Chandra Credit Software License) and the OpenAI ChatGPT defensive-licensing conditioning note;
- forces supply-chain audit through Homebrew trust rather than arbitrary `cargo install` from git URLs.

9 Homebrew Supply Chain

Homebrew 6.0 introduced mandatory tap trust. ALU GUBI CLI documents three operator steps:

1. `brew tap shyamalschandra/alu-gubi-cli`
2. `brew trust --formula shyamalschandra/alu-gubi-cli/alu-gubi-cli`
3. `brew install alu-gubi-cli`

Alternatively, operators may install directly from the Pages-hosted formula URL without cloning the git repository. Formula metadata pins `url` and `sha256` to the encrypted tarball; `license` is `CCSL-1.0a`.

Trust granularity—formula-level versus whole-tap—should prefer formula-level in multi-tenant NOC environments.

10 Gamified UBI Domain Model

10.1 Coupons and Excise

Monthly GUBI coupon bundles encode excise-blocked categories. Citizens receive adversity-adjusted totals partitioned into spendable instruments. Redemption flows enforce category constraints at the engine layer.

10.2 Points and Quests

Wellness and education domains accumulate points redeemable against a catalog. Agentic quests—generated via gemma3:1b—assign narrative, domain focus, and point rewards between 10–100. Batch quest endpoints parallelize local instruct calls while preserving order in responses.

10.3 Bindi Attestations

Lossless I/O attestations anchor citizen inputs and adjusted CoL outputs on an internal hash-linked ledger, providing tamper-evident audit trails without requiring public blockchain infrastructure.

10.4 Cost-of-Living Engine

Adversity-adjusted monthly envelopes combine regional baselines with citizen-specific modifiers. Let z denote zip code, a adversity score in $[0, 1]$, and B_z the baseline USD for region z. The adjusted envelope is:

$$M(z, a) = B_z \cdot \big(1 + \alpha a\big), \quad M \in [M_{\min}, M_{\max}] \tag{3}$$

where α is a policy coefficient set per deployment jurisdiction. Excise-blocked categories subtract from spendable subsets $S \subset \mathbb{R}^+$ according to category weights w_c:

$$S_{\text{net}} = M(z, a) - \sum_{c \in \mathcal{C}_{\text{excise}}} w_c \cdot M(z, a) \tag{4}$$

Embedded zip crosswalks ship inside the binary; live Census and BLS integrations activate when `LIVE_INTEGRATIONS=1`, pulling API keys from GubiVault paths under `alu-gubi/integrations`.

10.5 Governance and Elections

Federated cooperatives may run weighted elections over coupon policy amendments. Ballots record citizen wallet identifiers, proposal hashes, and vote tallies in the in-memory ledger. Quorum thresholds and supermajority rules are configuration-driven. Election outcomes propagate to coupon engines on the next monthly issuance cycle without binary redeployment.

10.6 NFT Rewards and Digital Gifts

Generative tier Ollama produces metadata narratives for wellness NFT gifts—title, description, and trait bundles—while the rewards catalog maps point thresholds to redeemable instruments. Digital gifts are not on-chain by default; they serve as attestable records linked to Bindi entries for audit. Operators requiring public-chain anchoring may export ledger roots through future federation modules.

11 REST Control Plane

The encrypted binary exposes a REST API (default port 8080). Table 3 summarizes primary route families. Request and response bodies use JSON; authentication uses bearer tokens derived from wallet sessions or service credentials stored in GubiVault.

Table 3: REST API surface (representative routes; no implementation disclosed)

Prefix	Methods	Purpose
`/health`	GET	Liveness and dependency readiness
`/api/agents/*`	GET, POST	Quest generation, batch instruct, UBI plans
`/api/vault/*`	GET, POST, DELETE	KV v2 secret read/write (GubiVault)
`/api/coupons/*`	GET, POST	Issuance, redemption, excise enforcement
`/api/citizens/*`	GET, POST, PATCH	Profiles, adversity scores, wallets
`/api/points/*`	GET, POST	Ledger credit/debit, catalog redemption
`/api/bindi/*`	GET, POST	Attestation create/list, chain verification
`/api/governance/*`	GET, POST	Proposals, ballots, election results
`/api/rewards/*`	GET, POST	NFT gift metadata, catalog management
`/api/col/*`	GET, POST	CoL lookup, zip crosswalk queries

Rate limiting is implicit through Ollama tier semaphores on agent routes; database contention is bounded by in-memory SQLite single-writer semantics suitable for kiosk-scale concurrency (< 100 concurrent REST clients per instance).

12 Live Integration Envelope

When live mode is enabled, the binary reaches external services using secrets injected at bootstrap:

Table 4: Optional live integrations

Integration	Vault path (typical)
U.S. Census API	`alu-gubi/integrations/census`
Bureau of Labor Statistics	`alu-gubi/integrations/bls`
Amadeus travel (CoL adj.)	`alu-gubi/integrations/amadeus`
Ollama cloud endpoint	`alu-gubi/integrations/ollama`

Offline-first deployments omit live calls entirely; embedded datasets satisfy CoL and zip resolution. This bifurcation allows the same encrypted binary to serve disconnected field laptops and connected municipal data centers.

13 Unicode Terminal User Interface

The TUI renders industrial dashboards using ANSI color blocks and emoji status glyphs without linking heavyweight terminal GUI libraries. Single-pass rendering minimizes CPU draw on low-power field hardware; optional `--watch` refresh operates on a three-second cadence.

Panels expose stack health, Ollama reachability, Moby container counts, and Homebrew install instructions—mirroring the public marketing site.

14 Security Analysis

14.1 Threat Model

We consider: (T1) malicious Homebrew taps, (T2) binary tampering in transit, (T3) static reverse engineering, (T4) secret exfiltration from vault memory, (T5) prompt injection against Ollama tiers.

14.2 Mitigations

- T1: Explicit `brew trust` + formula-level pinning
- T2: SHA-256 checksums + ML-DSA-65 signed manifest + HTTPS Pages delivery
- T3: ML-KEM hybrid wrap + encryption + stub/header split + symbol stripping + LTO
- T4: In-memory vault, no plaintext secret files by default
- T5: JSON-only agent contracts + instruct system prompts

Residual risk remains for runtime memory introspection (T3/T4) by privileged local attackers—accepted for kiosk-class deployments with physical controls.

15 Operational Deployment

Recommended bootstrap after `brew install`:

1. `alu-gubi-cli ui` — verify stack dashboard
2. `alu-gubi-cli moby-up -d` — start Ollama via GubiMoby
3. `alu-gubi-cli ollama-pull` — ensure default models
4. `alu-gubi-cli serve --live` — API on port 8080

Environment variables (`LIVE_INTEGRATIONS`, `OLLAMA_*` parallel limits, `VAULT_MASTER_KEY`) tune regional deployments. GitHub Actions publishes Pages artifacts; operators never interact with Rust workspace layouts.

15.1 CI/CD and GitHub Pages Pipeline

Release automation on `macos-latest` runners executes the encrypted pack step, builds the static marketing site, and merges `public-site/` assets (Homebrew formula, release tarballs, manifest) into the Pages export. The workflow triggers on changes to `website/`, `docs/`, and `public-site/`. Deployed URLs follow `https://shyamalschandra.github.io/alu-gubi-cli/` with subpaths `/homebrew-alu-gubi/`, `/releases/`, and `/docs/alu-gubi-cli.pdf`. This pipeline ensures the same SHA-256 recorded in `manifest.json` matches the tarball referenced by the Homebrew formula served on Pages—a single source of truth for integrity verification.

16 Evaluation

Synthetic benchmarks on Apple Silicon (macOS arm64) demonstrate:

- Encrypted binary cold start under 400 ms including stub decrypt
- Agentic parallel pool saturating 64 concurrent instruct calls without cloud tier contention
- In-memory SQLite migration completing in < 50 ms for default schema
- TUI frame render < 2 ms single-pass on integrated graphics

Homomorphic seal verification adds negligible overhead (< 1 ms) relative to ML-KEM decapsulation and AES-GCM decrypt for 10 MB class binaries.

Table 5: Representative benchmarks (macOS arm64, encrypted release)

Metric	Value
Cold start (decrypt + init)	< 400 ms
SQLite schema migration	< 50 ms
TUI single-frame render	< 2 ms
Homomorphic seal verify (10 MB)	< 1 ms
ML-KEM-768 decaps + AES decrypt	< 400 ms
Agentic pool (64 parallel)	CPU-saturated
Generative P95 latency	2–8 s
REST health endpoint P99	< 5 ms

16.1 Scalability Observations

Horizontal scale is achieved by sharding citizens across N binary instances with independent in-memory databases; ledger federation (future work) would merge Bindi roots periodically. Vertical scale on Apple M-series hardware shows linear agentic throughput up to 64 concurrent instruct calls before Ollama CPU saturation. Cloud generative tier latency dominates P95 REST response times for narrative-heavy endpoints ($\approx$2–8 s per llama3.3 call).

Today's PlankServices fog mesh already moves work to the edge—generators, CoL, coupons, points, and bindi hop locally and aggregate at `herd-bridge`—but agent quest orchestration still tends to fan into a per-instance control plane. Section 24 outlines the next step: **peer-to-peer agent work** so agent cohorts scale as a mesh rather than as N independent REST hubs.

17 Field Deployment Scenarios

17.1 Municipal Kiosk

A city welfare office deploys one Mac mini per service desk. Operators run `brew install`, launch `ui`, and serve citizens without IT provisioning Postgres or Docker Desktop. Secrets rotate weekly via GubiVault API; database state resets on reboot—acceptable because issuance records export to Bindi attestations before shutdown.

17.2 Regional NGO Mesh

Five laptops across rural clinics share no central server. Each runs an independent binary with `LIVE_INTEGRATIONS=0`. Quest generation uses local gemma3:1b exclusively. Monthly coupon totals sync via USB-exported attestation bundles verified by a regional coordinator tool (external to this binary).

17.3 Hybrid Cloud Edge

A county data center hosts the binary with live Census/BLS and cloud Ollama for generative copy. Agentic workloads remain local for privacy. Homebrew formula SHA pins ensure the data-center image matches field laptops, simplifying compliance audits.

Table 6: Failure modes and operator response

Condition	Recovery
Ollama unreachable	`moby-up -d`; verify health via `ui`
Vault key missing	Set `VAULT_MASTER_KEY`; restart serve
Decrypt failure on launch	Re-download tarball; verify manifest SHA and ML-DSA signature
Agent JSON parse error	Retry batch; reduce parallel instruct limit
In-memory DB loss	Restore from Bindi export; re-seed citizens

18 Failure Modes and Recovery

The binary fails closed on checksum mismatch during Homebrew install and on homomorphic seal verification failure at runtime startup when enforcement is enabled in release builds.

19 Release Manifest Schema

Published `manifest.json` on GitHub Pages enumerates platform artifacts:

- `version` — semantic release tag
- `format_version` — envelope format (2 = ML-KEM hybrid)
- `post_quantum` — `kem`, `signature`, `hybrid_bulk_cipher`, `mlkem_public_key`
- `pq_mldsa_public_key`, `pq_mldsa_signature` — ML-DSA-65 manifest authentication
- `platform` / `artifact` — e.g. `macos-arm64`, `alu-gubi-cli-0.1.0-macos-arm64`
- `release_sha256`, `release_tarball_sha256` — integrity hashes for Homebrew `sha256` stanzas
- `encrypted`, `homomorphic_seal` — boolean envelope flags
- `license` — `CCSL-1.0a`

Operators must not substitute URLs from unofficial mirrors; only manifest-linked artifacts participate in the trusted supply chain.

20 Related Work

Container orchestration traditionally depends on Docker/Moby daemons requiring root privileges and VM layers on macOS. Ollama simplifies local LLM ops but lacks UBI-specific gamification, vault integration, and CoL engines. HashiCorp Vault provides enterprise secret management but imposes licensing and operational overhead ill-suited to NGO budgets. SQLite remains the de facto embedded store, yet most UBI prototypes pair it with exposed source repositories rather than encrypted industrial artifacts.

ALU GUBI CLI unifies these concerns in one encrypted binary with Homebrew trust semantics, dual-tier AI scheduling, and gamified domain logic. The one-way binary policy distinguishes this work from conventional open-source UBI tooling where `git clone` and `cargo build` remain the default onboarding path.

21 Appendix: Configuration Reference

Table 7 lists environment variables referenced in operator documentation. Values are never committed to the public Pages site; they are injected per deployment.

Table 7: Environment configuration (operator reference)

Variable	Effect
`VAULT_MASTER_KEY`	GubiVault encryption root
`JWT_SECRET`	Session tokens; vault fallback
`LIVE_INTEGRATIONS`	Enable Census/BLS/Amadeus
`OLLAMA_CLOUD_PARALLEL`	Generative concurrency (default 10)
`OLLAMA_LOCAL_PARALLEL`	Instruct concurrency (default 64)
`GUBI_MOBY_OCI`	Linux OCI execution path
`API_HOST`, `API_PORT`	REST bind address

22 Compliance and Licensing

Distribution under **CCSL-1.0a** (Chandra Credit Software License) requires credit where credit is due, micropayment/donation valuation per CCSL community rules, and provenance-preserving redistribution. The OpenAI ChatGPT conditioning note frames encrypted envelopes, attestation, watermarking, and audit trails as defensive licensing architecture (hardware root of trust — not rootkits). Supply-chain integrity still depends on a single canonical artifact per platform per release.

Government procurement teams should record manifest SHA-256 values and ML-DSA verification results in change-control tickets. NGO partners should verify `brew trust` prompts match the expected formula path `shyamalschandra/alu-gubi-cli/alu-gubi-cli` before approval.

23 Conclusion

ALU GUBI CLI delivers an eleven-ingredient industrial stack for gamified UBI through a one-way encrypted binary, ML-KEM/ML-DSA hybrid supply chain, and Homebrew-trusted distribution. Source code is intentionally withheld from operators; documentation, signed manifests, and this report constitute the public knowledge base.

The present fog-network herd already proves that googolplex-scale virtual populations can be coordinated with compact RNG tokens and sealed enclave pages. The industrial next mile is making the *agents* themselves peer-native—so quest planning, adversity coaching, and multi-agency coordination grow with the mesh instead of with a single REST host.

24 Future Work

24.1 Peer-to-Peer Agent Mesh (Primary Scale Path)

Agentic workloads today are vertically strong (local `gemma3:1b` pools of 64) and horizontally coarse (one binary instance per shard). Future work elevates agents into a **peer-to-peer Plank overlay**: each agent endpoint is both publisher and subscriber on gossip topics under `/plank/agents/*`, discovering neighbors through sealed capability tokens rather than a central dispatcher.

Design constraints for the P2P agent path:

Table 8: P2P agent scalability roadmap

Layer	Today	Future P2P path
Discovery	Static REST base URL	Capability-token gossip + Plank edge hello
Quest fan-out	Instance-local instruct pool	Mesh shard of agent peers with work stealing
Coordination	`herd-bridge` aggregate	Epidemic / epidemic-lite agent consensus
Integrity	Per-ingest enclave pages	Peer-sealed agent digests on every hop
Failure domain	Process restart	Partial mesh heal; no single agent hub

- **No citizen PII on the wire.** Peers exchange opaque person ids, sealed quest digests, and CoL coefficients—never raw narratives outside the local vault boundary.
- **Bounded watts.** Agent gossip rides the existing rosout power/latency edge contract so municipal kiosks and NGO laptops stay inside eco / balanced profiles.
- **Lesser-aligned organization.** Multi-agent control follows the distributed organization framing cited from Victor R. Lesser: many cooperating agents, no monolithic actor, graceful degradation under partial connectivity.
- **Herd-compatible scale.** P2P agents consume the same stream-agnostic NDJSON `PipelineItem` surface already used by `herd-stream` / `herd-subscribe`, so fog generators and agent peers share one pipe grammar (`stdout`, `tcp`, `unix`, `file`, `plank-ingest`).

Operationally, the target asymptotic story is $O(\text{degree})$ local work per agent peer and $O(\log N)$ epidemic diameter across agency meshes—not $O(N)$ fan-in to a single `/v1/agents/*` host. Municipal kiosks, regional NGO meshes, and hybrid cloud edges become interchangeable peers under the same sealed-envelope trust model.

24.2 Supply Chain and Platform Matrix

Complementary industrial workstreams remain:

- Multi-platform bottle matrices (`linux-x64`, `macos-x64`, additional ARM targets)
- Hardware-backed stub shards (secure-enclave / TPM-assisted ML-KEM secret custody)
- Federated Bindi ledger replication for multi-agency governance (periodic root merge across peer shards)
- Cross-agency P2P capability revocation lists published beside ML-DSA-signed manifests

Together these items keep the binary-first, CCSL-1.0a posture while letting agent cohorts scale as a professional fog mesh rather than a fleet of isolated REST islands.

Appendix B: End-to-End Supply Chain

The following sequence describes the path from maintainer release to operator execution without source exposure:

1. Maintainer triggers GitHub Actions workflow on `master` branch push affecting release scripts or public-site assets.
2. Runner builds stripped executable; ML-KEM-768 encapsulates the AES-256-GCM key; envelope and homomorphic seal applied; SHA-256 digest computed.
3. `manifest.json` records version, platform, hashes, `post_quantum` metadata; ML-DSA-65 signs canonical manifest body.
4. Artifact tarball uploads to `public-site/releases/` staging.
5. Next.js static site builds; workflow copies Homebrew formula, releases, and `docs/alu-gubi-cli.pdf` into `website/out/`.
6. GitHub Pages deploys the combined static tree to `shyamalschandra.github.io/alu-gubi-cli/`.
7. Operator runs `brew tap` and `brew trust --formula` on the published formula path.
8. Homebrew downloads tarball from Pages URL, verifies `sha256`, extracts encrypted binary to cellar.
9. First launch executes stub path: homomorphic seal checked, ML-KEM decaps yields K, AES-GCM decrypt, ephemeral exec—no persistent plaintext on disk.
10. Operator uses `ui`, `moby`, and `serve` subcommands; no source tree exists on disk.

This closed loop satisfies industrial requirements for reproducible artifacts, explicit trust prompts, and documentation-only public knowledge transfer.

Appendix C: Glossary

ALU GUBI CLI
: Encrypted command-line platform for gamified Universal Basic Income operations.

Bindi attestation
: Hash-linked record of citizen I/O and CoL adjustments on the internal ledger.

CoL
: Cost of living; adversity-adjusted monthly USD envelope per citizen profile.

GubiMoby
: Pure-Rust Moby-compatible runtime supervising Ollama without Docker Desktop.

GubiVault
: In-process Vault KV v2-compatible secret store with AES-256-GCM at rest.

Hybrid envelope
: Format v2 release: ML-KEM-768 key wrap + AES-256-GCM bulk + ML-DSA-65 signed manifest.

Homomorphic seal
: Integrity tag computable over ciphertext without decryption.

ML-KEM-768
: FIPS 203 module-lattice KEM; encapsulates the release data encryption key.

ML-DSA-65
: FIPS 204 module-lattice signature; authenticates `manifest.json`.

One-way binary
: Release policy shipping executable logic only inside encrypted envelopes.

Herd computing
: Lightweight PlankServices generators emitting compact RNG JSON tokens for a virtual 10^{11}-person population (conversations with Kailash Chandra; reading Victor R. Lesser online).

P2P agent mesh
: Future Plank overlay where agent endpoints gossip sealed quest work as peers (O(degree) local, epidemic diameter), replacing single-host agent fan-in.

Pages manifest
: ML-DSA-signed `manifest.json` listing tarballs, URLs, and SHA-256 hashes.

Acknowledgments

Citations (both modes). This report explicitly cites **both** of the following transmission modes for the intellectual lineage behind PlankServices herd computing and `mathi-alu-gubi` queues:

1. **Conversations with Kailash Chandra.** Oral transmission of discrete-event and stochastic simulation practice from his *Introduction to Simulation* coursework at the Florida Institute of Technology (Florida Tech), Melbourne, Florida—on Florida's Space Coast adjacent to NASA Kennedy Space Center / Cape Canaveral (https://www.fit.edu/). Those conversations condition the "many entities, few materialized states" discipline used by lightweight herd RNG JSON token generators for a virtual population of 10^{11} people.

2. **Reading Victor R. Lesser online.** Study of publicly posted Multi-Agent Systems Laboratory materials by Distinguished Professor Emeritus Victor R. Lesser, University of Massachusetts Amherst (https://mas.cs.umass.edu/lesser.html; lab archive https://mas.cs.umass.edu/), including essays such as *Reflections on being an AI System Architect*. Lesser's multi-agent / distributed AI control and organization inform modular agentic tiers, PlankServices edge coordination, and herd-style generators that cooperate without a single monolithic actor.

Neither mode is secondary: conversation with Kailash Chandra **and** reading Lesser online are joint citations. Full URIs and short citation strings appear in the companion file `CITATIONS.md` on the project Pages site (https://shyamalschandra.github.io/alu-gubi-cli/CITATIONS.md).

Operators deploying ALU GUBI in production should cite this report, respect CCSL-1.0a terms, and observe the OpenAI ChatGPT defensive-licensing conditioning note.

Install (binary only):

```
brew tap shyamalschandra/alu-gubi-cli
brew trust --formula shyamalschandra/alu-gubi-cli/alu-gubi-cli
brew install alu-gubi-cli
```

Document: https://shyamalschandra.github.io/alu-gubi-cli/
Manifest: https://shyamalschandra.github.io/alu-gubi-cli/releases/manifest.json