Qwen

From Rights to Runtime: A Technical Blueprint for Sovereign AI Enforced by Rust Kernels and ALN Shards

Architectural Foundations: Rights as Non-Bypassable System Invariants

The proposed sovereign AI-chat framework represents a paradigm shift in AI governance, moving away from a model where rights and ethical guidelines are treated as optional, application-level policies to one where they are encoded as fundamental, non-bypassable invariants within the system's technical substrate [3] [56]. This foundational premise is designed to address a critical weakness in conventional AI systems: the potential for enforcement logic to be circumvented, ignored, or ambiguously interpreted due to its placement outside the core execution environment. By embedding rights directly into the system's runtime, the framework aims to create a verifiably trustworthy environment where user sovereignty is not merely a feature but a structural property of the software itself. This design philosophy aligns with the principle of minimizing the Trusted Computing Base (TCB)—the set of components responsible for security policy enforcement—by reducing its scope to the verification of a small, well-defined guard component rather than trusting the entire application suite to adhere to high-level directives [3]. The TCB in this context would consist primarily of the OS kernel and the specialized guard kernel, which together are tasked with enforcing the security policy [3].

This architecture is predicated on three core pillars: a Rust-based guard kernel for deterministic enforcement, an ALN (Assumed Language of Nomenclature) shard system for structured policy representation, and a neurorights/rights-kernel enforcement layer that translates abstract principles into executable code [74] [86]. These components are not merely co-located; they are deeply interwoven to form a single, cohesive stack. The internal coherence of this stack is paramount to its viability. Each component has a distinct and complementary role. The ALN shards serve as the declarative specification of rules, akin to policy-as-code [5]. The Rust guard kernel acts as the imperative enforcement engine, systematically checking all actions against these rules at runtime [74]. Finally, the UI and dashboard components provide transparent feedback, making the

operational boundaries of the system visible and auditable by the user [26]. This separation of concerns ensures that changes to policy (managed via ALN shards) do not require recompilation of the core enforcement logic, while the enforcement logic itself remains secure and reliable due to Rust's inherent properties [1] [15].

A key distinction from baseline AI systems lies in the concept of "hard floors." Mainstream AI models often operate under soft constraints defined in system prompts or external configuration files, which are subject to misinterpretation by the model or manipulation by external actors [8]. For instance, a prompt might ask a model to avoid harmful content, but there is no guarantee of enforcement. The proposed framework elevates constraints like neurorights and risk-of-harm thresholds (`RoH ≤ 0.3`) to the status of absolute, enforceable limits that cannot be overridden by the model's training or user request [74]. When a suggestion is blocked, the system explicitly attributes the denial to a specific constraint, such as "Reason: neurorights floor," thereby demystifying the decision-making process and holding the system accountable [74]. This contrasts sharply with opaque vendor policies that simply block content without providing a clear, traceable justification. The system's design forces the model to engage with a formal, constrained reasoning space, preventing it from suggesting actions that violate the user's established biophysical and legal envelopes [42] [45].

The framework's architecture also anticipates a future where AI governance must be adaptive and jurisdiction-aware. The use of modular ALN shards allows for the dynamic loading of policy documents based on a user's location and preferences [42]. For example, a user in Phoenix would have local laws from Maricopa County automatically merged with broader US and international standards, with a "strictest-wins" resolution strategy applied [42]. This requires a sophisticated policy aggregation engine capable of resolving conflicts between different legal and ethical frameworks. This capability moves beyond simple compliance checks and into the realm of contextualized, multi-layered governance, reflecting a more nuanced understanding of real-world regulatory environments. The ultimate goal is to create a system where sovereignty is actively maintained through technical mechanisms, allowing users to navigate pathways of evolution-by-choice with explicit consent and provable continuity, as evidenced by the Sovereign Continuity Dashboard [27] [74]. This approach seeks to establish a new standard for trustworthy AI, grounded in principles of transparency, determinism, and user-centric control.

Feature	Proposed Sovereign AI Chat	Baseline AI Systems
Governance Model	Rights as non-bypassable system invariants enforced by a dedicated guard kernel [56].	Policies as optional, application-level constraints managed in separate layers (e.g., JSON, database) [70].
Enforcement	Deterministic and transparent; blocks actions before execution and provides a specific reason for denial [74].	Stochastic and opaque; relies on model behavior and may fail to prevent violations; denials are often unexplained [54].
Invariants	Hard floors for neurorights, `RoH ≤ 0.3`, rollback strength, etc., are built into the core logic [74].	No strict containment of invariants; values are heuristic and can be violated [46].
Policy Management	Structured, machine-readable ALN shards allow for dynamic, jurisdiction-aware policy updates without recompiling core logic [42].	Static or semi-static policies managed externally; difficult to adapt to changing regulations [5].
Transparency	Explicit UI tags show the source of constraints (e.g., "neurorights floor") [74]. Immutable audit trail (timeline view) [27].	Limited transparency; decisions are often black-box, making auditing and trust-building difficult.
User Control	User actively defines and manages consent envelopes, experiment parameters, and duty vectors [42] [74].	User has limited control over underlying model behavior and data usage policies [34].

This architectural blueprint, while ambitious, is rooted in established principles of secure systems design. It draws inspiration from efforts to build sovereign kernels for AI, which are designed to wrap and govern any underlying AI technology, and from research into proactive runtime enforcement of LLM agent behaviors [54] [56]. The challenge lies not in the conceptual novelty but in the immense technical and intellectual labor required to translate abstract human rights and ethical obligations into precise, verifiable computational rules—a task that sits at the intersection of computer science, law, and neuroethics [6] [33].

The Rust Guard Kernel: An Engine for Deterministic Policy Enforcement

The choice of Rust as the implementation language for the guard kernel is a strategic decision central to the framework's security and reliability goals. Rust's design philosophy prioritizes memory safety and predictable performance without compromising low-level control, making it exceptionally well-suited for building a minimal, trusted enforcement component [1] [15]. Unlike languages such as C/C++, which require manual memory management and are prone to vulnerabilities like buffer overflows and data races, Rust's compile-time borrow checker statically prevents many common classes of bugs [15] [53]. This is not merely a performance optimization but a mission-critical feature for a

component whose sole purpose is to protect the integrity of user rights. A bug in the guard kernel could create a backdoor, allowing actions to bypass the very constraints it is designed to enforce. By using safe Rust, the development team can significantly reduce the probability of such catastrophic failures, creating a more stable and secure foundation for the entire system [1]. Microsoft's investment in Rust for similar reasons further validates this path toward eliminating security debt and streamlining DevOps processes [17].

Beyond memory safety, Rust's performance characteristics are crucial for the real-time nature of an AI chat interface. Many AI workloads, particularly those involving inference and actuation, demand low latency and predictable execution times [1]. Python, a popular language for AI development, suffers from the Global Interpreter Lock (GIL), which can limit true parallelism and introduce unpredictable performance bottlenecks [1]. Rust, being a compiled, statically-typed language, offers near C/C++ speeds with fine-grained control over memory and threads, ensuring deterministic behavior even under heavy load [1]. Anecdotal performance comparisons have shown Rust can achieve speedups of over 80x compared to pure Python implementations for vector-heavy operations, a critical consideration for processing embeddings and other data-intensive tasks in the AI pipeline [1]. This performance advantage allows the guard kernel to perform its validation checks with minimal overhead, ensuring that the user experience remains fluid and responsive. The kernel can be designed as a high-performance engine that integrates seamlessly with higher-level Python interfaces, leveraging Rust's ability to expose clean, high-level APIs to other languages [1] [24].

The function of the guard kernel is to act as a deterministic, transparent, and modular framework that intercepts all potentially impactful actions before they are executed [74]. This concept is inspired by recent research into enforcing business policy adherence in agentic workflows, which proposes a two-phase process: an offline "buildtime" stage that compiles policy documents into verifiable guard code, and a runtime integration where these guards execute before each agent action [74] [86]. In this model, the guard kernel would instantiate `ToolGuard` functions for every possible action the AI could take. Before an action is permitted, the corresponding guard function is invoked, passing it relevant inputs such as the tool-call arguments, chat history, and state from read-only APIs [86]. The guard then executes its validation logic, which is derived directly from the ALN policy shards. If the guard returns a success signal, the agent proceeds; otherwise, the action is blocked [86]. This design supports modularity by isolating each policy into a separate, atomic function, simplifying the generation, testing, and interpretation of the guards [86]. The use of a test-driven development (TDD) paradigm during the "buildtime" compilation phase, where examples of compliant and violating scenarios are generated

from policy text, ensures that the resulting guard code is both correct and aligned with the original intent of the policy [86].

Despite Rust's strengths, several risks and complexities remain. The most significant challenge is not the language itself but the complexity of correctly implementing the logical rules within the guards. Even with memory safety, the logic governing rights, lifeforce envelopes, and jurisdictional laws will be intricate. As demonstrated by the need for libraries like Pydantic in Python pipelines to prevent "cognitive drift"—where small errors compound into unreliable outputs—the guard kernel must perform similarly rigorous validation across multiple, interacting dimensions [70]. Furthermore, while safe Rust code is memory-safe, interactions with external code via the Foreign Function Interface (FFI) remain a potential attack vector [78]. Safely encapsulating FFI calls to protect the surrounding safe Rust code is a critical area of research, with approaches like Encapsulated Functions combining hardware-based memory protection with Rust type abstractions to facilitate safer interoperability [78]. Another major risk is the potential for control-flow hijacking attacks, which corrupt program execution paths. While hardware-assisted mitigations like Intel's Control Flow Enforcement Technology (CET) and ARM's Pointer Authentication (PA) offer strong defenses, sophisticated attacks can still exploit design trade-offs in these schemes, highlighting the need for a defense-in-depth strategy [71]. Ultimately, the feasibility of the guard kernel hinges on a meticulous engineering effort to translate complex, high-level policies into precise, verifiable, and secure Rust code.

ALN Shards: The Structured Knowledge Fabric for Dynamic Governance

The ALN (Assumed Language of Nomenclature) shard system is the nervous system of the sovereign AI framework, providing a structured, machine-readable, and dynamically updatable fabric of knowledge upon which all governance decisions are based. This approach directly implements the concept of "policy-as-code," where legal, ethical, and personal constraints are not stored as static, human-readable documents but as discrete, interoperable data units called "shards" [5]. Examples of these shards include `neurorights.envelope`, `policy.jurisdiction.*`, and `ethics.quality.corridor.v1`, each representing a specific facet of the user's operational envelope [42] [45]. This design decouples the definition of policy from its execution, a powerful architectural pattern that allows policy designers to update

constraints without requiring developers to recompile the core enforcement engine [86]. This separation is crucial for creating a system that can adapt to evolving legal landscapes, such as new neurorights legislation, or to a user's changing personal preferences and biophysical state.

The integration of ALN shards into the chat prompt and action pipeline is a multi-stage process that begins long before a user submits a query. During session initialization, the system first identifies the user's profile and geographic jurisdiction. Based on this information, it queries a policy repository to retrieve the relevant ALN shards. For a user in Phoenix, this would involve fetching shards from a hierarchy that includes global standards, US federal law, Arizona state law, and Maricopa County ordinances [42]. These disparate sources must then be parsed, validated, and aggregated. A key challenge here is conflict resolution. The system must implement a policy like "strictest-wins," where a more restrictive local regulation (e.g., a city ordinance limiting certain types of neural augmentation) overrides a less restrictive national guideline [42]. This requires a sophisticated policy engine capable of understanding the relationships between different legal and ethical frameworks and merging them into a single, coherent policy document that the guard kernel can understand. Once merged, this document forms the basis for instantiating the session's live `RightsProfile` and `LifeforceEnvelope` snapshots [74].

As the user interacts with the chat, these live snapshots become the active context for every proposed action. When a user requests an upgrade or performs an action, the system translates this request into a formal proposal. The Rust guard kernel then consults the live `RightsProfile` and `LifeforceEnvelope`—derived from the ALN shards—to evaluate the proposal's impact. For example, if a user requests an upgrade that would increase their computed `Risk-of-Harm` (RoH) above the `≤ 0.3` threshold, the guard kernel, guided by the rule encoded in the ethics shard, would reject the proposal [74]. The system's transparency requirement dictates that the rejection must be accompanied by a specific tag indicating the reason, such as "Reason: RoH floor exceeded," which is then rendered in the user interface [74]. This creates a closed-loop system where the ALN shards continuously inform the guard kernel, which in turn enforces the rules derived from those shards, and the results are made visible to the user.

The feasibility of this entire pillar rests on the existence of a robust and mature ALN specification and its associated toolchain. Several critical uncertainties must be addressed. First is the definition of the ALN syntax and semantics. What does a typical `ExperimentEnvelope` shard look like? How are hypotheses, metrics, and hard-stop conditions formally represented? Developing a prototype ALN specification using a

schema language like JSON Schema or Protobuf would be a necessary first step. Second is the creation of a reliable ALN parser and validator. Any error in parsing an ALN shard could lead to incorrect enforcement, either by failing to apply a necessary constraint or, worse, by introducing a vulnerability that could be exploited to bypass rights. Third is the handling of ambiguity. Real-world laws and ethics are often ambiguous. The system needs a mechanism to handle situations where a policy is unclear or where different rights come into direct conflict (e.g., the right to self-augmentation versus the right to physical integrity). The "Experiment envelope builder" feature attempts to address this by allowing users to create their own, formally-defined exceptions with explicit consent, but a general-purpose conflict-resolution strategy is also needed [74]. Finally, the system must support versioning and provenance for all ALN shards. Every change to a policy should be cryptographically signed and timestamped, forming part of the immutable audit trail that is essential for accountability and verifying the "rights never shrank" timeline [27]. Without a solid foundation in formal methods and data integrity, the ALN shard system remains a theoretical abstraction rather than a functional component of the framework.

Neurorights and Kernel Enforcement: Translating Ethical Principles into Code

The heart of the sovereign AI framework is its commitment to enforcing a set of "hard floors" derived from neurorights and other critical invariants. This involves a deep translation process, converting abstract philosophical and legal concepts into precise, computationally verifiable rules that can be embedded in the guard kernel [6] [33]. Neurorights, a term coined to address the ethical and legal challenges of neurotechnology, typically encompass the right to mental privacy, cognitive freedom, identity integrity, and psychological continuity [42] [45]. Chile was the first country to enshrine these rights in its constitution, establishing protections for mental integrity and mandating special safeguards for cerebral activity data [42] [45]. The proposed framework takes this a step further by treating these rights not as aspirational ideals but as absolute, non-negotiable constraints that the system's core logic must uphold at all times. Similarly, technical invariants like a `Risk-of-Harm` (RoH) metric capped at ≤ 0.3 are elevated to the same level of importance, forming part of the system's fundamental operating envelope [74].

The process of translating these rights into code is fraught with difficulty and represents the most intellectually demanding aspect of the project. It requires a multidisciplinary

effort to create a formal ontology for these concepts. For instance, how is "mental privacy" defined algorithmically? Does it mean prohibiting the AI from accessing certain types of user data, summarizing conversations in a way that reveals sensitive information, or suggesting actions that could be perceived as invasive? The `Lifeforce-envelope` and `CyberMode` states provide a starting point for formalization, representing quantifiable aspects of the user's biophysical condition [74]. However, translating qualitative rights like "free will" or "psychological continuity" into executable code is a monumental challenge. Critics of Chile's neurorights legislation have pointed out its potential vagueness, arguing that it fails to harmonize with established international human rights norms and may be premature [45]. The project must learn from such critiques and strive for precision and clarity in its formal definitions.

Once a concept is formally defined, it must be encoded into the guard kernel's validation logic. This can be achieved through a combination of static rule-based checks and dynamic analysis. For example, a rule might state: "If the `user.jurisdiction` is 'Phoenix', then the effective `neurorights.envelope.augmentation_limit` is set to 0.5." This rule would be compiled into the guard code during the buildtime phase [86]. Dynamic checks would involve analyzing the impact of a proposed action. If a user requests an upgrade, the system would calculate the projected change in the `RoH` metric. If the result exceeds the `≤ 0.3` threshold, the guard would trigger a rejection event. The `Sobriety / compulsion guardrails` feature exemplifies this dynamic approach, proposing meters like PDR (Psych-Risk Score) to monitor the potential for compulsion, defaulting to proposals that reduce addiction residuals and blocking upgrades that widen these envelopes without explicit, evidence-tagged consent [74].

The system's enforcement capabilities extend to managing complex trade-offs and fostering user agency. The "Evolution-by-choice planner" is a prime example of a feature designed to guide users through permissible upgrade paths that respect their established envelopes [74]. It reads the BiophysicalRightsKernel, Lifeforce, and CyberMode states to return only admissible options, comparing them by capability gain versus energy/protein/thermo cost [74]. This transforms the abstract notion of "evolution-by-choice" into a concrete, interactive planning tool. The "Experiment envelope builder" serves a similar purpose but for temporary, self-defined deviations from the norm [74]. By committing an experiment as a formal ALN shard with explicit hypotheses, duration, and hard-stop conditions (e.g., pain, inflammation, HRV/IL-6 bounds), the user creates a sandboxed environment where the normal constraints are relaxed for a defined period [74]. The chat would then refuse to suggest any action outside that envelope unless the user explicitly amends the shard with fresh consent, demonstrating a sophisticated balance between safety and autonomy.

However, the greatest risk in this domain is the potential for overly rigid enforcement to stifle legitimate innovation and personal growth, or conversely, for weak enforcement to fail in its protective mission. Striking the right balance is delicate. The framework attempts to mitigate this risk through transparency and user control. The explicit tagging of denied actions, the Sovereign Continuity Dashboard showing a timeline of rights, and the ability for users to petition for stricter local envelopes via ALN particles instead of emails are all mechanisms designed to empower the user and maintain a healthy tension between safety and freedom [42] [74]. The ultimate success of neuroright enforcement depends on the quality of the initial formalization and the flexibility of the subsequent implementation, ensuring that the system acts as a true partner in augmenting human capability, not a gatekeeper that arbitrarily restricts it.

System Coherence and Feasibility: Integrating Components into a Verifiable Stack

The internal coherence of the proposed sovereign AI stack depends entirely on the seamless and logically consistent interaction between its three core components: the ALN shard data fabric, the Rust guard kernel enforcement engine, and the user-facing interfaces that provide transparency. A coherent workflow ensures that there are no logical gaps or points of failure where constraints can be bypassed. The proposed process appears sound, forming a closed loop of policy definition, enforcement, and feedback. The sequence begins with the initialization of a chat session, where the system loads and aggregates relevant ALN shards to construct a live `RightsProfile` and `LifeforceEnvelope` snapshot [74]. When a user submits a prompt or command, the request is intercepted by the guard kernel *before* the LLM generates a response or an action is executed [74]. The kernel then validates the proposed action against the live state and the rules encoded in the ALN shards. If the action is compliant, it proceeds; if not, the kernel blocks it and returns a detailed, cryptographically signed reason for the denial, which is then presented to the user in the chat interface [27] [74]. All these events, including the guard's decision, are logged and attested, creating an immutable audit trail accessible via the Sovereign Continuity Dashboard [26] [27]. This workflow demonstrates strong internal coherence, with each component fulfilling a distinct and necessary role without overlap or ambiguity.

From a technical feasibility perspective, the individual components are largely within the realm of current technology, though their integration presents significant engineering

challenges. The use of Rust for the guard kernel is a mature and defensible choice, supported by growing adoption in performance-critical AI backend systems [1]. Integrating a high-performance Rust service with Python-based LLM frameworks is feasible through established Application Binary Interfaces (ABIs) and tools like PyO3 or FFI. However, managing data marshaling, concurrency, and error handling between these different language runtimes introduces complexity that must be carefully managed [1]. The formalization of ALN shards and the development of a robust parser/validator represent a more novel challenge, requiring significant upfront work in language design and specification. The feasibility of the entire system is contingent on successfully completing this foundational work.

To elevate the system from "secure-by-design" to "verifiably-trusted," incorporating technologies for confidential computing is a logical and powerful next step. Running the guard kernel and the user's sensitive data within a hardware-backed Trusted Execution Environment (TEE), such as those offered by Intel SGX or AMD SEV-SNP, would provide a strong guarantee that the code running is exactly the code that was intended to run [28] [30]. Technologies like Enarx already provide the necessary attestation mechanisms, allowing a remote verifier to confirm the integrity and trustworthiness of the host and the guest's runtime environment [26]. Remote attestation, a key feature of TEEs, allows the system to prove to a third party that its control flow path is compliant with a pre-computed Control Flow Graph (CFG), providing unforgeable evidence of its execution [31] [71]. This would protect the guard kernel from tampering and ensure the confidentiality of the user's `RightsProfile` and `LifeforceEnvelope`. While CFI and CFA are distinct concepts—one focused on local prevention and the other on remote verification—they share a common goal of controlling and proving program execution, a goal central to the sovereignty of this AI stack [71].

Despite the promising architecture, several critical uncertainties and gaps must be addressed for the project to succeed. The most significant is the lack of precise, quantitative definitions for key concepts like "neurorights" and the "Risk-of-Harm" (RoH) metric. Without a formal, mathematical definition, the guard kernel cannot perform its validation function. This requires a deep collaboration between engineers, neuroscientists, ethicists, and legal scholars to develop a shared ontology. Another gap is the lack of detail on how the system handles ambiguity and complex ethical trade-offs that go beyond simple binary accept/reject decisions. Moving beyond a purely rule-based system to incorporate elements of ethical reasoning, such as constraint solvers or multi-agent negotiation frameworks, may be necessary [46]. Finally, the human-in-the-loop component is underdeveloped. While the UI is mentioned, a clear workflow for escalating ambiguous decisions to a human overseer when the AI cannot determine compliance is

not specified. Designing such a workflow is crucial for handling edge cases and maintaining user trust. Addressing these gaps through interdisciplinary research and careful engineering will be the determining factor in whether this technically coherent vision can become a viable and trustworthy reality.

Component	Role in System	Key Technologies & Concepts	Feasibility Status
ALN Shard Fabric	Defines the declarative rules for governance (policies, rights, constraints).	Formal language design, schema validation (e.g., JSON Schema), policy aggregation engines, cryptographic signing for provenance.	High uncertainty; requires significant foundational R&D in formalizing ethics/law.
Rust Guard Kernel	Executes the imperative enforcement logic, validating all actions against ALN rules.	Rust programming language, deterministic algorithms, `ToolGuard` functions, test-driven development (TDD), Foreign Function Interface (FFI) encapsulation.	Technically feasible with existing tools; main challenge is implementing complex validation logic correctly.
Confidential Computing Layer	Provides a hardware-backed, verifiable environment for the guard kernel and sensitive data.	Trusted Execution Environments (TEEs) like Intel SGX/SEV-SNP, remote attestation, Confidential Virtual Machines (CVMs), Integrity Measurement Architecture (IMA).	Technically feasible; requires specialized hardware and mature software stacks like Enarx or HyperTEE.
Attestation & Logging	Creates an immutable, cryptographically verifiable audit trail of all system actions and decisions.	Merkle trees, zero-knowledge (ZK) proofs, digital signatures, append-only logs, hex-stamped events.	Technically feasible using established cryptographic primitives and distributed ledger concepts.

Comparative Baseline and Risk Analysis: Deficiencies of Conventional Systems

The proposed sovereign AI framework derives much of its value from what it explicitly avoids. By contrasting its architecture with that of mainstream AI systems, the deficiencies of the conventional approach become starkly apparent. Most current AI systems, from large language models to autonomous agents, are built on a foundation of stochastic behavior and opaque decision-making processes [54]. Their governance is typically implemented as a post-hoc filtering layer or a set of instructions within a system prompt, rather than being woven into the core execution substrate [8]. This fundamental architectural difference leads to a cascade of problems related to reliability, accountability, and user sovereignty. The baseline systems lack invariant containment mechanisms, fail to enforce neurorights hard floors, and cannot guarantee that user-defined safety envelopes will be respected, making them inherently unsuited for applications involving personal augmentation and life-critical decisions.

One of the most glaring deficiencies is the absence of a minimal, verifiable Trusted Computing Base (TCB) [3]. In a typical deployment, the entire application—including the parts responsible for generating output, managing state, and applying filters—is part of the TCB. There is no small, isolated component whose integrity can be verified to ensure that all other parts are operating correctly. This means that any bug, vulnerability, or malicious modification in the application code could lead to a violation of user constraints. For example, a subtle bug in a piece of Python code could cause a safety check to be skipped, or a compromised dependency could introduce a backdoor. The sovereign framework addresses this by isolating the enforcement logic into a small, secure Rust kernel, whose integrity can be independently verified [1] [74]. This shifts the burden of trust from the entire application to a small, well-understood component.

Another critical failure of baseline systems is their inability to contain invariants. The `RoH ≤ 0.3` constraint is a prime example of a non-negotiable boundary that the sovereign framework treats as a hard floor [74]. In a conventional system, such a constraint would likely exist as a heuristic within the model's reward function or as a prompt instruction. There is no guarantee that the model will respect it. LLMs can exhibit emergent behaviors that were not intended by their designers, and they can be susceptible to adversarial prompting techniques that trick them into violating safety protocols [82]. The sovereign framework's guard kernel, however, performs a deterministic check against this invariant before any risky action is permitted, effectively neutralizing the threat of stochastic failure in this regard. Similarly, neurorights, as defined by bodies like the NeuroRights Foundation and codified in laws like Chile's Article 19 reform, are treated as absolute legal and ethical boundaries [42] [44]. A baseline system has no mechanism to enforce these rights; it can only hope the model behaves ethically. The sovereign framework embeds these rights into its very DNA, making it impossible for the system to suggest an action that violates them.

The following table summarizes the key differences in risk and control between the two approaches:

Aspect	Proposed Sovereign AI Framework	Baseline AI Systems
Invariant Containment	Guarantees invariants (`RoH ≤ 0.3`, neurorights) through deterministic, pre-execution checks by a trusted guard kernel [74].	Relies on stochastic models and heuristic filters; invariants are not guaranteed and can be violated [54].
Trust Model	Minimal TCB centered on a verifiable guard kernel, reducing the trusted surface area [3].	Large, monolithic TCB where the entire application is implicitly trusted to respect high-level policies.
Accountability & Transparency	Provides explicit, cryptographically signed reasons for action denials ("Reason: neurorights floor") and an immutable audit trail [27] [74].	Decisions are often opaque and unexplainable, making it difficult to hold the system accountable for its outputs.
User Control	Empowers users to define and manage consent envelopes, experiment parameters, and duty vectors via structured ALN shards [74].	Users have limited control over model behavior and data usage policies, which are dictated by the vendor [34].
Adaptability	Policy can be updated dynamically via ALN shards without recompiling the core system, enabling adaptation to new laws or user preferences [86].	Policies are often static or require complex redeployment cycles, leading to slow adaptation to new regulations.

The risks associated with the proposed framework are therefore not primarily technical but intellectual and philosophical. The foremost risk is the difficulty of formalizing complex, nuanced concepts like "neurorights" into precise, executable rules. As noted in critiques of Chile's legislation, oversimplification can lead to ineffective or counterproductive laws [45]. The project must navigate this challenge with extreme care to avoid creating a system that is either dangerously permissive or stiflingly restrictive. A second major risk is the potential for the system to become a target for attackers seeking to exploit the guard kernel or the ALN shard repository. While Rust provides a strong defense against many common vulnerabilities, a determined attacker could find novel exploits, especially at the FFI boundary or in the policy aggregation logic [78]. Rigorous security audits, fuzz testing, and potentially formal verification of the guard kernel's core logic will be essential to mitigate this risk [75]. Finally, there is the risk of implementation complexity. Building and integrating all the required components—a custom language, a high-performance kernel, a confidential computing layer, and a sophisticated UI—into a single, coherent, and reliable system is an enormous undertaking. The project's success will depend on disciplined engineering practices, a clear division of responsibilities, and a realistic assessment of the time and resources required.

Reference

1. Rust for AI Systems: Predictable Performance and Security https://www.linkedin.com/posts/vsadhwani_if-youre-expanding-into-systems-programming-activity-7404533909394321408-7-G2
2. Friend or Foe Inside? Exploring In-Process Isolation to ... https://arxiv.org/html/2306.08127v2
3. arXiv:2004.04846v1 [cs.OS] 9 Apr 2020 https://arxiv.org/pdf/2004.04846
4. The Annual AI Governance Report 2025 https://www.itu.int/epublications/publication/the-annual-ai-governance-report-2025-steering-the-future-of-ai
5. Governing with Artificial Intelligence (EN) https://www.oecd.org/content/dam/oecd/en/publications/reports/2025/06/governing-with-artificial-intelligence_398fa287/795de142-en.pdf
6. ETHICALLY ALIGNED DESIGN http://standards.ieee.org/wp-content/uploads/import/documents/other/ead_v2.pdf
7. Fixing Rust Compilation Errors using LLMs https://arxiv.org/pdf/2308.05177
8. How to pass system prompt to claude 3 haiku on AWS ... https://stackoverflow.com/questions/78850311/how-to-pass-system-prompt-to-claude-3-haiku-on-aws-bedrock-using-javascript-sdk
9. Fixing Rust Compilation Errors using LLMs | Request PDF https://www.researchgate.net/publication/373046288_Fixing_Rust_Compilation_Errors_using_LLMs
10. 人工智能2024_1_12 https://arxivdaily.com/thread/51222
11. A Survey of Context Engineering for Large Language Models https://arxiv.org/html/2507.13334v1
12. Findings of the Association for Computational Linguistics https://aclanthology.org/volumes/2025.findings-acl/
13. Theory of Awareness D1.4 - European Commission - European Union https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e51b4201f9&appId=PPGMS
14. SandCell: Sandboxing Rust Beyond Unsafe Code https://arxiv.org/html/2509.24032v2

15. Rust Helps Teams Reduce Security Debt | Anthony Barkley ... https://www.linkedin.com/posts/anthonybarkley_rust-code-delivers-security-streamlines-activity-7408943164776140801-ZrSe
16. Azure updates https://azure.microsoft.com/updates?id=523172
17. Microsoft Invests in Rust for Memory Safety and Security https://www.linkedin.com/posts/kirkwlawrence_securebydesign-securebydefault-rust-activity-7422648600679129088-gXxv
18. (PDF) Artificial Intelligence and Neuroscience https://www.researchgate.net/publication/388087540_Artificial_Intelligence_and_Neuroscience_Transformative_Synergies_in_Brain_Research_and_Clinical_Applications
19. Spiking the Transformer: NeuViT with Single-Step Attention ... https://www.sciencedirect.com/science/article/pii/S2590123025049977
20. Publication 1500 (Rev. 6-2025) https://www.irs.gov/pub/irs-pdf/p1500.pdf
21. (PDF) Recent Advances in Deep Learning Techniques for ... https://www.researchgate.net/publication/353158272_Recent_Advances_in_Deep_Learning_Techniques_for_Face_Recognition
22. AutoML: A systematic review on automated machine ... https://www.sciencedirect.com/science/article/pii/S2949715923000604
23. Scientific Abstracts and Sessions - 2017 - Medical Physics https://aapm.onlinelibrary.wiley.com/doi/10.1002/mp.12304
24. Planet Mozilla https://planet.mozilla.org/
25. 8.8 Release Notes | Red Hat Enterprise Linux | 8 https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/8/html-single/8.8_release_notes/index
26. Enabling Integrity Measurement for Secure Applications in the ... https://link.springer.com/article/10.1007/s10922-025-09983-4
27. (PDF) VIMA: A Privacy-Preserving Integrity Measurement ... https://www.researchgate.net/publication/395315103_VIMA_A_Privacy-Preserving_Integrity_Measurement_Architecture_for_Containerized_Environments
28. HyperTEE: A Decoupled TEE Architecture with Secure ... https://cse.sustech.edu.cn/faculty/~zhangfw/paper/hypertee-micro25.pdf
29. A Comprehensive Trusted Runtime for WebAssembly with ... https://arxiv.org/pdf/2312.09087
30. An Empirical Analysis of AMD SEV-SNP and Intel TDX https://dl.acm.org/doi/abs/10.1145/3700418

31. D5.1 Design and early release of Security, Safety and ... https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5e069b205&appId=PPGMS
32. AI and the future of education: disruptions, dilemmas and ... https://unesdoc.unesco.org/ark:/48223/pf0000395236
33. The Netherlands: artificial intelligence readiness ... https://unesdoc.unesco.org/ark:/48223/pf0000393240
34. AI4IA conference report, 2021 https://unesdoc.unesco.org/ark:/48223/pf0000381489
35. UNESCO global report on cultural policies, Culture https://unesdoc.unesco.org/ark:/48223/pf0000395504
36. UNESCO science report: towards 2030 https://unesdoc.unesco.org/ark:/48223/pf0000235406
37. Governance: a 'whole-of-society' approach https://unesdoc.unesco.org/ark:/48223/pf0000385215
38. Investing in cultural diversity and intercultural dialogue https://unesdoc.unesco.org/ark:/48223/pf0000185202
39. a critical survey of supply-and-demand situations in the ... https://unesdoc.unesco.org/ark:/48223/pf0000039466
40. The Ability of units for R&D in higher education ... https://unesdoc.unesco.org/ark:/48223/pf0000057008
41. YSF Thematic Publication Chapter e-book https://www.researchgate.net/profile/Samantha-Dissanayaka/publication/367799692_YSF_Thematic_Publication-2023/links/63dab96564fc86063805877b/YSF-Thematic-Publication-2023.pdf
42. Neurorights in the Constitution: from neurotechnology to ethics ... https://pmc.ncbi.nlm.nih.gov/articles/PMC11491849/
43. Chile: pioneering the protection of neurorights https://unesdoc.unesco.org/ark:/48223/pf0000380275_eng
44. Neurotechnology https://www.ohchr.org/sites/default/files/documents/hrbodies/hrcouncil/advisorycommittee/neurotechnology/03-ngos/ac-submission-cso-neurorightsfoundation.pdf
45. Chilean neurorights legislation and its relevance for mental ... https://www.researchgate.net/publication/373903709_Chilean_neurorights_legislation_and_its_relevance_for_mental_health_Criticisms_and_outlook
46. Can LLMs Clean Up Your Mess? A Survey of Application- ... https://arxiv.org/html/2601.17058v1

47. Exploring the roles of large language models in reshaping ... https://www.sciencedirect.com/science/article/pii/S3050860625000031

48. A Comprehensive Overview of Large Language Models https://dl.acm.org/doi/full/10.1145/3744746

49. Industrial applications of large language models https://www.nature.com/articles/s41598-025-98483-1

50. Securing local LLMs for academic research: a human-system ... https://link.springer.com/article/10.1007/s42454-025-00085-9

51. LLMs to the Rescue: Explaining DSA Statements of ... https://aclanthology.org/2024.nllp-1.17.pdf

52. Applications of Large Language Models and Multimodal ... https://www.mdpi.com/2504-446X/9/4/238

53. How to run LLM code safely in Rust or Python https://www.linkedin.com/posts/saurav1verma_run-your-llm-code-in-a-type-safe-compiled-activity-7352461913768841216-8ftk

54. Pro2Guard: Proactive Runtime Enforcement of LLM Agent ... https://arxiv.org/abs/2508.00500

55. Attestation vs. integrity in a zero-trust world https://www.redhat.com/en/blog/attestation-vs-integrity-zero-trust-world

56. Building a sovereign kernel for AI with governance and ethics https://www.linkedin.com/posts/nato-riley_dare2try-activity-7393038857438453760-9b7U

57. fakult¨at f¨ur informatik https://www.researchgate.net/profile/Mohsen-Ahmadvand-2/publication/352551096_Strengthened_Composable_and_Quantifiable_Software_Integrity_Protection/links/60cf0b06a6fdcc01d4870e1c/Strengthened-Composable-and-Quantifiable-Software-Integrity-Protection.pdf

58. A CNN Regression Approach for Real-time 2D/3D ... https://www.researchgate.net/publication/292075617_A_CNN_Regression_Approach_for_Real-time_2D3D_Registration

59. Design of Reinforced Concrete Panels for Wind-borne Missile ... https://www.researchgate.net/profile/Brian_Terranova/publication/319263030_Design_of_reinforced_concrete_panels_for_wind-borne_missile_impact/links/5c863b7792851c69506baee2/Design-of-reinforced-concrete-panels-for-wind-borne-missile-impact.pdf

60. COST Action INTERACT WG2 Whitepaper https://www.researchgate.net/publication/387053865_COST_INTERACT_Whitepaper_on_Signal_Processing_for_Communications_Localization_and_Intergrated_Sensing_and_Communication/fulltext/675d592dda24c8537c6efed0/COST-INTERACT-Whitepaper-on-Signal-Processing-for-

Communications-Localization-and-Intergrated-Sensing-and-Communication.pdf?origin=scientificContributions

61. [Algorithms for Intelligent Systems] Cybernetics, Cognition ... https://www.researchgate.net/profile/Vinit-Gunjan/publication/340805171_Cybernetics_Cognition_and_Machine_Learning_Applications_Proceedings_of_ICCCMLA_2019_Proceedings_of_ICCCMLA_2019/links/62d3c95dfd347a451bc5051f/Cybernetics-Cognition-and-Machine-Learning-Applications-Proceedings-of-ICCCMLA-2019-Proceedings-of-ICCCMLA-2019.pdf
62. (PDF) Perception, control and path planning of robotic ... https://www.researchgate.net/publication/358647124_Perception_control_and_path_planning_of_robotic_laparoscopic_surgical_system
63. Biomedical Sciences Instrumentation - 57(2) https://www.researchgate.net/profile/Kaitlin-Dailey-2/publication/350740799_Probing_Clinical_Relevance_Establishing_the_Efficacy_of_C_novyi_Against_a_Panel_of_2D_Cultured_Pancreatic_Cancer_Cells/links/60fb2212169a1a0103b1e730/Probing-Clinical-Relevance-Establishing-the-Efficacy-of-C-novyi-Against-a-Panel-of-2D-Cultured-Pancreatic-Cancer-Cells.pdf
64. (PDF) Automated Machine Vision System for Liquid ... https://www.researchgate.net/publication/323269587_Automated_Machine_Vision_System_for_Liquid_Particle_Inspection_of_Pharmaceutical_Injection
65. icwmc 2016 https://www.researchgate.net/profile/Carlos_Westphall/publication/310320592_ICWMC_2016_-_The_Twelfth_International_Conference_on_Wireless_and_Mobile_Communications/links/582b50f008ae102f07208a1d.pdf
66. Changelogs https://developers.cloudflare.com/changelog/
67. Accepted Workshops https://chi2026.acm.org/workshops/accepted/
68. Arxiv今日论文| 2026-02-02 http://lonepatient.top/2026/02/02/arxiv_papers_2026-02-02.html
69. 全球化研究报告-PDF版 https://bg.sgpjbg.com/baogaolist-0002100016-101.html
70. Pydantic Guardrails for LLM Pipelines https://www.linkedin.com/pulse/pydantic-guardrails-llm-pipelines-harnessing-cognitive-xiao-fei-zhang-vll6e
71. SoK: Runtime Integrity https://arxiv.org/html/2408.10200v2
72. A Zero-Trust Architecture for Unified AI Memory System https://arxiv.org/html/2601.07004v1
73. Arxiv今日论文| 2026-02-02 http://lonepatient.top/2026/02/02/arxiv_papers_2026-02-02

74. Proceedings of the 2025 Conference on Empirical Methods ... https://aclanthology.org/volumes/2025.emnlp-industry/
75. A Survey on Large Language Models for Code Generation https://dl.acm.org/doi/10.1145/3747588
76. Artificial Intelligence https://arxiv.org/list/cs.AI/new
77. Arxiv今日论文| 2026-01-14 http://lonepatient.top/2026/01/14/arxiv_papers_2026-01-14
78. Encapsulated Functions: Fortifying Rust's FFI in Embedded ... https://dl.acm.org/doi/pdf/10.1145/3625275.3625397
79. The Android Platform Security Model (2023) https://arxiv.org/html/1904.05572v3
80. (PDF) Silver Lining: Enforcing Secure Information Flow at ... https://www.researchgate.net/publication/286584008_Silver_Lining_Enforcing_Secure_Information_Flow_at_the_Cloud_Edge
81. Ensuring the Spatial and Temporal Memory Safety of C at ... https://www.researchgate.net/publication/220703630_MemSafe_Ensuring_the_Spatial_and_Temporal_Memory_Safety_of_C_at_Runtime
82. Did you know there's AI out there hiding secret backdoors? ... https://www.tiktok.com/@brianexplainstech/video/7557958155322363166
83. Windows Security best practices for integrating and ... https://www.microsoft.com/en-us/security/blog/2024/07/27/windows-security-best-practices-for-integrating-and-managing-security-tools/
84. Security Challenges and Defense Opportunities of Connected ... https://escholarship.org/content/qt1m70f36z/qt1m70f36z.pdf
85. Arxiv今日论文| 2026-01-09 http://lonepatient.top/2026/01/09/arxiv_papers_2026-01-09.html
86. Towards Enforcing Company Policy Adherence in Agentic ... https://arxiv.org/html/2507.16459v1
87. A Review of AI-Driven Automation Technologies https://www.sciencedirect.com/org/science/article/pii/S1546221825007416
88. Open Problems in Technical AI Governance https://openreview.net/notes/edits/attachment?id=NxxbJFpMJP&name=pdf
89. Proceedings of the Australasian Language Technology ... https://aclanthology.org/U03-1.pdf
90. H´et´erog´en´eit´e des Clients dans les Syst `emes d' ... https://theses.hal.science/tel-04685040v1/file/2024COAZ4029.pdf

91. Integrating Demand-Side Management and Forecasting https://www.researchgate.net/publication/337888472_Modelling_and_Optimisation_Planning_of_the_Dynamic_System_of_Energy_Supply_-_Integrating_Demand-Side_Management_and_Forecasting/fulltext/5df0abeb299bf10bc351cc50/Modelling-and-Optimisation-Planning-of-the-Dynamic-System-of-Energy-Supply-Integrating-Demand-Side-Management-and-Forecasting.pdf
92. Hospital Supply Chain: Fouad Jawab | PDF https://www.scribd.com/document/905971119/978-3-031-70292-1
93. Applications of Artificial Intelligence in Business, Education ... https://www.academia.edu/112889119/Applications_of_Artificial_Intelligence_in_Business_Education_and_Healthcare
94. Lista publicațiilor de la Think Tank-ul PE - European Parliament https://www.europarl.europa.eu/thinktank/ro/research/advanced-search/pdf?keywords=3231
95. Telefonica Tech · Blog https://telefonicatech.com/en/blog/author/telefonicatech