Proposal Summary (Submission Draft)

Base repository URL (replace if needed): - https://github.com/petroslamb/eth-llm-poc/blob/main

This document is the submission-ready proposal summary for the Ethereum Foundation RFP on integrating LLMs into protocol security research. It is grounded in a working system (PoC 5) and describes a low-risk, auditable path to full execution and consensus coverage in 4–6 months.

Executive Summary

We propose a deterministic, auditable LLM-assisted verification system that ingests Ethereum specifications, maps EIP obligations to spec and client code, and produces reproducible discrepancy reports. The system already exists as PoC 5 for the execution layer and runs in CI with structured artifacts. We intentionally avoided complex multi-agent architectures and RAG because they reduce reproducibility, increase failure modes, and complicate audit trails. Instead, we use direct chained phases with strict boundaries, a proven production agent, and versioned artifacts.

The plan scales to consensus specs and clients, adds cross-client validation, and introduces quality thresholds and dashboards as optional extensions. The outcome is a practical, scalable verification workflow that EF can run today and extend over time. The core risk is already retired because the pipeline is live; the remaining work is coverage expansion and operational hardening.

Key Differentiators

- **Working PoC today:** execution-specs pipeline already runs with reusable CI workflows and auditable artifacts.
- **Deterministic and auditable:** strict phase boundaries, versioned runs, and structured outputs minimize ambiguity.
- **Lowest-risk architecture:** avoided multi-agent and RAG complexity after evaluation; chosen approach is reproducible.
- **Clear scaling unit:** `eip-verify` job enables batch execution by EIP × client without architecture change.

---

1. RFP Objectives and How We Meet Them

These objectives are already validated by the working PoC. The table below maps each RFP requirement to concrete outputs or artifacts produced by the current pipeline.

RFP Objective	How We Address It	Evidence
Automated spec compliance	Extract obligations, map to spec + client code, produce gap reports	PoC 5 run artifacts + indexed CSVs
Workflow integration	Reusable GitHub Actions workflow with auditable artifacts	Reusable CI workflow + job manifests
Efficiency and accuracy	Deterministic phases, low false positives, optional LLM judge pass	Phase manifests + evaluation notes

---

2. Why This Approach (and Why Now)

Ethereum's spec and client surface is broad and changes frequently. The right solution is not the most complex one, but the most reproducible and auditable. After evaluating multi-agent systems, MCP/A2A protocols, RAG, symbolic repo maps, and bleeding-edge research methods, we found they introduced instability, unpredictable results, and unnecessary integration risk. The current PoC demonstrates that a simpler chained approach produces stronger accuracy and reliable artifacts. This is the lowest-risk path to a productionizable workflow that the EF can trust.

---

3. Scope Boundaries

In scope (Phase 1–4): - Execution-specs ingestion and EIP obligation extraction. - Execution clients (forked in repo) validated per EIP and per fork. - Consensus-specs ingestion and consensus client validation. - Deterministic artifacts, manifests, and summary reports.

Out of scope (unless explicitly requested): - Automatic code changes or patches. - Formal verification tooling beyond structured discrepancy reports. - Production deployment inside client release pipelines.

---

4. System Overview (Ingest → Analyze → Report)

Ingest: EIP markdown, execution-specs, and client repos.
Analyze: chained phases (extract → locate spec → analyze spec → locate client → analyze client).
Report: CSV indices, manifests, and summaries for auditability.

Output	Purpose
obligations_index.csv	Spec-side obligation mapping
client_obligations_index.csv	Client-side mapping and gaps
run_manifest.json	Per-phase metadata for audit
summary.md / summary.json	Human + machine readable reports

Design principles: simplicity, determinism, auditability, reproducibility.

---

5. Technical Approach (Methodologies, Frameworks, Tools)

Methodology: direct chained agent calls with strict phase boundaries and deterministic artifacts.
Rejected approaches: loosely coupled multi-agent systems, MCP/A2A protocols, RAG, symbolic repo maps, and bleeding-edge research methods due to complexity and poor reproducibility.
Frameworks/models: Claude Agent SDK with Opus 4.5 as primary, Sonnet 4.5 as fallback; evaluated GPT-5.2, Gemini 3 Pro, LangChain deep agent, and aider repomap.
Tools: native filesystem and CLI tools with structured outputs and manifest metadata.

---

6. Deliverables and Acceptance Criteria

Each deliverable has a clear acceptance criterion so EF can validate progress without ambiguity. Evidence is available in the repository and from PoC outputs.

Deliverable	Acceptance Criteria	Evidence
Technical architecture & design	Architecture + approach cover ingest, analysis, report, and toolchain	Architecture overview + system diagrams
Working prototype	PoC 5 runs per-EIP/per-client pipeline with artifacts	CLI pipeline + run outputs
Integration guidelines	Reusable workflow integration documented	Workflow usage + examples
Operations & extension	Setup, maintenance, and future phases documented	Ops guide + extension plan

7. Success Metrics (Initial Targets)

Targets are refined with EF in Phase 1. Initial targets reflect feasible, auditable outcomes for a 4–6 month roadmap.

Early PoC runs for EIP-1559 and EIP-2930 against the Geth client indicate the pipeline is robust and already meets the intended coverage and low false-positive goals for those example cases.

Metric	Initial Target	Evidence/Method
Coverage	100% of selected EIPs per fork mapped to spec obligations	CSV indices + manifests
Client validation	Per-EIP runs across agreed execution and consensus clients	Run manifests + summaries
Accuracy	<=5% false positives after judge pass on sampled runs	Cross-model review + audit logs
Reproducibility	100% artifact completeness per run	Manifests + structured outputs
Throughput	200 runs/month baseline; batch support by EIP × client	CI batch runs
Run time	Target <=60 minutes per CI run on baseline runners	CI run logs

8. Project Plan and Timeline (4–6 Months)

The plan expands coverage in a controlled way: first harden the pipeline, then scale across execution and consensus specs and clients, and finally add optional higher-level protocol security phases if desired.

Phase	Timing	Outputs
Phase 1	Month 1	Validate pipeline, harden phase separation, tighten prompts
Phase 2	Month 2	Execution clients matrix coverage, repeatable batch runs

Phase	Timing	Outputs
Phase 3	Month 3	Consensus-specs ingestion and obligation extraction
Phase 4	Month 4	Consensus clients matrix, EL/CL linkage
Phase 5 (optional)	Month 5	CI gating, quality thresholds, dashboarding
Phase 6 (optional)	Month 6	Extended phases for broader protocol security mapping

Detailed plan: see Supporting Materials (Project plan and timeline).

Optional extension goals (Month 5–6): - Cross-layer invariants and EL/CL consistency checks. - EIP drift detection across forks and client versions. - Higher-level protocol security mapping goals as prioritized with EF.

---

9. Evaluation Criteria (RFP)

The proposed system is assessed against the RFP criteria with evidence from the PoC and a clear path to expansion.

Criterion	PoC Evidence	Future Expansion
Scalability	Single EIP or batch; `eip-verify` unit scales across clients	Extend to consensus specs/clients and additional phases
Accuracy	Opus 4.5 yields minimal false positives; judge pass optional	Tune prompts, add evaluators, expand validation
Reliability	Simple chained pipeline with deterministic outputs	Add harnesses, logs, monitoring
Security	Runs inside CI; report-only outputs; minimal surface	Add tighter sandboxing as needed

---

10. Risks and Mitigations

The approach is intentionally conservative: fewer moving parts, clear audit trails, and a fallback path when model outputs are uncertain.

Risk	Mitigation
Spec ambiguity or EIP overlap	Obligation extraction with citations and audit trails
Model drift or regressions	Fixed prompts, versioned runs, optional judge pass
Cost variability	Explicit run budgeting, batch scheduling, model fallback
Client divergence	Per-client runs, EL/CL separation, artifact comparisons
Over-complexity	Avoid multi-agent/RAG to preserve determinism

---

11. Budget and Cost Structure (EUR)

We provide a cost model that separates experimentation runs from batch coverage runs, using published model pricing and a realistic CI baseline. The default estimate assumes 200 runs/month with 4M tokens/run and provides batch formulas for EIP × client coverage. Opus is used when higher reasoning fidelity is needed; Sonnet provides a cost-effective baseline for breadth runs.

Cost Item	Estimate
Solo delivery (4 months, discounted)	EUR 53,760
Solo delivery (6 months, discounted)	EUR 80,640
LLM runs (Opus, 200/month)	EUR 6,751.20/month
LLM runs (Sonnet, 200/month)	EUR 4,050.72/month
CI (Linux baseline, 200 runs)	EUR 81.01/month

Full cost model: see Supporting Materials (Budget and cost structure).
CSV breakdown: see Supporting Materials (Budget CSV).

---

12. Vendor Background

Petros Lambropoulos is an independent consultant focused on production-grade agentic systems, evaluation pipelines, and ML monitoring workflows. Prior roles include Senior Software Engineer on the NLP team at Workable and Senior

Software Engineer at NannyML. Consulting engagements include Google Cloud / ADK on agent systems for code intelligence and automation. This project also delivers the working PoC 5 pipeline included in the repository. References are available on request.

Docs: see Supporting Materials (Vendor background) and resume.

---

13. Assumptions and Dependencies

- Access to forked execution and consensus client repositories in the project.
- Ability to run GitHub Actions workflows for batch jobs.
- EF feedback cadence on scope selection and acceptance criteria.

---

14. Supporting Materials (Repository Links)

Append the folder and file below to the base repository URL above.

- Technical architecture
  Folder: poc5/docs/proposal/
  File: TECHNICAL_ARCHITECTURE_AND_DESIGN.md
- Project plan and timeline
  Folder: poc5/docs/proposal/
  File: PROJECT_PLAN_AND_TIMELINE.md
- Budget and cost structure
  Folder: poc5/docs/proposal/
  File: BUDGET_AND_COST_STRUCTURE.md
- Budget CSV
  Folder: poc5/docs/proposal/
  File: BUDGET_AND_COST_STRUCTURE.csv
- Integration guide
  Folder: poc5/docs/proposal/
  File: INTEGRATION_GUIDE.md
- Operations and extension
  Folder: poc5/docs/proposal/
  File: OPERATIONS_AND_EXTENSION.md
- Evaluation criteria response
  Folder: poc5/docs/proposal/
  File: EVALUATION_CRITERIA_RESPONSE.md
- Vendor background
  Folder: poc5/docs/proposal/
  File: VENDOR_BACKGROUND_AND_REFERENCES.md
- Proposal readiness checklist
  Folder: poc5/docs/proposal/
  File: PROPOSAL_READINESS_CHECKLIST.md

- Canonical RFP
  Folder: poc5/docs/proposal/
  File: Request for Proposal (RFP)_ Integrating Large Language Models (LLMs) into Ethereum Protocol Security Research.md