# Automating Lean-Based Formal Verification of Cryptographic Protocols with a Peer-Review-Gated AI Agent

Banri Yanahama[1]
[1]Nyx Foundation

Formal Verification, Cryptographic Protocols, Lean, AI Agents

## Abstract

This paper proposes a pipeline that uses an AI agent to automate Lean-based formal verification of cryptographic protocols.

The method adopts a two-stage gate design:

(i) generation of informal proofs in natural language;

(ii) an accept/reject decision by a reviewer role;

(iii) generation and verification of Lean proofs that formalize only the accepted drafts.

To the best of our knowledge, there has been no prior report in the public literature on the systematic application of Lean-generating AI agents to the domain of cryptographic protocols.

## Background

Formal verification is increasingly required for cryptographic protocols that underpin critical infrastructure. Compared with testing alone, it makes attacker models and security claims explicit and mechanically checkable. In this project we:

- target proofs in Lean, an interactive theorem prover with dependent types, tactics, and a rich mathematical library;
- leverage AI agents that iteratively reason, plan, and act across natural language and Lean code;
- focus on security proofs for cryptographic protocols, a domain where Lean-based AI agents have not yet been systematically studied.

Our goal is to build a gated automation pipeline that turns informal security arguments into Lean developments that can be checked and reproduced.

## Proposed Method

Figure 1 situates our setting within AI-assisted formal proof research (e.g., [1], [2], [3]).

| | Math | Cryptographic Protocol |
|---|---|---|
| Lean | • DeepSeek-Prover<br>• BFS-Prover<br>• Gödel-Prover<br>• Prover Agent<br>• Aristotle | • **This Proposal** |
| Other Proof Assistant | • GamePad<br>• CoqGym + ASTactic<br>• PALM<br>• IsarStep<br>• HOList | • CryptoFormalEval/ CryptoFormaLLM<br>• P2FGPT<br>• LLM-Aided Automatic Modeling |

Figure 1: Position of Our Work in AI-Assisted Formal Proof Research

And the pipeline starts from a target proposition, chooses a Focal Theorem, and incrementally introduces the smallest set of Focal Stub Lemmas needed to complete its proof.

- Three phases—Retrieval, Exploration, Conclusion—structure the workflow from collecting specifications to synchronizing verified Lean proofs.
- Within Exploration, rounds pass through the gates IW → IR → FW → FR → FS; the Focal Theorem and all Focal Lemmas always share the same state.

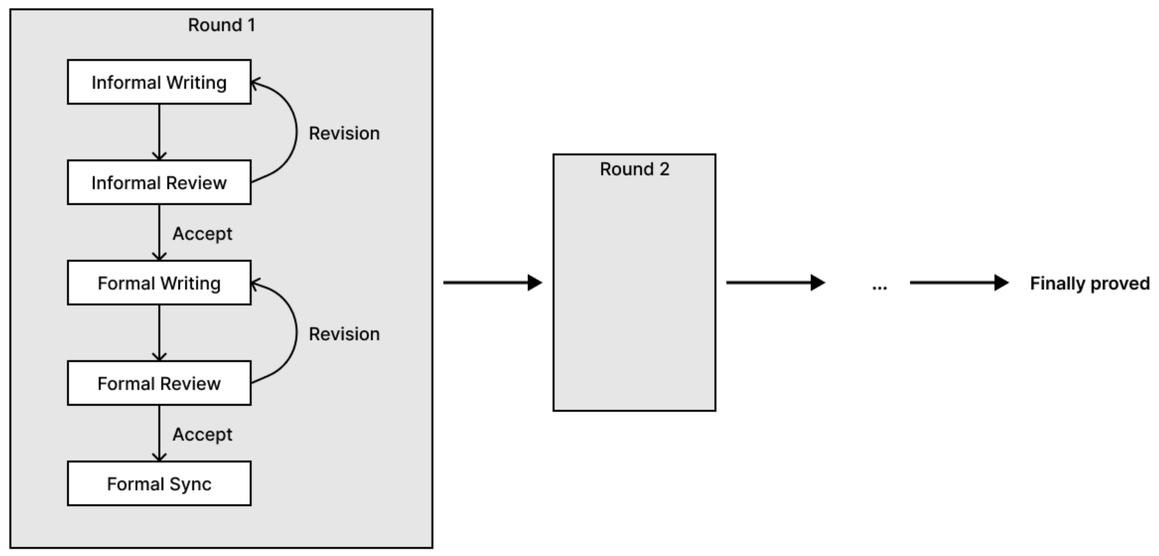


Figure 2: Overall Flow of the Proposed Pipeline

- A unified state model tracks each proposition as an informal or formal stub or artifact, while Lean imports define dependencies and the CLI orchestrates rounds via package scripts.

This design keeps the search focused, auditable, and convergent to Lean-checked security proofs.

## Result

We applied the pipeline to the security proposition for the Top Single Layer encoding, a building block of the quantum-resistant signature scheme XMSS. A run was counted as successful if it carried the proposition from an informal draft to a Lean development that passed all reviewer gates and reached the FS stage with Lean checking completed.

| | 1st time | 2nd time | 3rd time |
|---|---|---|---|
| Proof success or not | Success | Success | Success |
| Time taken (minutes) | 115 | 21 | 113 |
| Number of rounds | 4 | 1 | 4 |
| Number of turns | 22 | 5 | 22 |
| Num of new lemmas introduced | 3 | 0 | 3 |

Figure 3: Trial Results for Top Single Layer Encoding Verification

Overall, the agent consistently converged to final verification for this proposition.

## Discussion

Combining a two-stage gate design with focal decomposition yielded successful Lean verification in all three trials, both for a short path without new lemmas and for longer paths with additional lemmas. Future work includes:

- Apply the pipeline to additional protocols and properties (e.g., EUF-CMA for signatures, security proofs for zkSNARKs).
- Systematically ablate reviewer gates and their checklists to quantify their contribution to success, runtime, and rollback behavior.

## Key Reference

[1] T. A. et al., “Aristotle: IMO-level Automated Theorem Proving,” 2025.

[2] M. L. et al., “Proof Automation with Large Language Models,” 2024.

[3] C. C. et al., “CryptoFormalEval: Integrating LLMs and Formal Verification for Automated Cryptographic Protocol Vulnerability Detection,” 2024.

For further details: https://github.com/NyxFoundation/fv-agent-cans2025-poster