Signal Shot: End-to-End Formal Verification of Signal's Cryptographic Stack

Liao Zhang and the BAIF Team

Beneficial AI Foundation

BENEFICIAL AI FOUNDATION

Introduction

Signal

- security message sharing application
- over 70 million users
- Verification Signal's cryptography is crucial

Challenges in formal verification of Signal:

- The underlying mathematics of cryptography is complex and requires significant expertise to formalize
- Formal verification is inherently labor-intensive
- Ensuring long-term maintainability requires modularity and maintainable

Signal Shot addresses these challenges by leveraging:

- **Lean** as the proof assistant
- **Aeneas** (Rust-to-Lean transpiler) to establish a verifiable connection between formal security models and Signal's production codebase
- **AI integration** to imporve efficiency

Contributions

1. **Signal Shot**: Propose Signal Shot to addresses challenges in the formal verification of Signal.
2. **Large-Scale Verification**: Completed formal verification of the curve25519-dalek library, a widely deployed Rust implementation of Curve25519 operations
3. **AI Integration**: Developed specialized coding agents for formal verification
4. **Benchmarking**: Constructed a benchmark with ground-truth specifications and proofs for evaluating AI models on real-world program verification tasks

Technical Approach

The verification effort is organized along two complementary tracks:

Protocol Correctness:

- Formalize Signal's core cryptographic protocols in Lean: X3DH, PQXDH, Double Ratchet, and CKA
- Encode cryptographic papers into Lean
- Rigorously establish standard security properties

Function Correctness:

Verify that Signal's Rust implementations correctly realize developers' intentions:

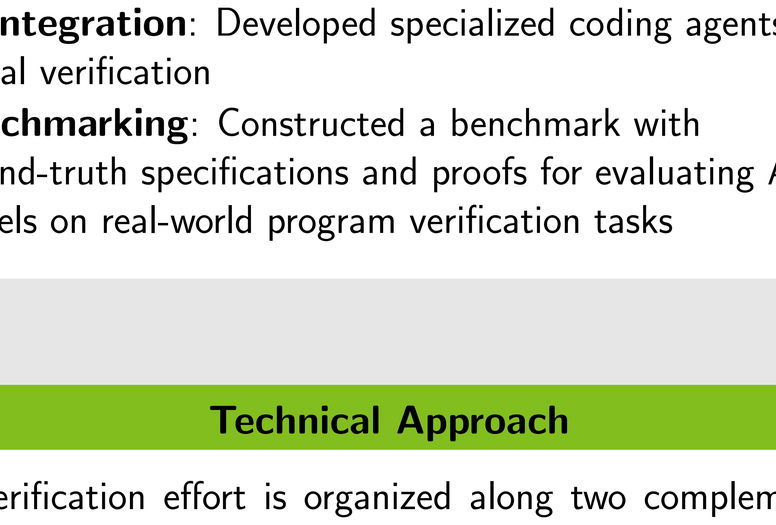


Figure 1: Lean Verification pipeline

Completed Verification of curve25519-dalek

We report the formal verification of **curve25519-dalek**, a core cryptographic library written in Rust that implements Curve25519 group operations.

Library Scope:

- **288 functions**, ∼**35,000 lines** of Rust code
- Used in Signal, Tor, and Transport Layer Security (TLS)
- **First** end-to-end formal verification of an elliptic-curve cryptographic library at this scale using Lean

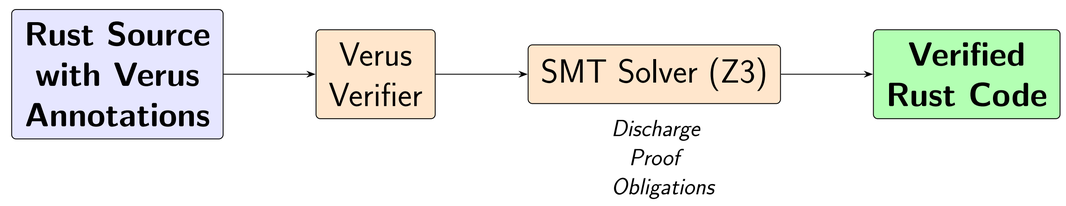


Figure 2: Verus verification workflow

Two-Team Verification Approach:

1. **Verus Team**: Annotated Rust source with compile-time Verus specifications, with proof obligations discharged by SMT solvers
2. **Lean Team**: Used Aeneas to extract semantically equivalent Lean representation and established its correctness

Technical Challenges Identified:

- Large proof states generated by Aeneas
- Instability in SMT solving

AI-Assisted Formal Verification

We significantly benefit from AI usage in the verification of curve25519-dalek and continue investigating how to effectively apply AI to formal verification using Lean.

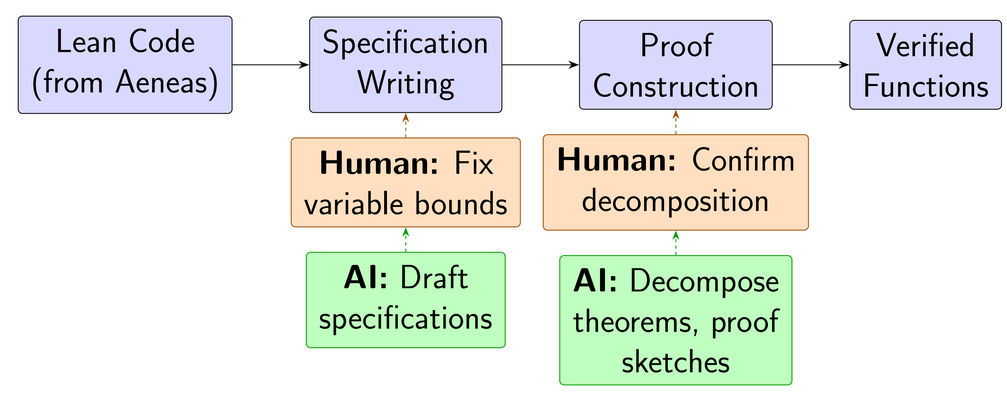


Figure 3: AI-assisted verification workflow

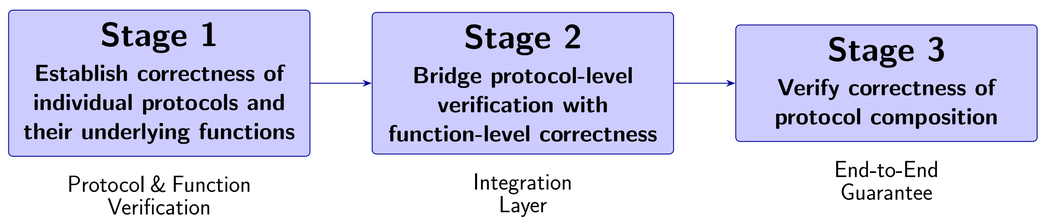
FVS: Formal Verification Skills

- Custom formal verification agents implemented as Claude Code skills
- Specialized techniques tailored to formal verification

Benchmark Construction

- Derived from our verified Lean codebase
- Provides ground-truth specifications and proofs
- Enables systematic evaluation of AI models across the verification pipeline: from specification synthesis to proof completion

Future Work



We're Hiring!

Join us at Beneficial AI Foundation

Scan for opportunities