CNVS Lean 4 Formal Verification Report

This document summarizes the formal verification activity performed on the CNVS (Closed Native Verification Systems) framework using Lean 4 and Mathlib.

AI-Assisted Formal Verification Statement

This research project employed artificial intelligence tools as part of the formalization workflow used to construct and refine Lean 4 verification modules.

In particular, AI-assisted code generation was used to:

- *draft Lean 4 formal structures;*
- *suggest Mathlib-compatible definitions;*
- *assist iterative debugging;*
- *support theorem-encoding workflows;*
- *accelerate formal verification prototyping.*

However, all generated Lean 4 modules were independently validated through:

- *Lean 4 kernel type checking;*
- *Mathlib compatibility verification;*
- *compilation-based proof validation.*

Only modules successfully accepted by the Lean 4 kernel with zero compilation errors were retained in the final verification corpus.

The use of artificial intelligence in this project should therefore be understood as:

- *AI-assisted formalization support,*
  not as:
- *autonomous mathematical proof certification.*

Formal correctness was ultimately determined exclusively by the Lean 4 proof kernel.

1. Objective

The objective of the verification campaign was to evaluate whether the CNVS theoretical framework could be encoded rigorously in Lean 4 without reducing the formal structure to trivial tautologies or purely symbolic identities.

The verification process progressively evolved from finite structural models toward probabilistic and asymptotic formulations involving:

- *Measure-theoretic probability spaces*
- *Conditional probabilities*
- *Dependent collusion models*
- *Binomial-tail probabilistic bounds*
- *Chernoff exponential bounds*
- *Asymptotic convergence*
- *Emergent security scaling*
- *Integrated probabilistic reconstruction models*

2. Verification Environment

Language: *Lean 4*

Libraries: *Mathlib*

Verification Mode: *Kernel-checked formal proofs and type verification*

Verification Outcome: *All final modules compiled successfully with zero errors*

3. Verified CNVS Modules

1. *Core Axioms*
2. *Axiom III*
3. *VLocal*
4. *Knowledge Restriction Principle*
5. *Knowledge Restriction Principle Asymptotic*
6. *Reconstruction Consistency*
7. *State Rejection*
8. *Information Density — Weighted Density*
9. *Probabilistic Security Core — Threshold Reconstruction*
10. *Target Attack Probability Bound*

11. *Binomial Tail Security Model*
12. *Binomial Tail Final Version (without temporary axiom)*
13. *Chernoff Security Bound*
14. *Emergent Security Scaling*
15. *Entropic Inference*
16. *Full Dependent-Collusion Probability Model*
17. *Dependent Collusion → Emergent Scaling Integration*
18. *Main Integration Theorem (abstract CNVS integration)*
19. *Theorem 17a.1 — Generalized Emergent Security Scaling under Dependent Collusion*

4. Main Formal Properties Tested

- *Consistency of the CNVS axiomatic layer*
- *Local/Global semantic separation*
- *Knowledge restriction under finite and asymptotic scaling*
- *Reconstruction consistency under non-uniform fragmentation*
- *Threshold reconstruction semantics*
- *Probabilistic attack bounds*
- *Binomial-tail reconstruction probability bounds*
- *Chernoff-type exponential security decay*
- *Emergent asymptotic security scaling*
- *Entropic adversarial inference monotonicity*
- *Dependent-collusion probabilistic bounds*
- *Measure-theoretic probability integration*
- *Integrated asymptotic reconstruction-decay theorem*

5. Technical Notes

The final CNVS verification framework no longer relies on simplistic tautological constructions of the form A → A. Instead, the verified modules use explicit mathematical structures including:

- *Finite combinatorics*
- *Binomial coefficients*
- *Real-valued inequalities*
- *Exponential bounds*
- *Filters and Tendsto convergence*
- *Conditional probabilities*
- *Probability spaces*
- *Dependent random-event structures*
- *Asymptotic scaling theorems*

6. Repository Structure

The repository contains Lean 4 source files and corresponding outcome reports for each verified module. The file names correspond to the verification tests performed during the formalization campaign.

- *CNVS code Lean 4 - Test axiom III*
- *CNVS code Lean 4 - Test axioms*
- *CNVS code Lean 4 - Test Binomial Tail No axiom*
- *CNVS code Lean 4 - Test Binomial Tail Security Model*
- *CNVS code Lean 4 - Test Chernoff Bound*
- *CNVS code Lean 4 - Test Core Axioms*
- *CNVS code Lean 4 - Test Dependent Collusion → Emergent Scaling Integration*
- *CNVS code Lean 4 - Test Emergent Security Scaling*
- *CNVS code Lean 4 - Test Entropic Inference*
- *CNVS code Lean 4 - Test Feasibility Constraint*
- *CNVS code Lean 4 - Test Full dependent-collusion probability model*
- *CNVS code Lean 4 - Test Information Density — weighted density*
- *CNVS code Lean 4 - Test Knowledge Restriction Principle*
- *CNVS code Lean 4 - Test Knowledge Restriction Principle Asymptotic*
- *CNVS code Lean 4 - Test Main Integration Theorem astratto CNVS*
- *CNVS code Lean 4 - Test Probabilistic Security Core — Threshold Reconstruction*
- *CNVS code Lean 4 - Test Reconstruction Consistency*
- *CNVS code Lean 4 - Test State Rejection*
- *CNVS code Lean 4 - Test Target Attack Probability Bound*
- *CNVS code Lean 4 - Test VLocal*
- *CNVS code Lean 4 - Theorem 17a.1 — Generalized Emergent Security Scaling under Dependent Collusion*

7. Final Status

The CNVS core framework has been formally prototyped, structurally integrated, and probabilistically modeled within Lean 4 using Mathlib.

The verification campaign demonstrates that the CNVS framework can be encoded as a mathematically coherent system involving probabilistic reconstruction bounds, dependent-collusion reasoning, and asymptotic security scaling.

8. GitHub Repository

https://github.com/massimocomitato-author/golden-protocol-nexus/tree/main/Code%20and%20Outcome%20CNVS%20on%20Lean%204.0

End Document

Author: Massimo Comitato

Italy, Milano (MI)

23 – 05- 2026