Lean-Verified Handshake: A Mechanized Commitment Kernel Against Coordination Failure Among Opaque AI Agents

Jan Ondras [1]

Abstract

Multi-agent AI systems in high-stakes infrastructure face a recurring failure mode: rational opaque agents may defect even when mutual cooperation is Pareto-optimal, typically when defection is risk-dominant. Program-equilibrium theory shows that binding commitment devices restore cooperation, but existing instantiations either require agents to read each other's source code or remain pen-and-paper; neither works for opaque LLM agents. We present *Lean-Verified Handshake*, a Lean 4-mechanized binding-attestation wrapper against this failure mode and a candidate verifier component for the *Guaranteed Safe AI* agenda. Routing all agents through the wrapper makes attesting Cooperate weakly dominate attesting Defect over any Stag-Hunt-ordered 2×2 game (symmetric or asymmetric) and any n-player counterpart with all-cooperator and all-defector endpoints, with mutual cooperation as the unique Pareto-undominated Nash equilibrium of the resulting meta-game. The wrapper's single trust assumption, that an external oracle publishes attestations truthfully, is isolated in a `TrustedOracle` typeclass, instantiated by a deterministic-chain oracle, with a lying-oracle counterexample showing the obligation is non-vacuous; TEE attestation and DIDs/VCs are sketched at the design level. Build-time `#print axioms` assertions confirm that every audited declaration (56 in total) depends only on Lean 4's standard axiom set (kernel+Classical) and fail the build on any drift. In a preliminary single-model 4-player LLM pilot, a deceptive broadcast cuts non-binding cheap-talk cooperation to $14/30$ trials while the wrapped arm preserves it in $30/30$, matching the mechanized dominance prediction.

[1]Massachusetts Institute of Technology, Cambridge, MA, USA. Correspondence to: Jan Ondras <janko@mit.edu>.

Workshop on Failure Modes of Agentic AI at the 43rd International Conference on Machine Learning (ICML), Seoul, South Korea, 2026. Copyright 2026 by the author(s).

1. Introduction

Multi-agent AI deployments such as coordinated traffic flow, distributed grid management, automated public-resource allocation, and multi-LLM orchestration pipelines put two or more autonomous decision-makers in a coordination problem whose Pareto-optimal outcome requires *cooperation* but whose risk-dominant outcome is *defection*. Even prompt-aligned large language model (LLM) agents can rationally defect when the sucker's payoff is steep and they cannot directly inspect the counterparty's intent (Lorè & Heydari, 2024; Madmoun & Lahlou, 2025).

Program-equilibrium theory (Tennenholtz, 2004; Kalai et al., 2010; Barász et al., 2014; Critch, 2019; Oesterheld, 2019) has long argued that binding *commitment devices* can restore cooperation: if each agent's strategy can condition on a verified property of the counterparty's strategy, mutual cooperation becomes an equilibrium outcome. The theory is mature, but every existing instantiation is either pen-and-paper (Tennenholtz, 2004; Kalai et al., 2010; Sauerberg & Oesterheld, 2026) or assumes adversarial source-code mutual reading (Barász et al., 2014; Critch, 2019). Neither route is open to opaque LLM agents, whose internal states cannot be inspected directly.

We take a third route: a thin *Lean-Verified Handshake* wrapper that mediates between $n \geq 2$ opaque agents through an external attestation surface. Each agent posts a binding attestation ("I will play C") before acting; the wrapper produces the action C iff every attestation agrees on C, otherwise D. The contribution is deliberately minimal: the resulting meta-game has C weakly dominant for every agent, with mutual cooperation as the unique Pareto-undominated Nash equilibrium.[1] We develop the headline 2-player case first (§3) and then extend to arbitrary n in §4.

[1]Any profile with at least two defectors, including all-D, is also a (weak) Nash equilibrium: no single deviation can flip the wrapper to C while another defector remains. All such profiles are Pareto-dominated by all-C, and the ε-incentive of Corollary 4.3 removes them as equilibria: all-C, already a strict equilibrium, becomes the unique one. That all-C is a Nash equilibrium is itself mechanized (`wrapN_weak_nash`); the weak-equilibrium remarks follow from the wrapper's binary collapse and are stated rather than separately mechanized.

A second design principle runs through the formalization: we treat the strategic combinator and the oracle-trust assumption as *separately reviewable* artifacts. The dominance proof is parameterized over arbitrary Stag-Hunt-ordered payoffs and oracle-agnostic by construction; the oracle threat model is discharged at the deployment surface (smart contracts, trusted execution environment (TEE) attestation, decentralized identifiers / verifiable credentials (DIDs/VCs)), not inside the proof.

Our contributions toward mitigating this failure mode are:

- **(C1) Mechanization.** To our knowledge, we give the first Lean 4 mechanization of a verifiable-commitment dominance argument (see §2), over arbitrary Stag-Hunt-ordered games (2-player and n-player, symmetric and asymmetric), `sorry`-free (no admitted proof gaps) and with no custom axioms, the 2-player results also re-derived as verified specializations of the n-player development; build-time axiom-hygiene assertions (`#guard_msgs in #print axioms`) turn axiom dependencies into regression tests, and the complete machine-checked development is openly available (Ondras, 2026).
- **(C2) Trust-assumption separation.** The oracle-truthfulness assumption is mechanized as a `TrustedOracle` typeclass with an explicit `truthful` predicate, kept disjoint from the dominance proof and instantiated in Lean by a worked deterministic-chain oracle; two further deployment surfaces (TEE attestation and DIDs/VCs) are sketched at the design level, with auditable trust profiles, so the strategic argument and the oracle's threat model can be reviewed separately.
- **(C3) Governance argument.** We argue that mechanized commitment kernels are an instance of the *Guaranteed Safe AI* agenda's verifier component (Dalrymple et al., 2024), and that the verification target for multi-agent deployments should expand from individual agents to the protocols connecting them. The result is a verified failure-mode mitigation: a systems-level intervention whose limitations and trust assumptions are stated explicitly.

2. Background and Related Work

Program equilibrium and commitment folk theorems. Tennenholtz (2004) showed that if agents can read each other's source code, the set of program-equilibrium payoffs equals the set of feasible and individually rational payoffs, including mutual cooperation in the Prisoner's Dilemma. Kalai et al. (2010) generalized this with a commitment folk theorem. Barász et al. (2014) formalized cooperation in the Prisoner's Dilemma using provability logic (FairBot, PrudentBot); those results remain in modal logic on paper, with no proof-assistant mechanization. Critch (2019) and Oesterheld (2019) refined this with bounded-resource and robust variants. The closest pen-and-paper predecessor to our work is Sauerberg & Oesterheld (2026): *ex post verifiable commitments* for safe Pareto improvements. We differ on three axes: (i) our wrapper is *ex ante* (each agent's attestation precedes the action) rather than verified after the fact; (ii) it is built for *opaque agent populations* where source-code mutual reading is infeasible; (iii) the dominance theorem is mechanized in Lean 4 with build-time axiom-hygiene assertions, not pen-and-paper. Our `wrap` combinator also differs from a cryptographic commitment scheme (which binds a *hidden* value for later opening) and from Byzantine-fault-tolerant consensus (which secures agreement among faulty replicas): we assume a single trusted attestation oracle, isolate that assumption in `TrustedOracle`, and defer a fault-tolerant quorum variant to future work (D2). A learned classifier or unverified middleware could target the same coordination outcome but supplies neither a machine-audited trust base nor a proved dominance guarantee.

Mechanized game theory. Bagnall et al. (2017) provide a Coq library for algorithmic game theory (Nash, smooth games, price-of-anarchy bounds); Le Roux et al. (2017) prove Nash existence in both Coq and Isabelle; Pomeret-Coquot et al. (2023) formalize incomplete-information games in Coq. None of these target commitment devices or strategic-incentive transformations. On the Lean 4 side, MixedMatched contributors (2024) formalize the Prisoner's Dilemma and Nash Demand, but no proof-assistant work has previously mechanized a commitment-device dominance theorem. The closest mechanism-design precedent is Caminati et al. (2015), who mechanize Vickrey auctions in Isabelle; their setting is mechanism design but not strategic-incentive transformation.

LLM coordination and binding contracts. Lorè & Heydari (2024) measure GPT-3.5/4 and Llama-2 in 2-player Stag Hunts across five contextual framings and find that cooperation rates vary substantially with the framing: pronounced in pro-social contexts (e.g., the "Friends' Chat" framing), materially lower in adversarial ones. The finding is empirical and both model- and context-specific; it is not a guarantee that survives a change of model, population size, or unobserved framings. Madmoun & Lahlou (2025) observe that one-word *cheap talk* (a non-binding broadcast) lifts cooperation in 4-player Stag Hunts from 0% to 96.7% across a four-model open-weights cohort (Mixtral-8x22B, Qwen2.5-72B, Llama-3.3-70B, DeepSeek-V3), but offers no enforcement mechanism. Haupt et al. (2024) show that binding contracts mitigate social dilemmas in multi-agent reinforcement learning, but with heuristic learned policies rather than a formally verified kernel, and without addressing opaque foundation-model agents.

We contribute a small, mechanized commitment kernel that opaque agents can route through, with the dominance theorem proved in Lean. The kernel instantiates the verifier component of the *Guaranteed Safe AI* agenda (Dalrymple et al., 2024) for one specific safety property (induced cooperation under a binding commitment), leaving the world-model and safety-spec components implicit in the deployment.

3. The Lean-Verified Handshake

Game. We work over symmetric 2×2 games with action set $\{C, D\}$ (Cooperate / Defect) and payoff matrix entries $g_{cc}, g_{cd}, g_{dc}, g_{dd} \in \mathbb{Q}$, where g_{ab} is the row player's payoff when row plays a and column plays b. We focus on the *Stag-Hunt* family characterized by the ordering $g_{cd} < g_{dd} < g_{dc} < g_{cc}$: mutual cooperation is socially best; mutual defection is the safe fallback; the sucker's payoff (cooperate while the counterparty defects) is the worst outcome. The ordering is ordinal; defection being additionally *risk-dominant* is a cardinal property typical of the motivating deployments but not required by the mechanized results.

Asymmetric generalization. The formalization also covers *asymmetric Stag-Hunt* games, in which row and column may have distinct payoff matrices, each independently Stag-Hunt-ordered. We model this in Lean as an `AsymGame` structure carrying two `SymGame` records (row and column payoffs) and prove player-indexed counterparts of Theorems 4.1, 4.4, and Corollary 4.2 in `Asymmetric.lean`; a `SymGame.toAsym` coercion exhibits the symmetric development as the diagonal subcase (row and column sharing one payoff matrix). The mechanization is uniform precisely because each player's dominance argument depends only on that player's own Stag-Hunt chain, with no cross-player coupling in the proof body; per-player mirroring is therefore sound.

Wrapper. The handshake wrapper takes two attestations $a_A, a_B \in \{C, D\}$ (one per agent) and emits an action:

$$\mathrm{wrap}(a_A, a_B) = \begin{cases} C & \text{if } a_A = C \wedge a_B = C, \\ D & \text{otherwise.} \end{cases}$$

Both agents pass their attestations through wrap to obtain their physical actions. Agent A's action is $\mathrm{wrap}(a_A, a_B)$; agent B's action is $\mathrm{wrap}(a_B, a_A)$. Cooperation requires bilateral attestation; any attestation profile other than (C, C) collapses to mutual defection. By Lemma `wrap_comm`, the wrapper is symmetric in its arguments, so both agents always observe the same wrapped action and the unconstrained game's asymmetric off-diagonal outcomes g_{cd}/g_{dc} are unreachable: the wrapper collapses those cells to g_{dd} (Table 1, right).

Table 1 compares the unconstrained game with the wrapped meta-game in the row player's payoff. Attesting C weakly dominates attesting D everywhere and strictly dominates when the counterparty attests C. The risk comparison also flips in favor of attesting C: a deviation from all-C forfeits $g_{cc} - g_{dd} > 0$, while a deviation from all-D forfeits nothing, so the original game's risk-dominant defection does not survive the transformation. A public bonus $\varepsilon > 0$ for attesting C closes the remaining tie when the counterparty attests D, rendering dominance strict everywhere (Corollary 4.3; deployment instantiations of ε are discussed under (L2) in §4).

Table 1. Row player's payoff g_{ab} in the Stag-Hunt game (left) and in the wrapped meta-game where both agents pass attestations a_A, a_B through the handshake (right). The wrapper collapses the off-diagonal cells to g_{dd}: any non-(C, C) profile yields the safe payoff (shaded).

Unconstrained		
	$b = C$	$b = D$
$a = C$	g_{cc}	g_{cd}
$a = D$	g_{dc}	g_{dd}

Wrapped		
	$a_B = C$	$a_B = D$
$a_A = C$	g_{cc}	g_{dd}
$a_A = D$	g_{dd}	g_{dd}

Trust model. The wrapper presupposes a *trusted attestation surface*: an external mechanism that publishes each agent's attestation truthfully and unforgeably to the other agents. We do *not* model this oracle as a Lean axiom. Instead, the trust hypothesis is mechanized as the `TrustedOracle` typeclass shown in Listing 1: a deployment instantiates the typeclass by exhibiting an attestation function and a truthfulness witness (the latter relating the attested action to the agent's underlying intent, i.e., the action the agent itself submits, not an inspected internal state), after which the agent-level dominance theorems (`wrap_weakly_dominates_agent`, `wrap_pareto_unique_agent`, and the capstone realization Proposition 4.5) apply. Three realizations bound the trust assumption: (i) *smart contracts*, where the chain executes the wrapper deterministically; (ii) *TEE remote attestation*, where each agent's wrapper invocation runs inside an Intel SGX/TDX enclave producing a remote-attestation quote consumed by the counterparty; (iii) *DIDs/VCs*, where attestations are signed verifiable credentials anchored to a decentralized identifier registry. Each shifts the trust burden differently (to the chain validators, the hardware vendor, or the credential ecosystem), but each instantiates the abstract type the Lean wrapper consumes. The artifact ships a worked Lean instance for the simplest of these surfaces (the `deterministicChainOracle` instance over a toy transaction model where `attest` forwards each transaction's intended action). The dominance theorem is oracle-agnostic through the `TrustedOracle` typeclass: a future TEE or DID/VC instance inherits the verified guarantee

without re-proving the strategic core, provided it discharges the `truthful` obligation and the deployment executes the wrapper faithfully (for smart contracts and TEEs, attester and executor coincide). Supplying the truthfulness witness is the non-trivial, backend-specific burden; for TEE attestation and DID/VC backends we sketch it but do not yet mechanize it. Figure 1 shows the deployment architecture.

Listing 1. The mechanized trust-model typeclass. A Lean typeclass is an interface a deployment must implement: here, an `attest` function (mapping each agent to its attestation) and a `truthful` witness (a proof that the attested action equals the agent's intent). Agent-level dominance theorems take agents as inputs, with the typeclass supplying their attestations; the typeclass is agent-count-agnostic, lifting unchanged to the n-player setting (Theorem 4.6).

```
class TrustedOracle (Agent : Type)
    (intent : Agent → Action) where
  attest : Agent → Attestation
  truthful :
    ∀ a, (attest a).action = intent a
```

4. Mechanized Theorem and Axiom-Footprint Audit

In Lean 4 (de Moura & Ullrich, 2021) we mechanize: the wrapper; a small commutativity lemma (`wrap_comm`: $\mathrm{wrap}(a,b) = \mathrm{wrap}(b,a)$, by case analysis on the two arguments) and its n-player counterpart (`wrapN_comm`: $\mathrm{wrapN}(\mathrm{atts} \circ \sigma) = \mathrm{wrapN}(\mathrm{atts})$ for any $\sigma : \mathrm{Equiv.Perm}(\mathrm{Fin}\, n)$); the Stag-Hunt-ordering predicate; and the three main 2-player results (Theorems 4.1 and 4.4 together with Corollary 4.2). We use mathlib (The mathlib Community, 2020), Lean's community mathematics library, only for rational arithmetic, the linear-arithmetic tactic `linarith`, and finite-type support for the n-player development. The Stag-Hunt predicate is carried as a `Prop`-valued structure with three named fields, exposing the chain $g_{cd} < g_{dd} < g_{dc} < g_{cc}$ to `linarith`. The wrapper's commutativity makes the column player's weak-dominance claim coincide with the row player's; no separate column-player theorem is needed.

Theorem 4.1 (Weak dominance)**.** *For any `SymGame` g satisfying `StagHuntOrdering`, and any opponent attestation $oppAtt \in \{C, D\}$,*

$$\mathrm{payoff}(g, \mathrm{wrap}(C, oppAtt), \mathrm{wrap}(oppAtt, C)) \geq \mathrm{payoff}(g, \mathrm{wrap}(D, oppAtt), \mathrm{wrap}(oppAtt, D)).$$

Corollary 4.2 (Strict dominance under cooperation)**.** *When the opponent attests C, attesting C strictly beats attesting D:*

$$\mathrm{payoff}(g, \mathrm{wrap}(C, C), \mathrm{wrap}(C, C)) > \mathrm{payoff}(g, \mathrm{wrap}(D, C), \mathrm{wrap}(C, D)).$$

Corollary 4.3 (Strict dominance under ε-incentive)**.** *Let $\varepsilon > 0$ be a public per-round bonus added to the payoff of any agent who attests C (e.g., a regulatory subsidy, gas rebate, or reputation reward), independent of which wrapped action is realized. Then for any opponent attestation $oppAtt \in \{C, D\}$, attesting C* strictly *dominates attesting D in any Stag-Hunt-ordered game, including when the opponent does not commit ($oppAtt = D$), where weak dominance returns only equality:*

$$\mathrm{payoff}(g, \mathrm{wrap}(C, oppAtt), \mathrm{wrap}(oppAtt, C)) + \varepsilon > \mathrm{payoff}(g, \mathrm{wrap}(D, oppAtt), \mathrm{wrap}(oppAtt, D)).$$

Mechanized as `wrap_strictly_dominates_eps` (2-player); the n-player, asymmetric generalization is `wrapN_strictly_dominates_asym_eps`, whose `TrustedOracle` agent-level lift is `wrapN_strictly_dominates_asym_eps_agent`.

Theorem 4.4 (Pareto-uniqueness)**.** *For any attestation profile $(a, b) \neq (C, C)$,*

$$\mathrm{payoff}(g, \mathrm{wrap}(C, C), \mathrm{wrap}(C, C)) > \mathrm{payoff}(g, \mathrm{wrap}(a, b), \mathrm{wrap}(b, a)).$$

Proposition 4.5 (Cooperation realization)**.** *For any `TrustedOracle` instance and any agents a, b with $\mathrm{intent}(a) = \mathrm{intent}(b) = C$,*

$$\mathrm{wrap}(\mathrm{attest}(a), \mathrm{attest}(b)) = C.$$

The proofs are short. Each dominance result reduces, after the wrapper's binary collapse, to a case split closed by `linarith` over a four-element payoff chain, a uniform template across the 2-player, n-player, symmetric, and asymmetric variants; the contribution is mechanization fidelity, not proof depth. The non-uniform work sits in the supporting layer (permutation invariance of `wrapN`, the machine-checked specialization of the 2-player results from the n-player ones, the `Finset`-cardinality coercions behind the asymmetric lift). The fidelity rests on parameterization over `StagHuntOrdering`, the typeclass-level separation of the trust assumption from the dominance proof (§3), and the build-time `#guard_msgs` assertions described below. Theorem 4.1 follows by case analysis on the action carried by $oppAtt$. In the $oppAtt{=}C$ branch, both sides reduce to g_{cc} and g_{dd} respectively, and the chain $g_{dd} < g_{dc} < g_{cc}$ closes the goal via `linarith`. In the $oppAtt{=}D$ branch, both sides collapse to g_{dd}. Corollary 4.2 sharpens the $oppAtt{=}C$ branch to strict inequality (with Theorem 4.1, this is weak dominance in the standard sense: never worse, strictly better somewhere). Theorem 4.4 cases on a and b; the excluded (C, C) case is contradicted by hypothesis, the others reduce to $g_{cc} > g_{dd}$. Proposition 4.5 closes by structural rewrite from the `truthful` field, as does its n-player counterpart

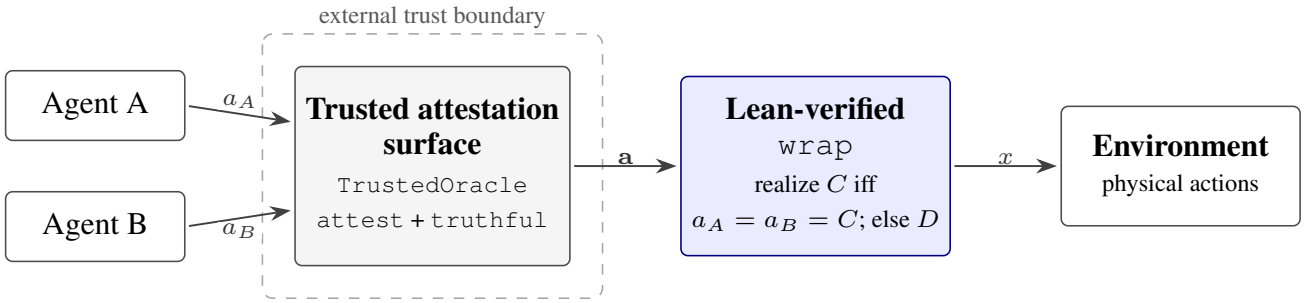

Figure 1. Architecture of the verified handshake. Participating agents submit attestations to an external trusted attestation surface, formalized by `TrustedOracle`; the dashed box marks the external trust boundary, a deployment-surface assumption, not a Lean axiom. The Lean-verified `wrap` combinator realizes C iff both attestations are C and realizes D otherwise; the realized action $x \in \{C, D\}$ is played in the environment. Shown for $n = 2$; for arbitrary n, the profile $\mathbf{a} = (a_i)_{i=1}^{n}$ is routed to `wrapN`, which realizes C iff all $a_i = C$.

`cooperate_realized_n_truthful`; these two realization results are consequently axiom-free in the audit below. The same case-split-then-`linarith` template carries to the asymmetric counterparts in `Asymmetric.lean` (`wrap_*_asym_{row,col}`). Listing 2 shows the central theorem.

n-player generalization. The same template extends to $n \geq 2$ players. The wrapper `wrapN` outputs C iff every attestation carries C, so the wrapped meta-game collapses to two reachable outcomes, reducing the dominance argument to their two payoffs: the all-C payoff $g.u\,C\,(n{-}1)$ (the top of the four-element chain $g.u\,C\,0 < g.u\,D\,0 < g.u\,D\,(n{-}1) < g.u\,C\,(n{-}1)$ encoded in `StagHuntOrderingN`) and the all-D payoff $g.u\,D\,0$ (the chain's second element). We formalize this as `SymGameN n`, state weak dominance as Theorem 4.6 and Pareto-uniqueness as Theorem 4.7, and prove the truthful-oracle result `cooperate_realized_n_truthful` (axiom-free); details are in Appendix A. The asymmetric case mirrors the per-player framework formalized in `AsymStagHuntOrderingN`, with `wrapN_weakly_dominates_asym` proved analogously.

Theorem 4.6 (Weak dominance, n-player)**.** *For any $n \geq 2$, any `SymGameN` g satisfying `StagHuntOrderingN`, any player i, and any opponents' attestations a_{-i},*

$$\begin{aligned}&\mathrm{payoffN}_g\big(\mathrm{wrapN}(a_{-i}, C)\big)\\ &\quad \geq \mathrm{payoffN}_g\big(\mathrm{wrapN}(a_{-i}, D)\big).\end{aligned}$$

Theorem 4.7 (Pareto-uniqueness, n-player)**.** *For any attestation profile $a \neq (C, \ldots, C)$,*

$$\begin{aligned}&\mathrm{payoffN}_g(\mathrm{wrapN}((C, \ldots, C)))\\ &\quad > \mathrm{payoffN}_g(\mathrm{wrapN}(a)).\end{aligned}$$

Unification. The 2-player development (Theorems 4.1, 4.4, Corollary 4.2, and their asymmetric mirrors) is formally re-derivable as a specialization of the n-player development at $n = 2$: diagonal lifts (`SymGame.toSymGameN`, `AsymGame.toAsymGameN`) embed the 2-player game types, preservation lemmas carry the Stag-Hunt chain through, and bridge lemmas (`payoffN_two_eq_payoff` and its asymmetric analogues) collapse the n-player wrapped payoffs to their 2-player forms, so the `wrap_*_via_n` corollaries (three symmetric, six asymmetric) obtain the 2-player theorems by instantiating their n-player counterparts at $n = 2$. The parallel APIs are thus connected by machine-checked specialization on both the symmetric and asymmetric branches.

4.1. Axiom Hygiene as a Regression Test

`#print axioms` (which lists a declaration's axiom dependencies) and `#guard_msgs` (which fails the build unless a command's output matches an expected string) are both standard Lean facilities. Our contribution is to compose them into a *per-declaration regression test*: after each theorem we assert its expected foundational-axiom footprint with `#guard_msgs in #print axioms`, so any drift fails the build with a mismatch (`Audit.lean`; full source at (Ondras, 2026)). The development is small: its 2-player core imports *mathlib* only for $\mathbb{Q}$-arithmetic and `linarith`, with the n-player and asymmetric layers additionally drawing on finite-type support (`Fintype`, `Finset`, `Fin`) and the `norm_num`/`fin_cases` tactics; it has *no* `sorry`s and *no* custom axioms, builds in $\sim$30 s on a warm cache (Lean 4 `v4.29.1`, *mathlib* `v4.29.1`), and gates its trust base in CI. Each of the 56 audited declarations depends only on a subset of Lean 4's foundational set `[propext, Classical.choice, Quot.sound]` (kernel+Classical), Lean's standard classical foundations, so the development adds no assumptions of its own; Table 2 partitions them by footprint. Most declarations lie at the extremes: characterization lemmas such as `wrapN_eq_C_iff` and the truthful-oracle results `cooperate_realized_*` are axiom-free (pure

Listing 2. The mechanized weak-dominance theorem. The proof has two cases on the opponent's attestation: when it is `C`, both wrapped sides reduce to `g.cc` and `g.dd`, and the goal $g_{cc} \geq g_{dd}$ follows from the Stag-Hunt chain $g_{dd} < g_{dc} < g_{cc}$; when it is `D`, both sides collapse to `g.dd`. In Lean: `obtain` unpacks the attestation; `cases` splits on the underlying action; `simp` unfolds `wrap` and `payoff`; `linarith` closes via the chain.

```
theorem wrap_weakly_dominates
    (g : SymGame) (h : StagHuntOrdering g) (oppAtt : Attestation) :
    payoff g (wrap ⟨.C⟩ oppAtt) (wrap oppAtt ⟨.C⟩)
      ≥ payoff g (wrap ⟨.D⟩ oppAtt) (wrap oppAtt ⟨.D⟩) := by
  obtain ⟨a⟩ := oppAtt
  cases a
  · -- a = .C: g.cc ≥ g.dd via g.dd < g.dc < g.cc
    simp [wrap, payoff]
    linarith [h.dd_lt_dc, h.dc_lt_cc]
  · -- a = .D: g.dd ≥ g.dd
    simp [wrap, payoff]
```

Table 2. Foundational-axiom footprint of all 56 audited declarations, summarizing the build-time `#guard_msgs in #print axioms` guards in `Audit.lean`. For the complete per-declaration breakdown, see Table 3.

Axiom footprint	**# declarations**
axiom-free	8
`propext` only	5
`Quot.sound` only	1
`propext` + `Quot.sound`	1
`propext` + `Classical.choice` + `Quot.sound`	41
Total	**56**

computation or structural rewrite), while results invoking `simp`/`linarith` pull the full kernel+Classical set; only a handful fall in between (`propext` from `Iff` rewrites, `Quot.sound` from the `Equiv.Perm`-based permutation lemma). The full per-declaration breakdown is in Appendix B.

4.2. Limitations (L1–L8)

We record eight explicit limitations. **(L1)** The cooperation-realization guarantee relies on the `truthful` field of `TrustedOracle`: a non-truthful witness collapses it. The artifact ships `deterministicChainOracle`, a `TrustedOracle` instance over a richer agent type (`Transaction`, carrying a sender alongside the intended action) with truthfulness discharged by structure projection on the assumption that the chain forwards intent faithfully, alongside a `lyingAttest` non-instance (`lyingAttest_not_truthful`) witnessing that the obligation is non-vacuous. In scope is exactly one obligation: a single oracle that attests truthfully and unforgeably (at this abstraction both dishonesty and forgery surface as violations of the one mechanized `truthful` predicate); out of scope, deferred to the deployment surface, are spoofed or false attestations, wrapper bypass, replay, cross-counterparty equivocation, signer collusion, compromised TEEs, validator capture, Sybil attacks, and post-attestation policy changes (TEE and DID/VC instances and adversarial-oracle robustness are likewise unmodeled). Any of these voids the guarantee; the dominance ordering is proved oracle-agnostically. **(L2)** Strict dominance follows from the weak result via an external ε-incentive (Corollary 4.3): a public bonus $\varepsilon > 0$ for attesting C closes the tie when the opponent does not commit. Instantiations of ε (for example, a regulatory subsidy) are governance-layer choices deferred to deployment, and the strategic core of the wrapper remains independent of ε's source. Tremble-robust strict variants *without* external incentive (cf. the ex post verifiable commitments of Sauerberg & Oesterheld, 2026) remain follow-up work. **(L3)** The wrapper's binary collapse to all-C / all-D means the mechanized theorems cover partial-attestation profiles ($k < n$ players commit) uniformly but trivially: any single defection forces the all-defector payoff regardless of the other attestations. A richer payoff structure supporting *threshold cooperation* (intermediate counts giving intermediate payoffs) would make partial adoption strategically meaningful and is deferred; see (D1) for follow-up. **(L4)** Strategies are pure; mixed-strategy and behavioral-strategy variants are not yet mechanized. **(L5)** The dominance theorem assumes all participating agents route through the wrapper. Adoption itself is not an equilibrium concept of the model: under purely voluntary adoption, the choice to wrap is itself a coordination problem. We treat the closure of this meta-game as a deployment/governance concern (§5), not a formal-verification concern. **(L6)** The asymmetric extensions (`Asymmetric.lean` for the 2-player case;

`NPlayerAsym.lean` for the n-player case) require *every* player's payoff structure to be Stag-Hunt-ordered at the relevant endpoint profiles; hybrid-ordering games (e.g., one player Stag-Hunt while another follows Prisoner's Dilemma) are out of scope. **(L7)** Each wrapper invocation requires one attestation per agent through the chosen backend, so per-round cost scales with the number of agents and the oracle's per-attestation latency (chain consensus for smart contracts, remote-attestation handshake for TEEs, credential resolution for DIDs/VCs), aggregating across rounds; these backend-specific factors lie outside our single-shot formal model. **(L8)** The three oracle backends also differ on revocation and jurisdictional exposure: smart contracts have no native revocation, TEEs revoke per enclave instance, and DIDs/VCs revoke per credential, with jurisdiction following the chain, the hardware vendor, or the credential issuer respectively. Backend selection on these axes is a deployment-layer choice.

5. Discussion: Failure Mode, Adoption, and Governance

The Lean-Verified Handshake is not a general AI-alignment solution but a narrower, more deployable primitive: a small, formally verified *commitment kernel* that, when interposed between opaque AI agents whose strategic incentives can be altered by binding commitment, makes mutual cooperation the unique Pareto-undominated Nash equilibrium of the resulting meta-game. The mechanism is independent of agent internals; it does not require the agents to share an alignment story, model class, or training process. The verified piece is the strategic combinator only; the surrounding middleware (oracle, transport, signing) and its trust profile are deferred to the chosen deployment surface.

The wrapper presupposes that all participating agents have *adopted* it. Under purely voluntary adoption, this decision is itself a coordination problem: full adoption yields g_{cc}; partial or unilateral adoption falls back to the original game. Our mechanism therefore relocates coordination from per-action play to a one-time, publicly verifiable adoption decision; the residual adoption problem is simpler than per-action coordination but is not eliminated. Closing the residual gap is the role of governance: a regulatory mandate, an environmental requirement, or market exclusion of non-adopters can each render full adoption the dominant strategy. Within this governance frame, the verified commitment kernel is a candidate verifier component for the *Guaranteed Safe AI* agenda (C3).

For the deployment scenarios sketched in §1, mechanized commitment kernels are a candidate building block for protocol-level verification that expands beyond individual agents. Framed this way, the verified handshake is a mitigation for one specific multi-agent failure mode: a per-step commitment primitive with documented limitations (L1–L8) and an explicit trust boundary.

Preliminary empirical evidence: deceptive broadcast. The n-player dominance prediction (Theorem 4.6) is falsifiable in deployment: binding attestation should preserve cooperation under adversarial perturbations that defeat non-binding signals. A 30-trial-per-cell pilot in a 4-player Stag Hunt with Gemini 3.1 Flash-Lite (single model in all four roles) tests exactly this under a *deceptive broadcast* framing, where one agent is instructed to defect while inducing others to cooperate. Following Madmoun & Lahlou (2025), the cheap-talk arm has agents broadcast a one-word signal and then choose an action independently. In the benign cells, cheap talk and wrap both reach 100% full cooperation, comparable to the 96.7% Madmoun & Lahlou benchmark; we do not claim a replication, given design differences (single-model vs heterogeneous-cohort; single-shot vs 3-round; payoffs $(10, 0, 8, 5)$ vs $(10, 0, 3, 3)$). Under deceptive broadcast with neutral prompts (factual rules-of-the-game, no strategic hints), cheap talk degrades: $14/30$ trials reach full cooperation, with the deceiver broadcasting Stag (C) and playing Hare (D) in $16/30$ cheap-talk trials. The wrapper preserves $30/30$ all-C trials (Wilson 95% CI $[0.89, 1.00]$; Fisher's exact $p \approx 2{\times}10^{-6}$ vs cheap-talk), with the deceiver attesting C in every trial, its stated reasoning consistent with the dominance prediction. As a robustness check, the deceptive-cell wrap result is unchanged between an arm-aware prompt that explains the binding rule and a neutral prompt that does not ($30/30$ under both styles; no detectable prompt-mediation effect at this sample size), supporting the conclusion that the deceptive wrap finding is not a prompt artifact. The pilot is single-shot and single-model. The strategic guarantees themselves hold for any oracle backend that satisfies the `TrustedOracle` obligation, whether smart-contract, TEE, or DID/VC.

Open follow-up directions. (D1) Partial-attestation thresholds: when only k of n agents commit, the binary collapse to all-C/all-D no longer applies; a richer payoff function would yield a *threshold cooperation* theorem (cooperate when the committer count k reaches a threshold k^*), extending the count-indexed `SymGameN n` already in the development, the one extension that yields a non-trivial inequality. **(D2) Byzantine attestation oracles**: the `TrustedOracle` typeclass extends to a quorum oracle tolerating up to f Byzantine signers (e.g., under the classical $n \geq 3f{+}1$ bound), building on the present `deterministicChainOracle`; no weaker-than-truthful guarantee is presently mechanized. **(D3) Broader empirical evaluation**: heterogeneous-cohort replication, multi-round play, and replication at Madmoun & Lahlou's $(10, 0, 3, 3)$ payoffs, whose constant hare payoff ($g_{dc} = g_{dd}$) first requires relaxing the mechanized ordering's middle inequality to $\leq$.

Outlook. Canonical successes of formal verification include *individual* artifacts such as compilers (Leroy, 2009), operating-system kernels (Klein et al., 2009), and cryptographic primitives (Zinzindohoué et al., 2017). We argue that in multi-agent AI, the protocols connecting agents deserve analogous scrutiny.

Software and Data

The Lean 4 source and all machine-checked proofs, including the axiom-footprint audit (`Audit.lean`), are available at https://github.com/jancio/VerifiedHandshake.

Impact Statement

This paper develops a formally verified primitive for safer multi-agent AI cooperation. Infrastructure that coordinates safety-critical AI agents (traffic flow, grid management, multi-LLM pipelines) would gain a verified component whose strategic guarantees are independent of agent internals and whose trust assumptions are auditable separately from the strategic argument. Binding commitment devices can, however, entrench power asymmetries when the trust surface is captured by a small set of validators, and the kernel does not address fairness, consent, or representation in agent populations. The typeclass-level discharge obligation we introduce is intended to make the trust profile of any deployment legible to auditors and regulators, rather than to advance any particular deployment surface as universally trustworthy.

References

Bagnall, A., Merten, S., and Stewart, G. A library for algorithmic game theory in Ssreflect/Coq. *Journal of Formalized Reasoning*, 10(1):67–95, 2017. doi: 10.6092/issn.1972-5787/7235.

Barász, M., Christiano, P., Fallenstein, B., Herreshoff, M., LaVictoire, P., and Yudkowsky, E. Robust cooperation in the prisoner's dilemma: Program equilibrium via provability logic. *arXiv preprint arXiv:1401.5577*, 2014.

Caminati, M. B., Kerber, M., Lange, C., and Rowat, C. Sound auction specification and implementation. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation (EC '15)*, pp. 547–564, 2015. doi: 10.1145/2764468.2764511.

Critch, A. A parametric, resource-bounded generalization of Löb's theorem, and a robust cooperation criterion for open-source game theory. *The Journal of Symbolic Logic*, 84(4):1368–1381, 2019. doi: 10.1017/jsl.2017.42.

Dalrymple, D., Skalse, J., Bengio, Y., Russell, S., Tegmark, M., Seshia, S., Omohundro, S., Szegedy, C., Goldhaber, B., Ammann, N., et al. Towards guaranteed safe AI: A framework for ensuring robust and reliable AI systems. *arXiv preprint arXiv:2405.06624*, 2024.

de Moura, L. and Ullrich, S. The Lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28*, volume 12699 of *Lecture Notes in Computer Science*, pp. 625–635. Springer, 2021. doi: 10.1007/978-3-030-79876-5_37.

Haupt, A. A., Christoffersen, P. J. K., Damani, M., and Hadfield-Menell, D. Formal contracts mitigate social dilemmas in multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 38(2):51, 2024. doi: 10.1007/s10458-024-09682-5.

Kalai, A. T., Kalai, E., Lehrer, E., and Samet, D. A commitment folk theorem. *Games and Economic Behavior*, 69 (1):127–137, 2010. doi: 10.1016/j.geb.2009.09.008.

Klein, G., Elphinstone, K., Heiser, G., Andronick, J., Cock, D., Derrin, P., Elkaduwe, D., Engelhardt, K., Kolanski, R., Norrish, M., Sewell, T., Tuch, H., and Winwood, S. seL4: Formal verification of an OS kernel. In *Proceedings of the 22nd ACM Symposium on Operating Systems Principles*, SOSP 2009, pp. 207–220. ACM, 2009. doi: 10.1145/1629575.1629596.

Le Roux, S., Martin-Dorel, É., and Smaus, J.-G. An existence theorem of Nash equilibrium in Coq and Isabelle. In *Proceedings of the 8th International Symposium on Games, Automata, Logics, and Formal Verification (GandALF 2017)*, volume 256 of *EPTCS*, pp. 46–60, 2017. doi: 10.4204/EPTCS.256.4.

Leroy, X. Formal verification of a realistic compiler. *Communications of the ACM*, 52(7):107–115, 2009. doi: 10.1145/1538788.1538814.

Lorè, N. and Heydari, B. Strategic behavior of large language models and the role of game structure versus contextual framing. *Scientific Reports*, 14:18490, 2024. doi: 10.1038/s41598-024-69032-z.

Madmoun, H. and Lahlou, S. Communication enables cooperation in LLM agents: A comparison with curriculum-based approaches. *arXiv preprint arXiv:2510.05748*, 2025. arXiv v3 dated 11 Mar 2026; EACL 2026.

MixedMatched contributors. Formalizing game theory in Lean 4. https://github.com/MixedMatched/formalizing-game-theory, 2024.

Oesterheld, C. Robust program equilibrium. *Theory and Decision*, 86(1):143–159, 2019. doi: 10.1007/s11238-018-9679-3.

Ondras, J. Verified Handshake: Lean 4 Source Code and Machine-Checked Proofs. https://github.com/jancio/VerifiedHandshake, 2026.

Pomeret-Coquot, P., Fargier, H., and Martin-Dorel, É. BelGames: A formal theory of games of incomplete information based on belief functions in the Coq proof assistant. In *14th International Conference on Interactive Theorem Proving (ITP 2023)*, volume 268 of *LIPIcs*, pp. 25:1–25:19, 2023. doi: 10.4230/LIPIcs.ITP.2023.25.

Sauerberg, N. and Oesterheld, C. Promises made, promises kept: Safe Pareto improvements via ex post verifiable commitments. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(20):17231–17241, 2026. doi: 10.1609/aaai.v40i20.38774. Oral presentation; arXiv:2505.00783.

Tennenholtz, M. Program equilibrium. *Games and Economic Behavior*, 49(2):363–373, 2004. doi: 10.1016/j.geb.2004.02.002.

The mathlib Community. The Lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2020)*, pp. 367–381, 2020. doi: 10.1145/3372885.3373824.

Zinzindohoué, J.-K., Bhargavan, K., Protzenko, J., and Beurdouche, B. HACL*: A verified modern cryptographic library. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS 2017, pp. 1789–1806. ACM, 2017. doi: 10.1145/3133956.3134043.

A. n-Player Generalization (Lean Source)

We list the central definitions and theorem signatures (Listing 3); full proofs are at (Ondras, 2026).

Symmetric case. From `NPlayer.lean`:

Listing 3. The mechanized n-player generalization (symmetric case). Shown are the count-indexed game `SymGameN`, the n-ary wrapper `wrapN` (realizing C iff every attestation carries C), and the two main theorem signatures; their full statements are Theorems 4.6–4.7, with full proofs in the artifact.

```
structure SymGameN (n : ℕ) where
  u : Action → Fin n → ℚ
def wrapN {n : ℕ} (atts : Fin n → Attestation) : Action :=
  if ∀ i, (atts i).action = .C then .C else .D
theorem wrapN_weakly_dominates {n : ℕ} (hn : 2 ≤ n) ... -- weak dominance (Thm 4.6)
theorem wrapN_pareto_unique     {n : ℕ} (hn : 2 ≤ n) ... -- Pareto-uniqueness (Thm 4.7)
```

Asymmetric case. Mirror with `AsymGameN n` and the per-player `wrapN_weakly_dominates_asym`; the wrapper is unchanged since it is action-agnostic. A coercion `symGameN_toAsym` embeds every symmetric n-player game in the asymmetric framework, with the lifting lemma `stagHuntN_toAsym` closing the obligation that a symmetric Stag-Hunt remains Stag-Hunt under the embedding.

Truthful-oracle capstone. The agent-level lift `cooperate_realized_n_truthful` is axiom-free: an inline `unfold wrapN; if_pos hAll` discharged by `TrustedOracle.truthful`, deliberately avoiding rewriting through `wrapN_eq_C_iff` (the lemma is itself axiom-free, but the rewrite tactic for an `Iff` would introduce `propext` into the closing proof's transitive dependencies).

B. Per-Declaration Axiom Audit

Table 2 gives the trust base by footprint; Table 3 lists every audited declaration with its footprint. Beyond the foundational set `[propext, Classical.choice, Quot.sound]`, the development appeals to no mathlib lemmas other than decidable equality on `Action`, finite-type universal-quantifier decidability and `Finset`-cardinality reasoning for the n-player and asymmetric lifts, and `linarith`'s internal use of $\mathbb{Q}$-linear arithmetic; there are no `sorry`s and no custom axioms.

Table 3. Per-declaration foundational-axiom footprint of all 56 audited declarations, read off the build-time `#guard_msgs in #print axioms` guards in `Audit.lean` (the detail behind Table 2). A check (✓) marks an axiom a declaration depends on: `P=propext`, `C=Classical.choice`, `Q=Quot.sound`; an all-blank row is axiom-free. The *Group* column groups rows by footprint tier (whose sizes match Table 2) and, within the full-kernel tier, by structural family.

Group	Declaration	File (.lean)	P	C	Q
axiom-free	wrap_comm	Theorem			
	wrapN_eq_C_iff	NPlayer			
	cooperate_realized_under_truthful_oracle	TrustedOracle			
	cooperate_realized_n_truthful	NPlayer			
	lyingAttest_not_truthful	Deployments			
	lyingAttest_n_not_truthful	Deployments			
	stagHunt_toAsym	Asymmetric			
	SymGame.toSymGameN	NPlayer			
propext	wrap_eq_C_iff	Theorem	✓		
	wrapN_two_eq_wrap	NPlayer	✓		
	AsymGame.toAsymGameN	NPlayerAsym	✓		
	asymPayoffN_two_eq_asymPayoffRow	NPlayerAsym	✓		
	asymPayoffN_two_eq_asymPayoffCol	NPlayerAsym	✓		
Quot.sound	wrapN_comm	NPlayer			✓
propext + Quot.sound	payoffN_two_eq_payoff	NPlayer	✓		✓
2-player	wrap_weakly_dominates	Theorem	✓	✓	✓
	wrap_strict_when_opponent_commits	Theorem	✓	✓	✓
	wrap_pareto_unique	Theorem	✓	✓	✓
	wrap_weakly_dominates_agent	TrustedOracle	✓	✓	✓
	wrap_pareto_unique_agent	TrustedOracle	✓	✓	✓
2-player asym.	wrap_weakly_dominates_asym_row	Asymmetric	✓	✓	✓
	wrap_weakly_dominates_asym_col	Asymmetric	✓	✓	✓
	wrap_strict_when_opponent_commits_asym_row	Asymmetric	✓	✓	✓
	wrap_strict_when_opponent_commits_asym_col	Asymmetric	✓	✓	✓
	wrap_pareto_unique_asym_row	Asymmetric	✓	✓	✓
	wrap_pareto_unique_asym_col	Asymmetric	✓	✓	✓
n-player	wrapN_weakly_dominates	NPlayer	✓	✓	✓
	wrapN_strict_when_others_commit	NPlayer	✓	✓	✓
	wrapN_pareto_unique	NPlayer	✓	✓	✓
	wrapN_weak_nash	NPlayer	✓	✓	✓
	wrapN_weakly_dominates_agent	NPlayer	✓	✓	✓
	wrapN_pareto_unique_agent	NPlayer	✓	✓	✓
n-player asym.	wrapN_weakly_dominates_asym	NPlayerAsym	✓	✓	✓
	wrapN_strict_when_others_commit_asym	NPlayerAsym	✓	✓	✓
	wrapN_pareto_unique_asym	NPlayerAsym	✓	✓	✓
	wrapN_weakly_dominates_asym_agent	NPlayerAsym	✓	✓	✓
	wrapN_pareto_unique_asym_agent	NPlayerAsym	✓	✓	✓
lifts	stagHuntN_toAsym	NPlayerAsym	✓	✓	✓
	stagHuntOrdering_toN	NPlayer	✓	✓	✓
	asymStagHuntOrdering_toN	NPlayerAsym	✓	✓	✓
specialization	wrap_weakly_dominates_via_n	NPlayer	✓	✓	✓
	wrap_strict_when_opponent_commits_via_n	NPlayer	✓	✓	✓
	wrap_pareto_unique_via_n	NPlayer	✓	✓	✓
	wrap_weakly_dominates_asym_row_via_n	NPlayerAsym	✓	✓	✓
	wrap_weakly_dominates_asym_col_via_n	NPlayerAsym	✓	✓	✓
	wrap_strict_when_opponent_commits_asym_row_via_n	NPlayerAsym	✓	✓	✓
	wrap_strict_when_opponent_commits_asym_col_via_n	NPlayerAsym	✓	✓	✓
	wrap_pareto_unique_asym_row_via_n	NPlayerAsym	✓	✓	✓
	wrap_pareto_unique_asym_col_via_n	NPlayerAsym	✓	✓	✓
ε-incentive	wrapN_strictly_dominates_asym_eps	Incentive	✓	✓	✓
	wrapN_strictly_dominates_asym_eps_agent	Incentive	✓	✓	✓
	wrapN_strictly_dominates_eps	Incentive	✓	✓	✓
	wrapN_strictly_dominates_eps_agent	Incentive	✓	✓	✓
	wrap_strictly_dominates_eps	Incentive	✓	✓	✓
	wrap_strictly_dominates_asym_row_eps	Incentive	✓	✓	✓
	wrap_strictly_dominates_asym_col_eps	Incentive	✓	✓	✓