The Separation Predicate™

A Formal Definition for Zero-Knowledge Compliance Verification

Technical Specification — Version 1.0

April 2026

A formal-methods companion to The AI Accountability Standard. Intended for cryptographers, applied security researchers, standards-body technical working groups, regulatory technology staff, and engineering teams implementing compliance verification for consequential algorithmic decisioning systems.

Abstract

We formalize the Separation Predicate, a cryptographic compliance primitive that allows a party executing an algorithmic decision to produce a succinct, publicly-verifiable certificate attesting that the decision's output is structurally independent of a declared set of prohibited inputs, with respect to a committed model and a published prohibited-inputs registry. The predicate is zero-knowledge: the certificate reveals nothing about the model's weights, the decision subject's feature values, or any input beyond what the registry and the decision's public identifier disclose. We define the predicate formally, state its security properties (completeness, soundness, zero-knowledge), specify its composition into a four-property compliance standard (Model Identity, Separation, Chained Integrity, Credentialed Review), and discuss its limitations, open problems, and relationship to prior work on verifiable machine learning and regulatory technology.

The Separation Predicate is intended to be construction-independent: it can be instantiated using any of several succinct proof systems (zk-SNARKs, zk-STARKs, and successors) on any cryptographically-appropriate curve or hash family, without altering the predicate's formal meaning or external security guarantees. We specify the predicate at the level of what must be proved, not how, so that the framework remains stable as the underlying cryptographic primitives evolve.

1. Introduction

1.1 Motivation

Algorithmic decisioning systems increasingly produce consequential outputs in domains — healthcare utilization management, consumer lending, employment screening, insurance underwriting, content moderation, government benefits — where regulators, affected subjects, and other third parties have legitimate interests in verifying specific properties of the system's behavior. The properties most commonly at issue are whether the system's output was influenced by inputs it should not have considered (protected characteristics, cost signals, litigation-propensity proxies) and whether the claimed model, features, and review workflow were in fact the ones that produced the decision.

Current governance infrastructure answers these questions through organizational attestation — policy documents, audit opinions, compliance attestations — that the verifying parties must accept without independent confirmation. This infrastructure is structurally limited: the evidence originates with the party whose behavior is being evaluated, and external parties have no mechanism to confirm the evidence corresponds to what actually occurred. The limitation is particularly acute for AI systems because the activity being evaluated occurs inside computational processes that leave no externally-observable traces.

We present the Separation Predicate as a formal cryptographic object that closes this gap for a specific and important class of compliance questions. Informally, the Separation Predicate proves that a decision's output would not have changed if a declared set of prohibited inputs had been zeroed or set to reference values — that is, the decision is structurally independent of those inputs. The proof is succinct (typically 1-2 kilobytes), publicly verifiable (in under 50 milliseconds on commodity hardware), and zero-knowledge (reveals nothing about the model, the subject's inputs, or any computation beyond what the predicate's public statement expresses).

1.2 Contributions

This paper makes four contributions. First, we give a formal definition of the Separation Predicate as an NP-relation parameterized by a committed model, a published prohibited-inputs registry, and a decision identifier. Second, we state and justify the predicate's security properties — completeness, knowledge soundness, zero-knowledge — under standard cryptographic assumptions. Third, we specify how the predicate composes with three other predicates (Model Identity, Chained Integrity, Credentialed Review) to form the four-property compliance standard described in prior work. Fourth, we discuss limitations: what the predicate does not capture, how adversarial adoption can weaken its guarantees, and where additional work is needed.

1.3 Scope and non-scope

The paper is a formal-definition paper, not a construction paper. We specify what the Separation Predicate proves, not a particular cryptographic construction for proving it. Several candidate constructions exist in the zk-SNARK and zk-STARK literatures; the predicate's formal meaning does not depend on the choice. Implementers should select constructions based on their own constraints — proof size, prover time, verifier time, trust assumptions, hardware — with the understanding that any sound construction instantiating the predicate produces certificates with the same external semantics.

The paper is also not a policy paper. It presumes familiarity with the compliance motivations developed in the companion papers on verifiable governance in prior authorization and the AI Accountability Standard. Readers seeking the policy argument for why the Separation Predicate matters should consult those papers. Readers seeking the technical specification should continue here.

2. Preliminaries and Notation

2.1 Actors

The Separation Predicate involves four actors. The Deployer is the organization operating the algorithmic decisioning system. The Subject is the individual or entity about whom a decision is being made (the applicant, the member, the candidate). The Prover is the computational role that produces certificates at decision time; typically the Prover is operated by the Deployer, but it may be operated by a third party,

an escrow service, or a regulatory infrastructure. The Verifier is any party subsequently checking a certificate; the Verifier may be a regulator, the Subject, a delegated counterparty, a court, a journalist, or any other party in possession of the certificate and the corresponding public verification key.

The Separation Predicate is designed to be sound against a potentially adversarial Prover — that is, the Verifier should be unable to accept a certificate for a decision that does not satisfy the predicate, even if the Prover is actively trying to cheat. This is the standard soundness requirement of succinct proofs and distinguishes the framework from attestation-based approaches in which the Verifier must trust the party producing the evidence.

2.2 Objects

We work with the following objects:

> **Model commitment** c_M: a binding cryptographic commitment to the weights and architecture of a specific model M. The commitment is published in a model registry such that any Verifier can obtain it independently. Commitments are binding under standard collision-resistance assumptions: the Prover cannot open c_M to two different models.
>
> **Feature space** $X = X_1 \times X_2 \times \ldots \times X_n$: the n-dimensional space of inputs the model consumes. Each dimension X_i corresponds to a named feature; the feature names and their positions in x are part of the model's committed specification and cannot be altered without changing c_M.
>
> **Prohibited-inputs registry** *R: a signed, versioned, publicly-published list specifying a subset $F_{prohibited} \subseteq \{1, \ldots, n\}$ of feature indices that the decision must be structurally independent of. R is committed by the Deployer and signed by an appropriate authority (the Deployer's governance board, a regulatory body, or a standards-setting entity). A reference-value function ref: $F_{prohibited} \rightarrow X_i$ specifies the value to which each prohibited feature should be set for the structural-independence check.*
>
> **Decision identifier** d: a unique identifier for the specific decision the certificate accompanies. d binds the certificate to this decision and prevents certificate reuse across decisions. d typically includes a timestamp, a subject pseudonym, and a workflow identifier.
>
> **Output space** Y: the space of possible decision outputs. For classification workflows, Y is discrete; for regression or scoring workflows, Y is a real interval. The notion of "output equivalence" used in the predicate may be exact equality (for discrete Y) or a tolerance-bounded equivalence (for continuous Y, parameterized by an acceptable deviation ε).

Feature vectors are drawn from X. A feature vector x is partitioned by the registry into x_prohibited (the features at indices in F_prohibited) and x_permitted (the remaining features). The reference-substituted vector x' is defined by replacing the values at prohibited indices with their reference values: x'_i = ref(i) for i ∈ F_prohibited, and x'_i = x_i otherwise.

3. Formal Definition of the Separation Predicate

3.1 The NP-relation

The Separation Predicate is defined as an NP-relation R_SP between public statements and private witnesses.

The public statement consists of:

```
stmt = (c_M, R_version, d, y, commit_x)
```

where c_M is the model commitment, R_version identifies the specific version of the prohibited-inputs registry in force at decision time, d is the decision identifier, y ∈ Y is the claimed decision output, and commit_x is a commitment to the feature vector x. The commitment commit_x is binding but hiding: the Verifier cannot recover x from commit_x, but the Prover is bound to a specific x by the commitment.

The private witness consists of:

```
wit = (M, x, r_x)
```

where M is the model (opening c_M), x is the feature vector (opening commit_x with randomness r_x), and the witness satisfies the relation below.

The relation R_SP holds for (stmt, wit) if and only if all of the following are true:

> **(i) Model consistency.** The committed model opens correctly: c_M = Commit(M).
>
> **(ii) Input consistency.** The committed feature vector opens correctly: commit_x = Commit(x, r_x).
>
> **(iii) Output correctness.** The model's evaluation on x yields the claimed output: M(x) = y.
>
> **(iv) Structural independence.** The model's evaluation on the reference-substituted input yields an equivalent output: M(x') ≡ y, where x' is defined relative to the prohibited-inputs registry version R_version, and ≡ is the output-equivalence relation appropriate to Y (exact equality for discrete outputs, tolerance-bounded equivalence for continuous outputs).

Condition (iv) is the core of the Separation Predicate. It states that zeroing or reference-substituting the prohibited features does not change the decision output. The condition is evaluated at the specific decision point — it is a claim about this decision's structural independence, not a general claim about the

model's behavior across the input distribution. The distinction matters: pointwise independence is a weaker property than distributional independence, but it is the property that is per-decision verifiable and that corresponds to the compliance question of interest.

3.2 Output equivalence for continuous outputs

For continuous output spaces, strict equality M(x') = y is typically too restrictive: numerical evaluation introduces floating-point noise, and reasonable compliance definitions admit small deviations. We parameterize the predicate by an output-equivalence relation ≡_ε defined by

```
y_1 ≡_ε y_2  iff  |y_1 − y_2| < ε
```

where ε is a tolerance published alongside the registry. The choice of ε is a governance question: a lax tolerance weakens the compliance claim; a strict tolerance may be practically unachievable for some models. We recommend ε be set at a level below any plausible decision-relevant threshold in the workflow — for example, in a binary-classification workflow with a threshold at 0.5, ε should be small enough that no reasonable reference substitution could push a decision across the threshold without ε being exceeded. The published ε becomes part of the governance artifact subject to external evaluation.

3.3 The decision output in the public statement

We include y (the decision output) in the public statement rather than treating it as a private witness. This is a deliberate design choice with two consequences. First, the certificate binds the decision output publicly: the Verifier learns what the decision was, not merely that some satisfying decision exists. This is appropriate for compliance verification, where the Subject and Verifier need to know the decision's content. Second, it prevents the Prover from producing a certificate that validates against multiple possible outputs, which would defeat the purpose.

In some deployment contexts, y itself is sensitive (for example, in highly regulated medical determinations). In those contexts, y can be replaced in the public statement by a commitment commit_y, with the opening of commit_y provided to the Subject and authorized Verifiers through a separate channel. The formal definition accommodates this variation without modification: the relation R_SP is defined identically, with Commit(y) replacing y where confidentiality requires it.

4. Security Properties

We require the Separation Predicate to be instantiated by a proof system satisfying three standard properties: completeness, knowledge soundness, and zero-knowledge. We state each property explicitly and note the assumptions under which it holds.

4.1 Completeness

For all statements stmt and witnesses wit such that R_SP(stmt, wit) holds, an honest Prover produces a proof π such that the Verifier accepts with probability 1 (for perfect completeness) or with probability at least 1 − negl(λ) (for statistical completeness), where λ is the security parameter and negl(λ) is a negligible function of λ.

Completeness is the guarantee that honest behavior by all parties produces accepted proofs. It rules out constructions in which even correct predicates occasionally fail to verify, which would be unworkable in a compliance setting where decisions must reliably produce valid certificates.

4.2 Knowledge soundness

For every probabilistic polynomial-time Prover P* and every auxiliary input, if P* produces an accepting proof π for statement stmt with non-negligible probability, there exists a probabilistic polynomial-time extractor E that, given black-box access to P*, extracts a witness wit such that R_SP(stmt, wit) holds with all but negligible probability.

Knowledge soundness is the guarantee that an accepting proof implies the Prover actually possesses a valid witness. In the compliance setting, this means that a Verifier who accepts a Separation Predicate proof can conclude, with cryptographic certainty, that the Deployer actually has a model and input consistent with the claimed output and structural independence — not merely that the claim appears plausible.

Knowledge soundness typically holds under computational assumptions that depend on the underlying construction. SNARK constructions based on pairing-friendly elliptic curves (Groth16, PLONK) require knowledge-of-exponent assumptions or power-of-tau trusted setup. Constructions based on hash functions (STARKs, Halo2 without pairing) require only collision-resistance of the hash. Implementers should select constructions whose assumptions are acceptable in their threat model; the Separation Predicate's formal meaning is unchanged by the choice.

4.3 Zero-knowledge

For every probabilistic polynomial-time Verifier V* there exists a probabilistic polynomial-time simulator S such that for every valid (stmt, wit), the distribution of (stmt, π) produced by an honest Prover interacting with V* is computationally indistinguishable from the distribution of (stmt, π') produced by S given only stmt.

Zero-knowledge is the guarantee that the proof reveals nothing beyond the statement. In the Separation Predicate setting, this means the Verifier learns:

> — The model commitment c_M (which identifies which model produced the decision, without revealing the model's weights);

> — The registry version R_version (which specifies the prohibited-inputs list, which is public anyway);
>
> — The decision identifier d (which is public by the nature of the compliance workflow);
>
> — The decision output y (or a commitment to y, per Section 3.3);
>
> — The fact that a valid witness exists — that is, the model evaluated the input to produce y and structural independence holds.

The Verifier learns nothing else. Specifically, the Verifier learns nothing about the model's weights beyond what c_M discloses (which, for a binding commitment, is nothing), nothing about the Subject's feature vector x beyond what the output y implies, and nothing about the model's behavior on any other input. This is the "zero-knowledge compliance verification" of the paper's title: compliance is verified without compromising the intellectual property of the model, the privacy of the Subject, or the integrity of the model's behavior on unrelated inputs.

4.4 Non-properties worth stating

Several properties that are sometimes confused with the above are explicitly not guaranteed by the Separation Predicate, and we state them here to prevent misunderstanding.

The predicate does not prove that the model is fair, accurate, or well-calibrated. Structural independence from the declared prohibited inputs is a specific, narrow property; it does not entail the absence of other forms of bias, the correctness of the model's decision criteria, or the appropriateness of the training data.

The predicate does not prove that the prohibited-inputs registry is complete. If the Deployer declares prohibition from feature X but omits feature X' which is a proxy for X, the predicate validates correctly despite X' being used. Completeness of the registry is a governance question, not a cryptographic one.

The predicate does not prove that the decision was made in good faith. A Deployer who selects an adversarial reference-value function ref — one that sets prohibited features to values at which the model happens to agree with the actual-input decision, even though the model's general behavior is influenced by those features — can produce valid certificates despite having constructed the reference substitution to hide the influence. Mitigation requires governance constraints on how ref is chosen, which we discuss in Section 7.

The predicate does not prove anything about training-time behavior. The model's weights, once committed, are evaluated at inference time; how they came to be what they are is outside the scope of this predicate. Training-time attestations are a distinct class of compliance primitive, complementary to the Separation Predicate but formally separate.

5. Construction Outline

We outline how the Separation Predicate is instantiated using a generic succinct proof system. The outline is construction-independent and is intended to guide implementation rather than specify it. Specific constructions will vary in their performance characteristics, trust assumptions, and engineering complexity; we identify the key design choices without prescribing them.

5.1 Arithmetization

The relation R_SP is arithmetized — expressed as a system of polynomial constraints over a finite field — for input to a proof system. The arithmetization expresses the four conditions of R_SP as constraints:

> — Commitment openings (conditions (i) and (ii)): constraints linking c_M to the model's weight vector w and commit_x to the feature vector x, via the commitment scheme's algebraic structure.
>
> — Forward evaluation (condition (iii)): constraints expressing the model's computation M(x) = y. For a neural network, this involves multiplication and addition constraints for each layer's affine transformation, and constraints expressing the nonlinear activation (ReLU, sigmoid, etc.) as piecewise-linear or lookup-table constraints. For a tree-boosted model, the constraints express the traversal of each tree and the aggregation of leaf values.
>
> — Reference substitution (condition (iv)): constraints that construct x' from x by applying the reference-value function at the prohibited indices, and a second forward evaluation M(x') = y' subject to the equivalence relation.

The arithmetization roughly doubles the constraint count relative to a proof of only the forward evaluation, because both M(x) and M(x') must be attested. For models of moderate complexity (a few million parameters, tens to hundreds of features), the total constraint count is in the range of tens to hundreds of millions of constraints, which is within the range of current proof systems on appropriate hardware.

5.2 Proof system selection

Several classes of proof system can instantiate the arithmetized relation. The choice involves tradeoffs across proof size, prover time, verifier time, trusted setup requirements, and post-quantum security posture.

> **SNARKs with trusted setup (Groth16 and derivatives):** produce the smallest proofs (a few hundred bytes) and fastest verification (a few milliseconds), but require a per-circuit trusted setup or a universal trusted setup with non-trivial ceremony requirements. Appropriate

when proof size and verifier speed are paramount and the trusted-setup requirement is acceptable.

Universal-setup SNARKs (PLONK, Marlin, SONIC): allow the same trusted setup to serve many different circuits, reducing the ceremony burden. Proof sizes are slightly larger than Groth16; prover time is comparable or slightly better for structured circuits. Appropriate when the same infrastructure serves many compliance predicates.

STARKs and transparent SNARKs (FRI, Halo2 without pairings): eliminate trusted setup entirely, basing security only on hash-function collision resistance. Proofs are larger (tens of kilobytes to hundreds of kilobytes) and verifier time is slower, but post-quantum security is typically better. Appropriate when trusted setup is unacceptable or when post-quantum concerns dominate.

For production compliance deployments at the scale of consequential AI decisioning, the current preferred configuration is a universal-setup SNARK with recursive proof composition, which allows per-decision proof generation in the hundreds of milliseconds on FPGA hardware and proof sizes in the 1-2 kilobyte range. The specific choice will evolve with the cryptographic literature; the Separation Predicate's formal meaning is invariant to the choice.

5.3 The reference-value function as a governance artifact

The reference-value function ref is cryptographically straightforward — it is a mapping from prohibited feature indices to reference values — but its choice is a governance decision with substantive implications. Several standard choices exist:

Zero reference (ref(i) = 0): straightforward but may produce reference inputs that are out-of-distribution for the model, yielding unreliable output comparisons.

Population-mean reference (ref(i) = mean(X_i)): keeps reference inputs closer to the training distribution but requires the mean to be published and versioned alongside the registry.

Redacted-input reference (ref(i) = a special REDACTED token the model is trained to handle): requires model-side support for the token but produces the cleanest semantic interpretation of "the decision in the absence of this feature."

Multiple-reference aggregation: the predicate is evaluated against several references simultaneously and independence is required against all of them. Strongest guarantee but multiplies the proof cost.

The choice of ref is the single most important governance decision in a Separation Predicate deployment, because adversarial choice of ref can weaken the predicate's substantive meaning. A deployer who selects

a reference function under which the model happens to produce the actual output — even for inputs where the model's general behavior is influenced by prohibited features — can produce valid certificates that mask the influence. Mitigation requires external specification of ref (through regulation, standards body, or independent governance) rather than Deployer self-selection, and is the single most important complement to the cryptographic machinery itself.

6. Composition into the Four-Property Standard

The Separation Predicate is one of four cryptographic compliance primitives that compose into the AI Accountability Standard defined in companion work. We briefly specify the other three and describe how they compose with the Separation Predicate.

6.1 Model Identity Predicate (π_ID)

π_ID is the relation that binds a decision to a specific committed model version. Its statement includes (c_M, d, y) and its witness includes (M, x) with the relation holding iff c_M = Commit(M) and M(x) = y. This is the "forward evaluation only" subset of R_SP — condition (iv) is dropped. π_ID establishes which model produced the decision without establishing structural independence; it is appropriate as a standalone primitive in workflows where independence is not required but model identity is.

6.2 Chained Integrity Predicate (π_CI)

π_CI is a structural predicate over sequences of certificates rather than individual ones. For a sequence of decisions d_1, d_2, …, d_k for the same subject, the same model version, and the same workflow, π_CI attests that each certificate's commitment includes a hash of the previous certificate in the chain, and that the chain's head commits to a value that is periodically anchored in a public tamper-evident registry maintained outside the Deployer's control. π_CI is straightforwardly verifiable given access to the chain and the anchor.

6.3 Credentialed Review Predicate (π_CR)

π_CR binds a decision to a specific human reviewer's credential at the time of decision. The credential is represented as an attestation issued by a credentialing authority (a licensing board, an organizational authority, a regulatory registry). π_CR's statement includes (credential_commit, d) and its witness is an opening of credential_commit against the credentialing authority's published attestation chain, restricted to the decision's timestamp. The predicate is soundly instantiated using standard digital-signature and attestation-chain techniques and typically does not require SNARK machinery; a signed attestation suffices.

6.4 Composition

The four predicates compose by concatenation: a decision's full certificate is (π_ID, π_SP, π_CI, π_CR) together with the shared public statement components. Verification is conjunctive — the certificate is accepted only if all four proofs verify against their respective public statements. The composition's proof size is the sum of the component proof sizes; verification time is the sum of verification times. At the representative sizes in prior work (Sections 3.1-3.3 of the Accountability Standard paper), the composed certificate totals roughly 1.4-2 kilobytes with verification time under 50 milliseconds.

More sophisticated compositions are possible and advantageous. Recursive proof composition — wrapping the four proofs inside a single outer SNARK that attests to their joint validity — can reduce the composed certificate to a single constant-size artifact at the cost of additional prover work. Aggregate proof composition — producing a single proof over many decisions' certificates — supports population-level regulatory reporting, where a regulator verifies a single proof covering millions of decisions rather than verifying each individually. These compositions are orthogonal to the Separation Predicate's formal definition and are implemented at the proof-system level.

7. Limitations and Open Problems

We name the limitations of the Separation Predicate explicitly, both because honesty about limits is essential for a cryptographic compliance primitive intended for regulatory use and because open problems are where future work should focus.

7.1 Pointwise versus distributional independence

R_SP proves structural independence at the specific decision point — given this input, the output would be the same with the prohibited features substituted. It does not prove that the model is distributionally independent of the prohibited features — that in general, across the input distribution, the features have no influence. The pointwise property is verifiable per decision; the distributional property is not, and can only be established by other means (training-time constraints, population-level statistical testing, causal-inference analyses) that are complementary to the predicate but not entailed by it.

This distinction has been a source of misunderstanding in early discussions of the predicate. The predicate's guarantee is precisely the pointwise one: for this decision, the prohibited features did not affect the output. A model that is generally influenced by prohibited features may still produce decisions that pointwise satisfy the predicate — simply because for this particular input, the influence happens to not flip the output. The population-level pattern of such non-flipping decisions is itself an empirical artifact that distributional methods can study; per-decision pointwise independence is what the predicate establishes.

7.2 Adversarial reference selection

As noted in Section 5.3, the reference-value function ref can be chosen adversarially to produce valid certificates that mask genuine feature influence. The theoretical worst case is a ref specifically constructed to produce the actual decision output as its reference output, in which case R_SP validates trivially for the decision at issue. Mitigation requires that ref be externally specified — by regulation, by a standards body, or by an independent governance process — rather than chosen by the Deployer. We recommend that regulatory adoption of the Separation Predicate include specification of ref, at least to the level of requiring a standard family of references (zero, population-mean, redacted-input) and prohibiting bespoke Deployer-selected references.

7.3 Registry completeness

The predicate proves independence only from declared prohibited features. Completeness of the declaration is outside the cryptographic scope and depends on governance. The architectural mitigation is making the registry public and subject to external scrutiny; the deeper mitigation is developing sector-specific regulatory minimums for what must be declared, so that the registry's completeness is not entirely at the Deployer's discretion.

7.4 Training-time behavior

R_SP is an inference-time predicate. It says nothing about how the model's committed weights came to have the values they have. A model trained on biased data can exhibit inference-time independence from the prohibited features and still produce systematically problematic decisions, because the bias is encoded in the parts of the model R_SP does not examine. Complementary predicates — training-data commitments, training-configuration attestations, training-time fairness proofs — are an active area of research but are less mature than inference-time predicates. We expect training-time predicates to develop over the next three to five years and to compose with R_SP into a more complete compliance standard.

7.5 Prover corruption and side channels

Knowledge soundness holds under standard cryptographic assumptions but presumes the Prover executes the cryptographic machinery honestly with respect to the witness it holds. A Prover with access to side-channel information about the Verifier, the Subject, or other parties can potentially compromise zero-knowledge. A Prover that substitutes witnesses before proving (using a different M than the one that produced the actual decision, for example) can produce accepting proofs that do not correspond to the actual decision. These are implementation concerns rather than formal-definition concerns, but they matter for deployments. Standard mitigations — hardware-attested execution environments, side-channel-resistant implementations, witness-binding at decision time — are available but must be engineered carefully.

7.6 Model classes not yet efficiently supported

Current proof systems are most efficient for models with structured arithmetic — neural networks with standard activations, tree-boosted ensembles, linear and generalized linear models. Large language models pose open challenges: their trillion-parameter scales exceed current prover capacities by orders of magnitude, their autoregressive generation is less naturally arithmetized than classification, and their non-determinism under sampling complicates the output-equivalence relation. Work on LLM-specific proof systems is active but not yet at production scale. Consequential decisioning systems that use LLMs in the decision loop require either careful engineering to isolate the proof-relevant computation, or acceptance that the Separation Predicate's current form does not efficiently cover LLM-based decisions. This is the most significant open problem in extending the framework.

7.7 Quantum security

Some candidate constructions (pairing-based SNARKs) rely on assumptions known to be broken by sufficiently-capable quantum computers. Others (hash-based STARKs, lattice-based SNARKs) are conjectured post-quantum secure. Deployers in sectors with long retention requirements for compliance certificates — particularly public-sector deployments where certificates may need to be verifiable decades after generation — should prefer post-quantum-conjectured constructions even when their performance is less favorable. The Separation Predicate's formal meaning is construction-independent, so migration between constructions is possible without altering the predicate's semantics.

8. Relationship to Prior Work

8.1 Succinct proof systems

The Separation Predicate is a special-purpose instantiation of general-purpose succinct non-interactive arguments. The technical machinery draws on the SNARK and STARK literatures — including Groth16 and its derivatives, PLONK, Halo2, FRI-based systems, and recursive proof composition — without contributing new cryptographic techniques. Our contribution is in the specification of the predicate itself: what to prove, not how to prove it. The how is well-addressed by the existing literature and is expected to improve over time.

8.2 Verifiable machine learning

A body of recent work addresses cryptographic verification of machine learning inference — demonstrating that a specific model produced a specific output on a specific input. This literature, sometimes referred to as zkML, provides the forward-evaluation machinery that instantiates condition (iii) of R_SP. The Separation Predicate extends this work by incorporating conditions (i), (ii), and especially (iv) — structural independence — as first-class properties of the proof. zkML systems typically focus on proving correctness of inference for commercial applications (attested model outputs for paid inference

services) rather than compliance verification against governance constraints. Our framing applies the same cryptographic machinery to a different end.

8.3 Differential privacy and anti-discrimination ML

Differential privacy and the anti-discrimination machine learning literature address substantive fairness concerns — whether model behavior is biased, whether a system discriminates against protected groups, whether privacy is preserved in outputs. These literatures operate on population-level statistical properties and training-time constraints rather than per-decision cryptographic verification. The Separation Predicate is complementary: it does not replace distributional fairness work, but provides a per-decision evidentiary layer that distributional work does not. A rigorous deployment would use both.

8.4 Regulatory technology and compliance automation

Regulatory technology (RegTech) has produced a range of compliance automation tools — KYC verification, transaction monitoring, reporting pipelines, algorithmic audit tools. These tools primarily automate the production and processing of attestation-based compliance artifacts. The Separation Predicate is qualitatively different: it replaces attestation with cryptographic evidence. RegTech infrastructure and cryptographic compliance primitives are not in opposition; the predicate can be integrated into existing RegTech pipelines as a higher-assurance evidence layer, with the RegTech infrastructure handling workflow, storage, and reporting while the predicate provides the underlying evidence.

8.5 Standards and regulatory frameworks

NIST's AI Risk Management Framework, ISO/IEC 42001, the EU AI Act's technical standards, and sector-specific frameworks (including CMS rules on AI in Medicare Advantage, state-level AI-in-UM statutes, NYC Local Law 144) specify compliance expectations for AI systems without yet specifying the cryptographic primitives that would make compliance verifiable. The Separation Predicate is intended as a candidate technical primitive that standards bodies and regulators can reference in making their compliance expectations externally verifiable. We do not propose the predicate as a replacement for these frameworks but as an implementation primitive for properties the frameworks describe.

9. Conclusion

We have presented a formal definition of the Separation Predicate, a cryptographic compliance primitive that enables zero-knowledge verification of structural independence in algorithmic decisioning. The predicate is defined as an NP-relation over a public statement (model commitment, registry version, decision identifier, decision output, input commitment) and a private witness (model, input), with a relation that entails model consistency, input consistency, output correctness, and structural independence from a declared prohibited-inputs registry. The predicate is completeness-preserving,

knowledge-sound under standard cryptographic assumptions, and zero-knowledge against polynomial-time verifiers.

The predicate is construction-independent: it can be instantiated by any of several succinct proof systems, and implementers should select constructions based on performance and trust-model constraints without altering the predicate's formal meaning. The predicate composes with three other primitives — Model Identity, Chained Integrity, Credentialed Review — to form the four-property AI Accountability Standard developed in companion work.

We have named the predicate's limitations explicitly: it is a pointwise rather than distributional property; it depends on the external specification of a reference-value function; it cannot ensure the completeness of the prohibited-inputs registry; it addresses inference-time rather than training-time behavior; it faces open engineering problems for large language models and long-retention quantum-secure deployments. None of these limitations undermine the predicate's value in the domain where it is intended to operate — per-decision evidentiary verification of structural compliance properties — but all of them should be understood by adopters who intend to rely on the predicate as part of a broader compliance posture.

The Separation Predicate is not a cryptographic novelty. The proof-system machinery it depends on is a decade of prior work in the SNARK and STARK literatures, combined with recent engineering that has made that machinery practical at production scales. The contribution of this paper is specifying the predicate precisely enough that implementations, critiques, and standards adoption can proceed against a stable reference. Whether the Separation Predicate becomes the technical substrate of verifiable AI governance across sectors is a question that adoption will decide. Our goal here has been to make the technical object well-defined enough that the adoption question can be answered on its merits.

The formal definition presented here is version 1.0. We anticipate refinements — tighter specifications, additional security properties, improved construction efficiency, extensions to new model classes — as the framework is adopted and pressure-tested. The definition is intended to be durable enough to reference without requiring frequent revision, and specific enough that implementations can be checked against it. Feedback from the cryptography, policy, regulatory, and deployment communities is welcomed and will inform future versions.

About this paper

This paper specifies the Separation Predicate at the formal-definition level intended for cryptographers, applied security researchers, standards-body technical working groups, regulatory technology staff, and engineering teams implementing compliance verification for consequential algorithmic decisioning systems. It is a companion to the policy and cross-sector papers in the series, which address the motivation and applications the predicate is designed to serve. Implementations of the Separation Predicate, and commercial deployments built around it, are available

from Lithosense LLC and, through open specification, from any party choosing to build against the definition given here. The Separation Predicate is a trademark of Lithosense LLC; the formal definition presented in this paper is made available for reference, critique, and implementation by standards bodies, regulators, and the research community.