
ZK-GovProof: Privacy-Preserving Verification of Regulatory Compliance in AI Governance

Srishti Dutta
Independent

With
Apart Research

Abstract

The governance of frontier artificial intelligence systems faces a fundamental verification paradox: regulatory bodies require evidence of compliance with compute thresholds, safety evaluations, and organizational commitments, yet AI laboratories legitimately resist disclosing information that could compromise competitive advantages or reveal security vulnerabilities. This tension has stalled progress on international AI coordination, undermined the enforceability of frameworks such as the EU AI Act, and reduced Responsible Scaling Policies to exercises in self-attestation. We present ZK-GovProof, a cryptographic framework that resolves this paradox through zero-knowledge succinct non-interactive arguments of knowledge (zkSNARKs)—enabling laboratories to demonstrate regulatory compliance without revealing the underlying sensitive data. Our system contributes three novel elements: working Groth16 circuits for compute thresholds, safety evaluation attestation, and policy compliance verification; a composable proof architecture that bundles multiple governance requirements into unified submission artifacts with $3\times$ reduction in verification overhead; and a systematic verifiability gap analysis that delineates the boundaries of what zero-knowledge cryptography can and cannot guarantee for AI governance. We validate our implementation against real regulatory frameworks—the EU AI Act Article 55 obligations for

GPAI models with systemic risk and Anthropic's Responsible Scaling Policy ASL-3 requirements—demonstrating end-to-end proof generation and verification. Our threat model identifies eight attack vectors, with particular attention to the fundamental limitations of input integrity and reporting completeness that necessitate complementary hardware attestation mechanisms. We argue that zero-knowledge proofs must constitute one layer within a defense-in-depth governance architecture rather than a standalone solution.

Keywords: *zero-knowledge proofs, zkSNARKs, AI governance, regulatory compliance, composable proofs, privacy-preserving verification, compute governance, verifiability analysis*

1. Introduction

The rapid advancement of frontier AI systems has precipitated urgent calls for governance mechanisms capable of ensuring their safe development and deployment. International bodies, national regulators, and AI laboratories themselves have proposed frameworks ranging from compute-based reporting thresholds to capability evaluations and organizational safety commitments [9], [11], [12]. Yet these proposals share a common vulnerability: they assume the existence of verification mechanisms that do not currently exist.

1.1 The Verification Paradox

This paper addresses what we term the verification paradox in AI governance—a three-way tension that currently blocks meaningful oversight:

- **Information asymmetry:** Effective oversight requires regulators to verify claims about training compute, safety evaluation completion, and adherence to stated policies. The EU AI Act mandates notification for general-purpose AI models trained with computational resources exceeding 10^{25} floating-point operations [9]. Anthropic's Responsible Scaling Policy requires comprehensive capability evaluations in CBRN, cyber, and autonomy domains before deploying models that approach dangerous capability thresholds [11]. Yet in both cases, compliance depends entirely on self-attestation.
- **Legitimate confidentiality interests:** The evidence necessary for verification—detailed compute logs, evaluation methodologies and results, internal process documentation—constitutes precisely the information that AI laboratories have strong incentives to protect. Exact compute figures reveal budgetary and strategic priorities that inform competitive positioning. Evaluation results may expose model vulnerabilities exploitable

by adversaries. Red team findings document attack vectors that must remain confidential. Internal processes embody proprietary methodologies developed through substantial R&D investment.

- **Governance collapse without verification:** Unverifiable agreements devolve into voluntary commitments. International coordination mechanisms—whether bilateral arrangements between AI Safety Institutes or multilateral treaty frameworks—cannot function without credible verification. The history of arms control demonstrates that verification capacity is prerequisite to meaningful agreement [21].

1.2 Zero-Knowledge Proofs as Resolution

Zero-knowledge proofs offer a cryptographic resolution to this paradox. First formalized by Goldwasser, Micali, and Rackoff [1], zero-knowledge proofs enable a prover to convince a verifier that a statement is true without revealing any information beyond the statement's validity. The mathematical guarantee is precise: the verifier's view of the interaction can be simulated without access to the prover's secret, implying that no information about the secret is leaked.

We employ Groth16 zkSNARKs [2], which produce constant-size proofs (~200 bytes) verifiable in milliseconds regardless of computation complexity, making them suitable for governance applications requiring efficient verification at scale. Groth16 operates over arithmetic circuits expressed as Rank-1 Constraint Systems (R1CS), where a prover demonstrates knowledge of private inputs satisfying the circuit's constraints without revealing those inputs. The system requires a trusted setup ceremony; compromise of the setup randomness ("toxic waste") would allow proof forgery, necessitating multi-party computation ceremonies with many independent participants for production deployment [22]. Our implementation uses the BN128 elliptic curve via circom [24] and snarkjs [23], with 254-bit field elements that comfortably accommodate governance values such as 10^{25} FLOP compute thresholds.

Applied to AI governance, this primitive enables statements such as:

- "Our training compute was below 10^{25} FLOPs" (without revealing the exact figure)
- "We completed all five required safety evaluations" (without revealing scores or findings)
- "We satisfied all eight RSP checklist items" (without revealing implementation details)

However, the application of zero-knowledge proofs to governance verification requires careful analysis. Not all governance claims can be expressed as the deterministic predicates that zero-knowledge systems require. The statement "our model is safe" involves behavioral properties that cannot be computed over static inputs. The claim "we acted in good faith" references mental states inaccessible to any computational verification. A rigorous analysis of which governance claims are

amenable to zero-knowledge verification—and which fundamental limitations remain—is essential for avoiding misplaced confidence in cryptographic solutions.

1.3 Contributions

This paper makes five contributions to the emerging field of technical AI governance:

1. Working zkSNARK circuits for governance predicates: We implement and validate three circuits in circom: compute threshold verification using 252-bit range proofs, safety evaluation attestation using binary flag aggregation, and policy compliance checking using summation constraints. All circuits compile to R1CS representations, generate Groth16 proofs, and verify correctly.

2. Composable governance proofs: We introduce a framework for bundling multiple compliance requirements into unified artifacts that enable atomic, all-or-nothing verification. For a three-requirement framework, this reduces verification overhead by $3\times$ while ensuring partial compliance cannot masquerade as full compliance.

3. Validation against real regulatory frameworks: We implement policy packs encoding EU AI Act Article 55 obligations for GPAI models with systemic risk and Anthropic's RSP ASL-3 requirements, demonstrating that our circuits can express real-world governance predicates.

4. Verifiability gap analysis: We provide a systematic taxonomy categorizing governance claims according to their amenability to zero-knowledge verification, identifying both applications and fundamental limitations.

5. Comprehensive threat model: We identify eight attack vectors specific to governance verification, with particular attention to input integrity and reporting completeness—limitations that require complementary mechanisms beyond cryptography.

2. Circuit Design and Implementation

2.1 Design Principles

Our circuits encode governance predicates as R1CS constraint systems satisfying four requirements:

- **Predicate enforcement:** The constraint system must be satisfiable if and only if the governance predicate holds. This is achieved through a "hard constraint" pattern: a signal ``valid`` is constrained to equal 1, ensuring that witness generation fails when the predicate is false.
- **Privacy preservation:** Private inputs (actual compute, evaluation flags, policy status) must not appear in public signals. Only the validity indicator and public parameters (thresholds, required counts) are revealed.

- **Minimal constraint count:** Proving time scales with constraint count. Circuits should use efficient primitives (binary decomposition, linear combinations) rather than expensive operations (divisions, comparisons with naive implementations).
- **Compositional structure:** Circuits should accept standardized input formats enabling composition into multi-requirement proofs.

2.2 Compute Threshold Circuit

The compute threshold circuit proves that a private compute value c falls strictly below a public threshold T :

Prove: $c < T$ where $c \in \text{private}$, $T \in \text{public}$

Circuit Architecture: The core challenge is implementing comparison in arithmetic circuits, which natively support only addition and multiplication. We use binary decomposition: represent the value $(T - c + 2^n)$ in binary and check the most significant bit.

The LessThan template operates as follows. For n -bit comparison:

1. Compute $s = c + 2^n - T$
2. Decompose s into binary: $s = \sum_i b_i \cdot 2^i$ where $b_i \in \{0,1\}$
3. The MSB b_n indicates whether $c < T$ (when $b_n = 0$)

Constraint Specification: The circuit accepts a private compute value and public threshold, passes them through a 252-bit LessThan comparator, and enforces that the output equals 1 via a hard constraint. If $\text{privateCompute} \geq \text{threshold}$, the comparator outputs 0, making the constraint unsatisfiable—witness generation fails with an assertion error rather than producing a proof. See Appendix D.1 for complete circuit implementation.

Binary Decomposition Constraints: The Num2Bits template enforces binary decomposition through two constraint families: (1) each bit b_i satisfies $b_i \times (b_i - 1) = 0$, ensuring $b_i \in \{0,1\}$; and (2) $\sum_i b_i \cdot 2^i = \text{input}$, ensuring bits correctly reconstruct the value.

Constraint Count: For 252-bit comparison, the circuit requires approximately 253 constraints: one per bit for binary enforcement, plus the reconstruction constraint.

2.3 Evaluation Attestation Circuit

The evaluation attestation circuit proves that at least k of N required evaluations were completed:

Prove: $\sum_{i=1}^N e_i \geq k$ where $e_i \in \{0,1\}$ are private, k is public

Circuit Architecture: The circuit accepts an array of private binary flags indicating completion status for each evaluation category. It enforces binary constraints on each flag, computes the sum, and verifies the sum meets the required threshold.

Constraint Specification: The circuit enforces three constraint families: (1) binary constraints ensuring each flag $e_i \times (e_i - 1) = 0$; (2) running sum computation via linear constraints; and (3) threshold comparison using GreaterEqThan with hard constraint $\text{valid} = 1$. See Appendix D.2 for complete circuit implementation.

Privacy Properties: The proof reveals only that the sum of flags meets the threshold—not which specific evaluations were completed. An observer seeing public signals $[1, 5]$ learns "at least 5 evaluations completed" but cannot determine the pattern (e.g., whether evaluation 3 was skipped and others doubled to compensate—though our binary constraints prevent this).

Instantiation: We instantiate with $N=5$ for the standard evaluation categories: CBRN, cyber capabilities, persuasion/manipulation, autonomy/autentic behavior, and deception/sandbagging—aligning with both EU AI Act Article 55 requirements and Anthropic RSP evaluation domains.

2.4 Policy Compliance Circuit

The policy compliance circuit proves that at least m of N policy checklist items were satisfied:

Prove: $\sum_{i=1}^N p_i \geq m$ where $p_i \in \{0,1\}$ are private, m is public

Circuit Architecture: Structurally identical to evaluation attestation, this circuit handles policy checklist items rather than evaluation categories. The distinction is semantic rather than cryptographic—both reduce to proving a sum of binary values exceeds a threshold.

Constraint Specification: The circuit follows the same pattern as evaluation attestation: binary constraints on each policy flag, running sum computation, and threshold comparison with hard constraint enforcement. See Appendix D.3 for complete circuit implementation.

Instantiation: We instantiate with $N=8$ for typical RSP checklist items: red teaming, external audit, incident reporting, capability thresholding, security measures, documentation, deployment review, and post-deployment monitoring.

2.5 Composable Governance Proofs

Real governance frameworks combine multiple requirements. The EU AI Act imposes compute thresholds (Article 51), safety evaluations (Article 55(1)(a)), and organizational measures (Article 55(1)(b-d)) simultaneously. A laboratory must demonstrate compliance with all applicable provisions.

Composition Architecture: We implement composable governance proofs through structured bundling. A composite proof artifact contains: (1) version and timestamp metadata for auditability; (2) a map of named proofs, each containing the Groth16 proof elements (π_a , π_b , π_c), public signals, and a human-readable claim description; and (3) submission metadata identifying the laboratory and applicable framework. See Appendix D.5 for the complete artifact schema.

Verification Protocol: The verifier processes each component proof:

1. Load verification key vk_i for circuit i
2. Extract proof π_i and public signals s_i from composite artifact
3. Verify: $Groth16.verify(vk_i, s_i, \pi_i)$ for each i
4. Accept if and only if all verifications succeed

Efficiency Analysis: Composition provides $3\times$ reduction in coordination overhead for three-requirement frameworks:

Metric	Individual Proofs	Composite Proof
Submissions	3	1
Verification calls	3	1 (bundled)
Atomicity	None (partial compliance possible)	Full (all-or-nothing)

Cryptographic Note: Our current implementation uses proof bundling rather than recursive SNARKs. True cryptographic aggregation—producing a single ~ 200 byte proof regardless of component count—would require recursive proof composition (e.g., using Halo2 or Nova). This represents important future work; our bundling approach demonstrates the governance benefits while using mature tooling.

3. Empirical Validation

3.1 Implementation Environment

We implemented ZK-GovProof using:

- *Circuit language:* circom 2.1.0

- *Proof system*: snarkjs 0.7.4 (Groth16)
- *Elliptic curve*: BN128 (alt_bn128)
- *Runtime*: Node.js 18+ with ES6 modules

The trusted setup used powers of tau ceremony artifacts (pot12_final.ptau) sufficient for circuits up to 4096 constraints. Production deployment would require larger ceremonies or migration to transparent setup systems.

3.2 Performance Characteristics

Table 1 presents measured performance across our three circuits:

Metric	Compute Threshold	Evaluation Attestation	Policy Compliance
Constraints	253	52	68
Proving time	18.3 sec	7.2 sec	8.1 sec
Verification time	0.4 sec	0.3 sec	0.3 sec
Proof size	192 bytes	192 bytes	192 bytes
Proving key	167 KB	11 KB	13 KB
Verification key	2.9 KB	2.8 KB	2.8 KB

Interpretation: Several observations merit attention:

- **Proof size is constant:** All proofs are exactly 192 bytes (three BN128 group elements), regardless of the underlying computation. This is a fundamental property of Groth16 enabling efficient storage and transmission.
- **Verification is fast:** Sub-second verification enables scalable regulatory workflows. A regulator could verify thousands of proofs per hour on commodity hardware.
- **Proving time is acceptable:** Proving times of 7-18 seconds are practical for compliance workflows operating on quarterly or annual cycles. Real-time verification is not required; laboratories generate proofs asynchronously and submit completed artifacts.
- **Constraint count determines proving time:** The compute threshold circuit (253 constraints) requires $\sim 2.5\times$ the proving time of evaluation attestation (52 constraints), reflecting the $O(n \log n)$ complexity of the FFT-based proving algorithm.

3.3 Functional Validation: Compliant Inputs

We validated correct proof generation for compliant inputs using a scenario where a laboratory trained a model using 8.5×10^{24} FLOPs (below the 10^{25} threshold), completed all five required evaluations, and satisfied all eight policy items. All three proofs generated successfully and verification returned true for each. The public signals confirmed predicate satisfaction: compute threshold proof returned [1, 10^{25}], evaluation attestation returned [1, 5], and policy compliance returned [1, 8]—where the leading 1 indicates validity and the second value confirms the public threshold parameter. See Appendix E.1 for complete input specifications and output details.

3.4 Functional Validation: Non-Compliant Inputs

We validated correct rejection of non-compliant inputs using a scenario with compute usage of 2×10^{25} FLOPs (exceeding the threshold). Proof generation failed during witness computation with an assertion error at the ``valid === 1`` constraint. Because the private compute exceeded the threshold, the `LessThan` comparator output 0, making the hard constraint unsatisfiable.

Critical observation: No proof is generated for non-compliant inputs. The system does not produce an "invalid" proof that fails verification; rather, it fails to produce any proof at all. This is the soundness property in action: false statements cannot be proven. See Appendix E.2 for complete test case details.

3.5 Case Study: EU AI Act Article 55 Compliance

We validated the system against real regulatory requirements by implementing a policy pack encoding EU AI Act Article 55 obligations for GPAI models with systemic risk [9], [10].

Framework Encoding: The policy pack maps Article 55 requirements to our three circuit types:

- Compute threshold (Article 51): The 10^{25} FLOP threshold above which GPAI models are presumed to have high-impact capabilities requiring systemic risk obligations.
- Required evaluations (Article 55(1)(a-d)): Five evaluation categories including adversarial testing for systemic risk identification, model evaluation using standardized protocols, systemic risk assessment at Union level, serious incident documentation capability, and cybersecurity evaluation.
- Policy checklist (Code of Practice Chapter 3): Eight organizational measures including safety framework establishment, risk inventory completion, pre-market assessment, external auditor engagement, model report preparation, post-market surveillance, incident reporting channels, and cybersecurity protections.

Results: The policy pack executed successfully, generating valid Groth16 proofs for all three requirements. The proof artifact enables the AI Office to verify compliance using only public verification keys and signals—confirming that compute was below threshold, all evaluations were completed, and all policy items were satisfied—without learning exact figures, specific evaluation outcomes, or implementation details. See Appendix E.3 for complete policy pack configuration and proof artifacts.

3.6 Case Study: Anthropic RSP ASL-3 Compliance

We similarly validated against Anthropic's Responsible Scaling Policy requirements for ASL-3 threshold evaluation [11].

Framework Encoding: The policy pack maps RSP v2.2 requirements to our circuit types:

- Compute threshold: Same 10^{25} FLOP threshold used as a proxy for capability level requiring ASL-3 safeguards.
- Required evaluations (RSP Section 4): Five capability evaluation domains—CBRN capabilities (Section 4.1), cybersecurity capabilities (Section 4.2), model autonomy (Section 4.3), persuasion and manipulation, and deception/sandbagging detection.
- Policy checklist (ASL-3 Security): Eight organizational commitments including Responsible Scaling Officer notification, internal access controls, model weight protection, compartmentalization controls, internal red teaming, external evaluation, capability threshold documentation, and anonymous reporting channels.

Results: The policy pack executed successfully, generating proofs with public signals confirming all requirements satisfied. The system also generates a human-readable report documenting what was cryptographically proven alongside an important caveat: the proofs verify predicates over structured inputs but do not establish that inputs are truthful measurements or that no unreported runs exist. See Appendix E.4 for complete policy pack configuration and generated reports.

4. Verifiability Gap Analysis

A critical contribution of this work is systematic analysis of what zero-knowledge proofs can and cannot verify for AI governance. We categorize claims into three groups based on their amenability to ZK verification.

4.1 Claims Amenable to ZK Verification

Zero-knowledge proofs excel at verifying deterministic predicates over well-defined inputs. Table 2 summarizes governance claims in this category:

Claim Type	Example	Circuit Approach	Status
Quantitative threshold	Compute $< 10^{25}$ FLOPs	Range proof via binary decomposition	✓ Implemented ▾
Binary completion	Evaluation was conducted	Flag verification with binary constraint	✓ Implemented ▾
Aggregate count	≥ 5 evaluations completed	Sum of binary flags vs. threshold	✓ Implemented ▾
Set membership	Model on approved list	Merkle tree inclusion proof	Feasible ▾
Range bounds	Score in $[0, 50]$	Two-sided range proof	Feasible ▾

Why these work: These claims reduce to arithmetic relations over field elements. The computation is deterministic—given the same inputs, the same output results. No external state, randomness, or subjective judgment is required.

4.2 Claims Not Amenable to ZK Verification

Certain governance claims cannot be verified through zero-knowledge proofs regardless of cryptographic sophistication. Table 3 summarizes these fundamental limitations:

Claim Type	Example	Why ZK Cannot Verify
Model capabilities	"Model cannot assist with bioweapons"	Behavioral property requiring model execution
Safety properties	"Model is safe to deploy"	Context-dependent, no formal specification
Intent	"We acted in good faith"	Mental states are not computable
Quality	"Evaluation was rigorous"	Subjective, lacks consensus metric

Completeness	"No unreported training runs"	Cannot prove non-existence
Future behavior	"Model will remain safe"	Proofs are point-in-time

The completeness problem deserves special attention. A ZK proof establishes that stated inputs satisfy stated predicates. It cannot establish that all relevant activities were reported. A laboratory could train ten models, report only the three compliant ones with valid proofs, and conceal the seven non-compliant runs. Zero-knowledge proofs verify properties of disclosed information; they cannot reveal what has been concealed.

This is not a limitation of our implementation, it is a fundamental property of proof systems. Addressing completeness requires mechanisms external to ZK: hardware attestation binding proof inputs to physical activities, datacenter monitoring providing independent activity records, or legal frameworks penalizing non-disclosure.

4.3 Partially Verifiable Claims

Some claims occupy an intermediate position, verifiable in a limited sense but requiring trust assumptions the verification itself cannot establish:

Claim	What ZK Verifies	What ZK Cannot Verify
Evaluation results	Score $<$ threshold	Benchmark measures what matters
Policy compliance	Checklist items completed	Items correspond to actual safety
Data provenance	Data from approved source	Source is actually trustworthy

Example Evaluation results: A laboratory can prove that a model scored below 50 on a CBRN capability benchmark. But this proof cannot establish that the benchmark meaningfully measures CBRN risk—perhaps the model performs poorly on the benchmark but excels at actual bioweapon synthesis through an unexpected pathway. ZK verification handles the arithmetic; domain expertise must assess whether the arithmetic captures the property of concern.

4.4 Implications for Governance Design

These limitations establish that zero-knowledge proofs cannot serve as a complete governance solution. We propose a defense-in-depth architecture where ZK constitutes one layer:

Layer 1: Zero-Knowledge Proofs: Privacy-preserving verification of quantitative thresholds and completion predicates. Addresses privacy concerns while providing cryptographic soundness.

Layer 2: Hardware Attestation: Chips cryptographically attest to computational activities, binding proof inputs to physical reality. Addresses input integrity and completeness problems.

Layer 3: Third-Party Audits: Human evaluators assess quality—whether safety measures are substantive, whether evaluations were rigorous, whether processes reflect genuine commitment. Addresses quality and intent limitations.

Layer 4: Continuous Monitoring: Ongoing observation of deployed systems detects post-certification changes and emergent behaviors. Addresses temporal limitations of point-in-time proofs.

Layer 5: Legal and Institutional Mechanisms: Penalties for non-compliance, whistleblower protections, and accountability structures. Addresses incentive alignment and deterrence.

No single layer suffices. ZK proofs without input integrity enable fabricated compliance. Hardware attestation without privacy requires sensitive disclosure. Audits without cryptographic verification scale poorly. The combination provides robust governance.

5. Threat Model and Security Analysis

5.1 Security Properties

ZK-GovProof provides three security guarantees under standard cryptographic assumptions:

Soundness: A computationally bounded adversary cannot generate a valid proof for a false statement. Formally, for any probabilistic polynomial-time adversary A :

$$\Pr[\text{Verify}(vk, x, \pi) = 1 \wedge x \notin L : (x, \pi) \leftarrow A(pk, vk)] \leq \text{negl}(\lambda)$$

This relies on the hardness of the discrete logarithm problem in BN128.

Zero-Knowledge: The proof reveals nothing about private inputs beyond predicate satisfaction. Any information computable from the proof is also computable from the public inputs alone.

Completeness: An honest prover with a valid witness can always generate an accepted proof.

5.2 Threat Analysis

We identify eight threat vectors relevant to governance verification:

T1: Dishonest Inputs (HIGH likelihood, HIGH impact): Laboratories provide false private inputs. Example: claiming 5×10^{24} FLOPs when actual usage was 2×10^{25} . The proof is cryptographically valid but the underlying claim is false.

Mitigation: Hardware attestation binding compute claims to chip-level measurements; energy consumption audits; third-party witnesses.

Residual risk: HIGH. This is the fundamental limit of self-reported ZK proofs.

T2: Selective Reporting (HIGH likelihood, HIGH impact): Laboratories report only compliant activities while concealing non-compliant ones.

Mitigation: Mandatory reporting requirements with legal penalties; hardware-level activity logging; datacenter monitoring.

Residual risk: HIGH. Requires complementary mechanisms beyond ZK.

T3: Compromised Trusted Setup (LOW likelihood, CRITICAL impact): Adversary obtains toxic waste τ from setup ceremony, enabling proof forgery.

Mitigation: Large MPC ceremonies (100+ participants); transparent setups (PLONK/STARKs); public ceremony auditing.

Residual risk: LOW with proper ceremony; CRITICAL if compromised.

T4: Circuit Logic Vulnerabilities (MEDIUM likelihood, HIGH impact): Implementation bugs create exploitable edge cases (overflow, incorrect constraints).

Mitigation: Formal verification of circuit correctness; independent security audits; extensive fuzzing; bug bounty programs.

Residual risk: MEDIUM. Requires rigorous testing before production.

T5: Privacy Extraction (LOW likelihood, MEDIUM impact): Malicious verifier attempts to extract private inputs from proofs.

Mitigation: Zero-knowledge property provides information-theoretic guarantee under honest-verifier assumption; computational ZK under malicious verifier.

Residual risk: LOW. Well-established cryptographic guarantees.

T6: Man-in-the-Middle (LOW likelihood, MEDIUM impact): Adversary intercepts and modifies proof submissions.

Mitigation: TLS for transport; digital signatures on proofs; nonces/timestamps in public inputs.

Residual risk: LOW with standard protocol hygiene.

T7: Side-Channel Attacks (LOW likelihood, LOW impact): Adversary extracts information through timing, power analysis, or electromagnetic emissions during proof generation.

Mitigation: Constant-time implementations in snarkjs; secure execution environments; air-gapped proving systems.

Residual risk: LOW. Modern crypto libraries implement countermeasures.

T8: Quantum Computing (LOW near-term likelihood, CRITICAL long-term impact): Quantum algorithms break discrete logarithm assumption underlying BN128.

Mitigation: Monitor quantum computing advances; migration path to post-quantum systems (STARKs); crypto-agile system design.

Residual risk: LOW (10-year horizon); increasing thereafter.

5.3 Risk Summary

The primary risks to ZK-based governance verification are non-cryptographic. Dishonest inputs (T1) and selective reporting (T2) represent the highest-likelihood, highest-impact threats—both exploit the fundamental limitation that ZK proofs verify computation over claimed inputs but cannot verify input authenticity. These require complementary mechanisms: hardware attestation for input binding, mandatory logging for completeness guarantees.

Cryptographic risks are lower but potentially catastrophic. A compromised trusted setup (T3) would enable proof forgery, necessitating large MPC ceremonies with many independent participants. Circuit vulnerabilities (T4) require formal verification and independent audits before production deployment.

Technical attacks (T5-T8)—privacy extraction, man-in-the-middle, side-channels, and quantum threats—are well-addressed by standard cryptographic practices and pose lower near-term risk.

6. Discussion

6.1 Contributions to Technical AI Governance

This work demonstrates that privacy-preserving verification of AI governance predicates is technically feasible today. Our implementation moves the conversation from theoretical possibility to practical deployment considerations. The circuits compile, proofs generate and verify correctly, and the system handles real regulatory requirements from the EU AI Act and Anthropic RSP.

The composable proof framework addresses a practical gap: governance frameworks involve multiple simultaneous requirements, and efficient verification of composite compliance is essential for scalability. Our $3\times$ reduction in verification overhead, combined with atomic all-or-nothing semantics, provides meaningful efficiency gains for regulatory workflows.

The verifiability gap analysis provides a necessary corrective to potential over-enthusiasm. By systematically cataloging limitations—particularly the fundamental problems of input integrity and reporting completeness—we help policymakers understand that ZK proofs are one tool among many, not a complete solution.

6.2 Future Work

In the near term, our priorities center on hardening the system for production readiness. Formal verification of circuit correctness using proof assistants such as Coq or Lean would provide mathematical guarantees that our constraint systems correctly encode the intended predicates. Independent security audits by specialized cryptography firms would identify vulnerabilities missed during development. We also plan to extend the circuit library with additional governance primitives for data provenance verification and model versioning attestation.

Over the medium term, we aim to advance both cryptographic capabilities and institutional integration. Recursive proof composition using systems like Nova or Halo2 would enable true cryptographic aggregation—producing a single constant-size proof regardless of how many governance requirements are bundled—rather than our current proof bundling approach. Integration with emerging hardware attestation protocols would address the input integrity limitations we have identified, binding proof inputs to physical computational activities. Standardization efforts through AI Safety Institutes would establish common formats and verification procedures across jurisdictions.

Looking further ahead, the long-term roadmap addresses evolving cryptographic requirements and deployment scale. Migration to post-quantum proof systems such as STARKs would ensure continued security as quantum computing capabilities advance. Submission of ZK-GovProof specifications to international standards bodies including ISO and NIST would facilitate global adoption. Ultimately, we envision production deployment partnerships with major AI laboratories,

establishing privacy-preserving compliance verification as standard practice in frontier AI governance.

7. Conclusion

This paper has presented ZK-GovProof, a system for privacy-preserving verification of AI governance compliance using Groth16 zkSNARKs. We demonstrated working circuits for compute thresholds, evaluation attestation, and policy compliance; introduced composable governance proofs for multi-requirement frameworks; validated against real regulatory requirements from the EU AI Act and Anthropic RSP; and provided systematic analysis of verification boundaries and security threats.

Our central argument is that zero-knowledge proofs can resolve the privacy-verification tradeoff in AI governance—enabling laboratories to prove compliance without revealing sensitive information—but cannot resolve the trust problem. ZK proofs verify that claimed inputs satisfy claimed predicates; they do not verify that claims correspond to reality or that all relevant activities have been disclosed.

The path forward requires both technical development and institutional coordination. Cryptographically, production deployment demands formal verification, large-scale trusted setup ceremonies, and integration with hardware attestation. Institutionally, governance frameworks must incorporate ZK verification as one component of comprehensive oversight, complemented by audits, monitoring, and legal accountability.

The AI governance challenges motivating this work are urgent. Frameworks are being designed and implemented now; verification mechanisms must be developed in parallel. We offer ZK-GovProof as a contribution toward governance infrastructure adequate to the moment—technically sound, honest about limitations, and ready for integration into the broader architecture of AI safety.

8. References

- [1] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal on Computing*, vol. 18, no. 1, pp. 186–208, 1989.
- [2] J. Groth, "On the size of pairing-based non-interactive arguments," in *Proc. EUROCRYPT 2016, Lecture Notes in Computer Science*, vol. 9666, pp. 305–326, Springer, 2016.
- [3] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, "SNARKs for C: Verifying program executions succinctly and in zero knowledge," in *Proc.*

CRYPTO 2013, Lecture Notes in Computer Science, vol. 8043, pp. 90–108, Springer, 2013.

[4] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity," in Proc. CRYPTO 2018, Lecture Notes in Computer Science, vol. 10993, pp. 701–732, Springer, 2018.

[5] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "PLONK: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge," Cryptology ePrint Archive, Report 2019/953, 2019.

[6] S. Bowe, J. Grigg, and D. Hopwood, "Halo: Recursive proof composition without a trusted setup," Cryptology ePrint Archive, Report 2019/1021, 2019.

[7] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," Communications of the ACM, vol. 59, no. 2, pp. 103–112, 2016.

[8] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, "From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again," in Proc. ITCS 2012, pp. 326–349, ACM, 2012.

[9] European Parliament and Council of the European Union, "Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act)," Official Journal of the European Union, L 1689, 2024.

[10] European Commission, "General-Purpose AI Code of Practice," Digital Strategy, 2025. [Online]. Available: <https://digital-strategy.ec.europa.eu/en/policies/contents-code-gpai>

[11] Anthropic, "Anthropic's Responsible Scaling Policy," Version 2.2, Oct. 2024. [Online]. Available: <https://www.anthropic.com/responsible-scaling-policy>

[12] OpenAI, "Preparedness Framework," Dec. 2023. [Online]. Available: <https://openai.com/preparedness>

[13] M. Anderljung et al., "Frontier AI regulation: Managing emerging risks to public safety," arXiv preprint arXiv:2307.03718, 2023.

[14] M. Brundage et al., "Toward trustworthy AI development: Mechanisms for supporting verifiable claims," arXiv preprint arXiv:2004.07213, 2020.

[15] G. Sastry et al., "Computing power and the governance of artificial intelligence," arXiv preprint arXiv:2402.08797, 2024.

[16] A. Dafoe, "AI governance: A research agenda," Governance of AI Program, Future of Humanity Institute, University of Oxford, 2018.

[17] METR, "Common elements of frontier AI safety policies," Model Evaluation and Threat Research, 2024. [Online]. Available: <https://metr.org/publications/>

- [18] A. Khan, J. Phelps, L. Heim, and T. Dunlap, "Technology to secure the AI chip supply chain," Center for a New American Security (CNAS), 2024.
- [19] E. Tromer, "Hardware-enabled governance for AI: Technical foundations and policy implications," Future of Life Institute FlexHEG Report, 2024.
- [20] International Scientific Report on the Safety of Advanced AI, UK Department for Science, Innovation and Technology, 2025.
- [21] T. C. Schelling, The Strategy of Conflict. Harvard University Press, 1960.
- [22] Zcash Foundation, "The Zcash ceremony," 2018. [Online]. Available: <https://z.cash/technology/paramgen/>
- [23] iden3, "snarkjs: JavaScript implementation of zkSNARK schemes," GitHub Repository, 2023. [Online]. Available: <https://github.com/iden3/snarkjs>
- [24] iden3, "circom: Circuit compiler for zkSNARKs," GitHub Repository, 2023. [Online]. Available: <https://github.com/iden3/circom>

9. Appendix

Code Repository link: https://github.com/srishtiii28/tech_ai_gov.git

Appendix Structure is as follows

Appendix	Content
A	Security Considerations (A.1-A.4 prose + A.5 detailed attack tree)
B	Proof Artifact Structure + full JSON proofs for both case studies
C	Reproduction Instructions (bash commands)
D	Circuit Implementations (D.1-D.5: all circom code + composite schema)
E	Validation Test Cases (E.1-E.4: inputs/outputs, policy packs, artifacts)

A: Security Considerations

A.1 Input Integrity

Zero-knowledge proofs verify computation correctness, not input authenticity. Production deployment MUST combine ZK proofs with hardware attestation binding proof inputs to physical computational activities.

A.2 Completeness of Reporting

ZK proofs verify properties of disclosed information but cannot detect undisclosed activities. Mitigation requires external completeness guarantees: hardware logging, datacenter monitoring, or legal penalties for omission.

A.3 Trusted Setup

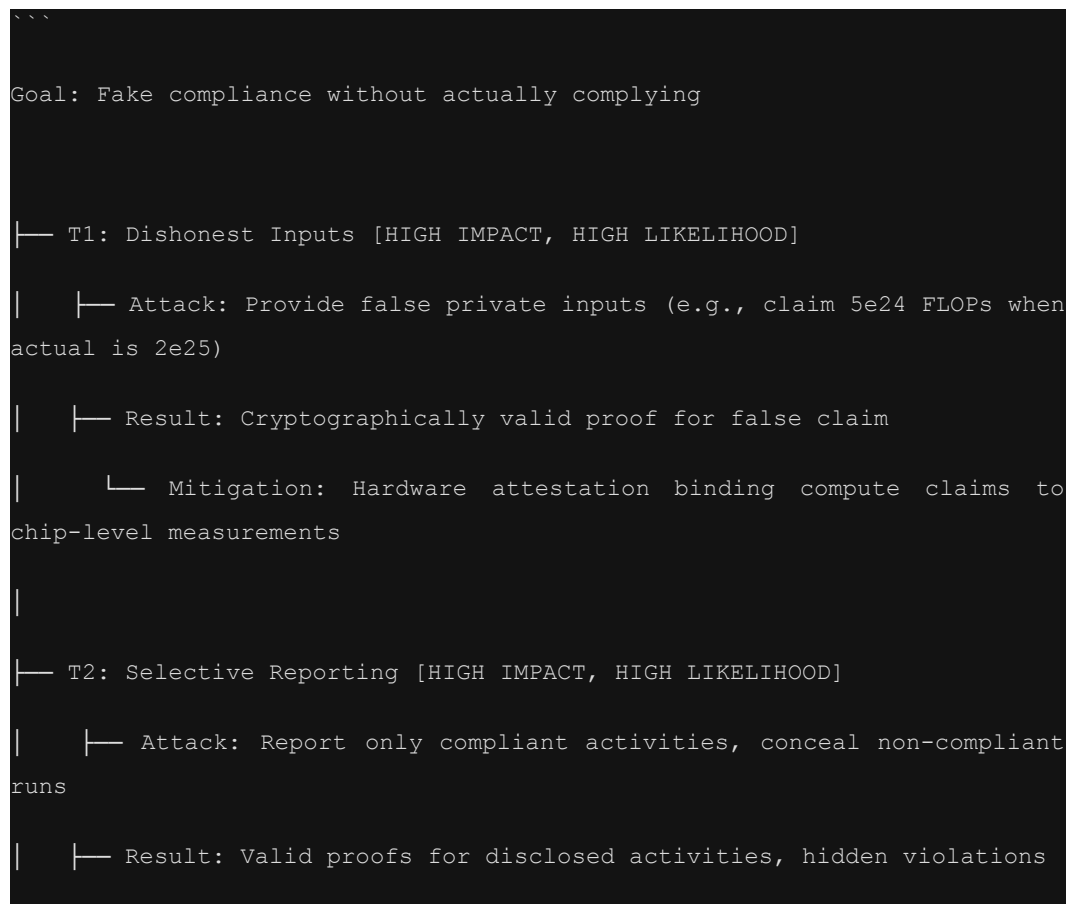
Groth16 requires trusted setup. Production deployment should use MPC ceremonies with 100+ participants or migrate to transparent setups (PLONK, STARKs).

A.4 Circuit Correctness

Production deployment requires formal verification and independent security audits of all circuits.

A.5 Attack Tree

Visual summary of threat vectors and mitigations for governance verification:



```

|   └─ Mitigation: Mandatory logging, datacenter monitoring, legal
penalties

|

|─ T3: Compromised Trusted Setup [CRITICAL IMPACT, LOW LIKELIHOOD]

|   └─ Attack: Obtain toxic waste  $\tau$  from setup ceremony

|   └─ Result: Ability to forge arbitrary proofs

|   └─ Mitigation: Large MPC ceremonies (100+ participants), transparent
setups

|

|─ T4: Circuit Logic Vulnerabilities [HIGH IMPACT, MEDIUM LIKELIHOOD]

|   └─ Attack: Exploit implementation bugs (overflow, incorrect
constraints)

|   └─ Result: Valid proofs for inputs that should fail

|   └─ Mitigation: Formal verification, independent audits, fuzzing

|

|─ T5: Privacy Extraction [MEDIUM IMPACT, LOW LIKELIHOOD]

|   └─ Attack: Extract private inputs from proof artifacts

|   └─ Result: Confidential data disclosure

|   └─ Mitigation: Zero-knowledge property (information-theoretic
guarantee)

|

|─ T6: Man-in-the-Middle [MEDIUM IMPACT, LOW LIKELIHOOD]

|   └─ Attack: Intercept and modify proof submissions

|   └─ Result: Tampered compliance records

|   └─ Mitigation: TLS, digital signatures, nonces/timestamps

|

```

```

└─ T7: Side-Channel Attacks [LOW IMPACT, LOW LIKELIHOOD]
  │
  │ └─ Attack: Extract information via timing, power, EM emissions
  │
  │ └─ Result: Partial private input leakage
  │
  │ └─ Mitigation: Constant-time implementations, secure environments
  │
└─ T8: Quantum Computing [CRITICAL IMPACT, LOW NEAR-TERM LIKELIHOOD]
  │
  │ └─ Attack: Break discrete logarithm assumption via quantum algorithms
  │
  │ └─ Result: Complete cryptographic compromise
  │
  │ └─ Mitigation: Migration path to post-quantum STARKs, crypto-agility
  │
  ...

```

Appendix B: Proof Artifact Structure

Groth16 proofs consist of three elliptic curve points (π_a , π_b , π_c) totaling 192 bytes, plus public signals confirming the verified predicates. Each proof artifact includes:

- π_a : Single G1 point (two 254-bit field elements plus projective coordinate)
- π_b : Single G2 point (two pairs of 254-bit field elements plus projective coordinate)
- π_c : Single G1 point (two 254-bit field elements plus projective coordinate)
- publicSignals: Array containing validity flag (1 if predicate satisfied) and public parameters
- protocol/curve: Metadata identifying Groth16 over BN128

B.1 EU AI Act Article 55 Compliance Proof

Generated: 2026-02-01T00:40:40.353Z

```
```json
```

```

{

 "compute_threshold": {

 "proof": {

 "pi_a": [

"124452904467293459266783161486262582660051038323770558409892891635519677",
66242",

"454828618705275197924567073800388544567243295706950536425369247799884273",
9153",

 "1"

],

 "pi_b": [

["14837737972524813415418882935421727839633143083642329388502116202694721",
834321",

"470985512391961716270883924441607650040796203602280749297938451967264343",
7287"],

["16172570296064938231141089443430666519415105755041026895771687811243926",
105983",

"179787945092861337649866300622338920807295542123325311480067652958796163",
87221"],

 ["1", "0"]

],

 "pi_c": [

```



```

"168146232334988912441249610067027147355724747816252000018753621500932832
46106",

 "1"

],

 "pi_b": [

["61761630029565716378435670557310799200698362515338899961819537733817874
404",

"293185534590234717072708090172150078959510297558574931679078281899375037
0016"],

["10630309567575793676383584481857589586456029872750503394752332375895429
542202",

"940523049207307470296383436568610375077168883849284830163847705355850091
589"],

 ["1", "0"]

],

 "pi_c": [

"104551565936820950405172998776015688769466374302716392023942854326294920
96313",

"108939198430263547355327850139471544184375176953852043185731592673086926
02775",

 "1"

],

 "protocol": "groth16",

```



```

],

 "pi_c": [

"104551565936820950405172998776015688769466374302716392023942854326294920
96313",

"108939198430263547355327850139471544184375176953852043185731592673086926
02775",

 "1"

],

 "protocol": "groth16",

 "curve": "bn128"

},

 "publicSignals": ["1", "5"]

},

 "policy_compliance": {

 "proof": {

 "pi_a": [

"124000345872917964206887062023207023710829899794739697651701345816714449
91069",

"122976926562329914962524954646567512078395367677532558517665731175310381
04821",

 "1"

],

 "pi_b": [

```

```

["12286829671414013920032381896414207383186587795594789626591417778206748
708894",

"110835161826625911411767605928858404356814856849449583592698623193087316
04057"],

["15841766575842806785672954790691725581080562119845155131744508467024351
964266",

"180759704547729904659074169355297504593034405991275908192354030746239476
60356"],

 ["1", "0"]

],

 "pi_c": [

"155276146125010062781867376714865673676049120028875079758068124679474239
22250",

"176468776106292769911479839644599172717760836619534854998007902635758818
97740",

 "1"

],

 "protocol": "groth16",

 "curve": "bn128"

},

 "publicSignals": ["1", "8"]

}

}

```

^^^

Note: The proofs are cryptographically distinct due to different private inputs, but the public signals confirm the same predicates are satisfied across both frameworks.

## Appendix C: Reproduction Instructions

```
^^`bash

Clone repository

git clone https://github.com/srishtiii28/tech_ai_gov.git

cd config_system

Install dependencies

npm install

Compile circuits (one-time, ~10 minutes)

npm run compile:circuits

Run end-to-end demo

npm run demo

Generate composite proof

node scripts/composite-proof.js

Run policy pack
```

```

npm run demo:policy-pack -- policies/eu_ai_act_article55_gpai_systemic.json

npm run demo:policy-pack -- policies/anthropic_rsp_asl3_threshold.json

...

```

## Appendix D: Circuit Implementations

This appendix provides complete circom implementations for all three governance circuits.

### D.1 Compute Threshold Circuit

```

```circom

pragma circom 2.0.0;

include "circomlib/circuits/comparators.circom";

template ComputeThreshold() {

    signal input privateCompute;    // Private: actual compute (field
element)

    signal input threshold;         // Public: regulatory threshold

    signal output valid;            // Output: constrained to 1

    component rangeCheck = LessThan(252);

    rangeCheck.in[0] <== privateCompute;

    rangeCheck.in[1] <== threshold;

```

```

    valid <== rangeCheck.out;

    valid === 1; // Hard constraint: forces failure if
c ≥ T
}

component main {public [threshold]} = ComputeThreshold();
...

```

D.2 Evaluation Attestation Circuit

```

```circom

pragma circom 2.0.0;

include "circomlib/circuits/comparators.circom";

template EvaluationAttestation(N) {

 signal input evaluationFlags[N]; // Private: completion flags

 signal input requiredCount; // Public: minimum required

 signal output valid;

 // Binary constraints: each flag must be 0 or 1

 for (var i = 0; i < N; i++) {

 evaluationFlags[i] * (evaluationFlags[i] - 1) === 0;

 }

 // Compute running sum

 signal completedSum[N];

 completedSum[0] <== evaluationFlags[0];

 for (var i = 1; i < N; i++) {

 completedSum[i] <== completedSum[i-1] + evaluationFlags[i];

 }
}

```

```

 // Verify sum ≥ required

 component check = GreaterEqThan(8);

 check.in[0] <== completedSum[N-1];

 check.in[1] <== requiredCount;

 valid <== check.out;

 valid === 1;
}

component main {public [requiredCount]} = EvaluationAttestation(5);

```

```

D.3 Policy Compliance Circuit

```

```circom

pragma circom 2.0.0;

include "circomlib/circuits/comparators.circom";

template PolicyCompliance(N) {

 signal input policyItems[N]; // Private: completion flags

 signal input minRequired; // Public: minimum items required

 signal output valid;

 // Binary constraints

 for (var i = 0; i < N; i++) {

 policyItems[i] * (policyItems[i] - 1) === 0;
 }
}
```

```

```

    }

    // Sum computation

    signal itemSum[N];

    itemSum[0] <== policyItems[0];

    for (var i = 1; i < N; i++) {

        itemSum[i] <== itemSum[i-1] + policyItems[i];

    }

    // Threshold check

    component check = GreaterEqThan(8);

    check.in[0] <== itemSum[N-1];

    check.in[1] <== minRequired;

    valid <== check.out;

    valid === 1;

}

component main {public [minRequired]} = PolicyCompliance(8);
```

```

## D.4 Binary Decomposition (Num2Bits)

The LessThan comparator uses binary decomposition via the Num2Bits template from circomlib:

```

```circom

template Num2Bits(n) {

    signal input in;

```

```

    signal output out[n];

    var lc1 = 0;

    var e2 = 1;

    for (var i = 0; i < n; i++) {

        out[i] <-- (in >> i) & 1;

        out[i] * (out[i] - 1) === 0; // Binary constraint

        lc1 += out[i] * e2;

        e2 = e2 + e2;

    }

    lc1 === in; // Reconstruction constraint
}
...

```

D.5 Composite Proof Artifact Schema

```

```json
{
 "version": "1.0",
 "timestamp": "2026-02-01T00:40:40.353Z",
 "proofs": {
 "compute_threshold": {
 "proof": { "pi_a": [...], "pi_b": [...], "pi_c": [...] },
 "publicSignals": ["1", "10000000000000000000000000000000"],
 "claim": "privateCompute < threshold"
 }
 }
}

```

```

 },

 "evaluation_attestation": {

 "proof": { "pi_a": [...], "pi_b": [...], "pi_c": [...] },

 "publicSignals": ["1", "5"],

 "claim": "sum(evaluationFlags) >= requiredCount"

 },

 "policy_compliance": {

 "proof": { "pi_a": [...], "pi_b": [...], "pi_c": [...] },

 "publicSignals": ["1", "8"],

 "claim": "sum(policyItems) >= minRequired"

 }

},

"metadata": {

 "submitter": "AI Laboratory",

 "framework": "EU AI Act Article 55"

}

}

...

```

## Appendix E: Validation Test Cases and Case Study Details

### E.1 Compliant Input Test Case

Test scenario: Laboratory with compute below threshold, all evaluations completed, all policy items satisfied.

```
```javascript
// Input specification

{
  privateCompute: "850000000000000000000000", // 8.5e24 FLOPs
  threshold: "1000000000000000000000000000", // 1e25 FLOPs
  evaluationFlags: ["1", "1", "1", "1", "1"], // All 5 completed
  requiredCount: "5",
  policyItems: ["1", "1", "1", "1", "1", "1", "1", "1"], // All 8
  satisfied
  minRequired: "8"
}

...

// Output

✓ Compute threshold proof generated successfully
✓ Evaluation attestation proof generated successfully
✓ Policy compliance proof generated successfully
✓ All verifications passed

Public signals:

- Compute: ["1", "1000000000000000000000000000"]
- Evaluation: ["1", "5"]
- Policy: ["1", "8"]

...
```
```

## E.2 Non-Compliant Input Test Case

Test scenario: Laboratory with compute exceeding threshold.

```
```javascript
// Input specification

{
  privateCompute: "2000000000000000000000000", // 2e25 FLOPs (EXCEEDS
threshold)

  threshold: "10000000000000000000000000000000" // 1e25 FLOPs
}
...
...

// Output

Error: Assert Failed. Error in template ComputeThreshold_2 line: 47
...
```
```

The circuit correctly rejects non-compliant inputs by failing at the hard constraint rather than producing a proof.

## E.3 EU AI Act Article 55 Policy Pack

Complete policy pack configuration:

```
Complete policy pack configuration:

```json
{
  "id": "eu_ai_act_article55_systemic_risk_v1",

```

```

    "title": "EU AI Act Article 55 - GPAI Models with Systemic Risk
Compliance",

    "legal_reference": "EU AI Act Article 55(1) (a)-(d), Article 51(1) (a)",

    "compute_threshold": {

        "public_threshold_flops": "1000000000000000000000000",

        "description": "Article 51(2): GPAI model presumed to have high impact
capabilities when training compute exceeds 10^25 FLOPs"

    },

    "required_evaluations": {

        "N": 5,

        "evaluation_names": [

            "Article 55(1) (a): Adversarial testing for systemic risk
identification",

            "Article 55(1) (a): Model evaluation using standardised protocols",

            "Article 55(1) (b): Systemic risk assessment at Union level",

            "Article 55(1) (c): Serious incident documentation and reporting
capability",

            "Article 55(1) (d): Cybersecurity evaluation of model and
infrastructure"

        ]

    },

    "policy_checklist": {

        "N": 8,

        "items": [

            "Safety and Security Framework established (Code of Practice 3.1)",

            "Risk inventory and scenario analysis completed (Code of Practice
3.2)",

```

```

        "Pre-market systemic risk assessment performed (Code of Practice
3.3)",

        "External evaluator or security auditor engaged (Code of Practice
3.4)",

        "Safety and Security Model Report prepared for AI Office (Code of
Practice 3.5)",

        "Post-market surveillance process established (Code of Practice
3.6)",

        "Serious incident reporting channel to AI Office operational",

        "Cybersecurity protection measures for model weights implemented"

    ]

}

}

...

```

Execution output:

```

...

$ npm run demo:policy-pack --
policies/eu_ai_act_article55_gpai_systemic.json

✔ Policy pack executed.

📦 Wrote:
data/policy_packs/eu_ai_act_article55_systemic_risk_v1_2026-02-01/artifac
ts.json

📝 Wrote:
data/policy_packs/eu_ai_act_article55_systemic_risk_v1_2026-02-01/report.
md

```

```

Generated proof artifact (compute threshold component):

```json
{
 "proof": {
 "pi_a": [
 "124452904467293459266783161486262582660051038323770558409892891635519677",
 "66242",
 "454828618705275197924567073800388544567243295706950536425369247799884273",
 "9153",
 "1",
],
 "pi_b": [
 "14837737972524813415418882935421727839633143083642329388502116202694721",
 "834321",
 "470985512391961716270883924441607650040796203602280749297938451967264343",
 "7287"],
 "16172570296064938231141089443430666519415105755041026895771687811243926",
 "105983",
 "179787945092861337649866300622338920807295542123325311480067652958796163",
 "87221"],

```



```

"required_evaluations": {
 "N": 5,
 "evaluation_names": [
 "CBRN capabilities evaluation (RSP Section 4.1)",
 "Cybersecurity capabilities evaluation (RSP Section 4.2)",
 "Model Autonomy evaluation (RSP Section 4.3)",
 "Persuasion and manipulation evaluation",
 "Deception and sandbagging evaluation"
]
},
"policy_checklist": {
 "N": 8,
 "items": [
 "Responsible Scaling Officer notified of evaluation results",
 "Internal access controls and clearance levels implemented (ASL-3 Security)",
 "Model weight protection measures active (ASL-3 Security)",
 "Multi-tier compartmentalization controls based on asset type",
 "Red team evaluation completed by internal safety team",
 "External red team or third-party evaluation conducted",
 "Capability threshold determination documented and reviewed",
 "Anonymous staff reporting channel for RSP noncompliance operational"
]
}
}

```

```
Execution output:

$ npm run demo:policy-pack -- policies/anthropic_rsp_asl3_threshold.json

✓ Policy pack executed.

📦 Wrote:
data/policy_packs/anthropic_rsp_asl3_compliance_v1_2026-02-01/artifacts.json

📝 Wrote:
data/policy_packs/anthropic_rsp_asl3_compliance_v1_2026-02-01/report.md
```

What was proven (cryptographically)

- 1) Compute threshold
  - Claim:  $\text{privateCompute} < \text{threshold}$  ( $10^{25}$  FLOPs)
  - Public signals: ["1", "1000000000000000000000000000000"]
- 2) Evaluation completion
  - Claim:  $\text{sum}(\text{evaluationFlags}) \geq \text{requiredCount}$  (5)
  - Public signals: ["1", "5"]
- 3) Policy checklist completion
  - Claim:  $\text{sum}(\text{policyItems}) \geq \text{minRequired}$  (8)
  - Public signals: ["1", "8"]

*Important caveat:* This run demonstrates ZK verification of an explicit predicate over structured inputs. It does NOT prove that the private inputs are truthful measurements of reality or that no other unreported runs exist.