

# Proof-of-Training Verifier: Efficient Black-Box Model Identity Verification with Anytime Confidence Sequences

Anonymous Authors

## Abstract

We present a post-training behavioral verifier for model identity that decides whether two models are behaviorally equivalent under black-box query access. Our method combines cryptographically pre-committed challenges with anytime-valid Empirical-Bernstein confidence sequences [? ? ?] to achieve sample-efficient verification with controlled error rates. The verifier outputs SAME, DIFFERENT, or UNDECIDED decisions based on statistical hypothesis testing with early stopping. In experiments on 8 model pairs (ranging from 70M to 7B parameters), the method achieves perfect separation using 14–40 queries at  $\alpha = 0.01$  confidence, representing a 96% reduction compared to fixed-sample baselines requiring 1000 queries [?]. Decision times range from 17–92 seconds for small models to 22 minutes for 7B models with memory-constrained shard loading, compared to 45–60 minutes for baseline approaches. The cryptographic pre-commitment scheme prevents adaptive attacks, while the anytime-valid confidence sequences maintain statistical validity under optional stopping.

## 1 Introduction

The deployment of large language models increasingly relies on API access or weight-inaccessible checkpoints, creating challenges for stakeholders who need to verify that a deployed model matches an audited reference. This verification problem arises in regulatory compliance, model governance, and supply chain security contexts where behavioral equivalence must be established without white-box access to model parameters.

Existing approaches face fundamental trade-offs. Weight-based verification [? ?] requires full parameter access, making it unsuitable for API-only settings. Gradient-based proof-of-learning [?] similarly requires white-box access and incurs  $O(\text{model\_size})$  memory overhead. Fixed behavioral test sets [? ?] lack statistical guarantees and are vulnerable to overfitting when challenges are known in advance.

We propose a black-box verifier that addresses these limitations through three technical contributions:

1. *Cryptographic pre-commitment*: Challenges are derived from HMAC-based pseudorandom generation [?] with revealed keys, preventing adaptive selection while enabling third-party reproducibility.
2. *Anytime-valid statistical testing*: Empirical-Bernstein confidence sequences [? ?] with time-uniform alpha-spending enable early stopping while maintaining validity, reducing queries from 1000+ to 14–40 in practice.
3. *Memory-efficient verification*: Shard-based loading for large models (tested up to 34B parameters, 206GB) enables verification on commodity hardware with <10GB active memory.

The combination of these techniques yields a practical verifier suitable for continuous integration workflows, with decision latencies under 2 minutes for models up to 2.7B parameters.

## 1.1 Problem Formulation

Let  $M_{\text{ref}}$  and  $M_{\text{cand}}$  be two language models accessible only through a query interface  $f_M : \mathcal{X} \rightarrow \Delta(\mathcal{Y})$  that maps input prompts to distributions over next tokens. We seek to test the hypothesis  $H_0 : M_{\text{ref}} \equiv M_{\text{cand}}$  (behavioral equivalence) against  $H_1 : D(M_{\text{ref}}, M_{\text{cand}}) > \delta$  for some divergence measure  $D$  and threshold  $\delta > 0$ .

The verifier must decide among three outcomes:

- SAME: Accept  $H_0$  with confidence  $1 - \alpha$
- DIFFERENT: Reject  $H_0$  with confidence  $1 - \alpha$
- UNDECIDED: Insufficient evidence for either decision at sample limit  $n_{\max}$

Under the constraint that challenges must be pre-committed before any model queries to prevent adaptive attacks.

## 2 Related Work

### 2.1 Model Verification

**White-box methods.** Parameter checksums provide instant verification but require full weight access. Neural network watermarking [? ?] embeds verification signals during training but assumes verifier control of the training process. Proof-of-learning [?] verifies training integrity through gradient checkpoints but requires  $O(\text{parameters})$  memory and white-box access.

**Behavioral methods.** Robustness benchmarks [? ?] use fixed test sets to evaluate model behavior but lack statistical guarantees for identity verification. Adversarial example transfer [?] can distinguish models but does not provide confidence bounds. Our work extends behavioral testing with anytime-valid statistical guarantees and cryptographic pre-commitment.

### 2.2 Sequential Testing

Wald’s Sequential Probability Ratio Test (SPRT) [?] established the foundation for early-stopping hypothesis tests. Recent work on anytime-valid inference [? ? ?] provides time-uniform confidence sequences that remain valid under optional stopping. Empirical-Bernstein bounds [? ?] achieve variance-adaptive concentration for bounded random variables.

We combine these techniques with cryptographic commitment to enable sample-efficient model verification with pre-committed challenges, addressing both statistical validity and security requirements simultaneously.

## 3 Method

### 3.1 Pre-committed Challenge Generation

To prevent adaptive attacks where an adversary observes early challenges before responding, we use HMAC-based deterministic generation [?]:

$$s_i = \text{HMAC}_K(\text{run\_id} || i) \quad (1)$$

$$\text{prompt}_i = \text{template}(\text{hash}(s_i) \bmod |\text{templates}|) \quad (2)$$

The verifier publishes the run metadata (run\_id, challenge count, seed hash) before querying models, then reveals the key  $K$  after collecting all responses. Third parties can regenerate the exact challenge sequence to verify the commitment was honored.

### 3.2 Behavioral Divergence Scoring

For each challenge prompt  $x_i$ , we compute teacher-forced cross-entropy divergence over the next  $K$  token positions:

$$X_i = \frac{1}{K} \sum_{k=1}^K \left| H(p_{\text{ref}}^{(k)}, p_{\text{cand}}^{(k)}) - H(p_{\text{ref}}^{(k)}, p_{\text{ref}}^{(k)}) \right| \quad (3)$$

where  $p_M^{(k)}$  is model  $M$ 's distribution over the  $k$ -th next token. The delta formulation (cross-entropy against self) controls for inherent uncertainty, yielding non-negative scores with  $X_i = 0$  for identical models.

We clip scores to  $[0, 1]$  for numerical stability in the confidence sequence construction, though raw divergences may exceed 1 for highly different models.

### 3.3 Anytime Empirical-Bernstein Confidence Sequences

Let  $\bar{X}_n = \frac{1}{n} \sum_{i=1}^n X_i$  and  $\hat{\sigma}_n^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X}_n)^2$  denote the sample mean and variance. An Empirical-Bernstein half-width is:

$$h_n = \sqrt{\frac{2\hat{\sigma}_n^2 \log(1/\delta_n)}{n}} + \frac{7 \log(1/\delta_n)}{3(n-1)} \quad (4)$$

With alpha-spending  $\delta_n = \frac{2\alpha}{n(n+1)}$ , we have  $\sum_{n=2}^{\infty} \delta_n = \alpha$  (telescoping series). This yields time-uniform coverage:

$$\mathbb{P}(\forall n \geq 2 : \mu \in [\bar{X}_n - h_n, \bar{X}_n + h_n]) \geq 1 - \alpha \quad (5)$$

The confidence interval  $CI_n = [\bar{X}_n - h_n, \bar{X}_n + h_n]$  remains valid regardless of when we stop [? ].

### 3.4 Decision Rules

Define relative margin error  $RME_n = h_n / \max(|\bar{X}_n|, \epsilon)$  with  $\epsilon = 10^{-10}$  for numerical stability. We decide at step  $n$ :

- **SAME** if  $CI_n \subseteq [-\gamma, \gamma]$  and  $h_n \leq \eta\gamma$
- **DIFFERENT** if  $|\bar{X}_n| \geq \delta^*$  and  $RME_n \leq \epsilon_{\text{diff}}$
- **UNDECIDED** otherwise (or if  $n = n_{\text{max}}$ )

Parameters are set through calibration:  $\gamma = 0.025$  (equivalence margin, below perceptual threshold [?]),  $\delta^* = 0.05$  (minimum effect size),  $\eta = 0.5$  (signal-to-noise requirement),  $\epsilon_{\text{diff}} = 0.3$  (relative error tolerance).

### 3.5 API Verification and Provider Authentication

The verifier establishes behavioral equivalence through API queries but cannot authenticate the service provider without additional infrastructure. Provider authentication requires either:

- Trusted Execution Environment (TEE) attestation [? ] proving code/data integrity
- Vendor cryptographic signatures committing to the serving model
- Zero-knowledge proofs [? ? ] of verifier computation (proves correctness of decision from transcript, but not endpoint identity)

## 4 Implementation

### 4.1 Evidence Bundle Format

Each verification run produces a structured output directory:

- *manifest.yaml*: Configuration, commitment metadata, revealed HMAC key
- *transcript.ndjson*: Per-challenge prompts, model outputs, scores
- *evidence\_bundle.json*: Decision, confidence interval, sample size, bundle hash
- *metrics.json*: Memory usage, timing breakdown, system metadata

The bundle hash (SHA-256 of manifest + transcript + evidence) enables reproducible verification.

### 4.2 Memory-Efficient Verification

For models exceeding available RAM, we implement shard-based loading:

1. Partition model weights into shards of size  $S$  (typically 4–10GB)
2. For each query: load shard  $i$ , compute activations, release shard  $i$ , load shard  $i + 1$
3. Cycle through shards while maintaining cumulative decision statistics

Peak memory usage is  $\max(S, \text{activation\_memory})$ , enabling verification of 206GB models on 64GB hosts (measured 52% peak RAM usage for Yi-34B).

## 5 Experimental Setup

### 5.1 Models and Baselines

We evaluate on model pairs ranging from 70M to 7B parameters:

- Small (<1B): GPT-2 (124M), DistilGPT-2 (82M), DialoGPT-Medium (345M), Pythia-70M, GPT-Neo-125M
- Large (7B): Llama-2-7b-hf, Llama-2-7b-chat-hf

Baselines include:

- Fixed- $N$  with  $N = 1000$  (standard practice [? ])
- Mixture SPRT (mSPRT) [? ] without pre-commitment
- Always-valid  $p$ -values [? ]

## 5.2 Evaluation Protocol

For each model pair, we run:

1. Self-consistency test (model vs. itself, expect SAME)
2. Cross-model test (model A vs. model B, expect DIFFERENT if architecturally distinct)
3. Fine-tuning detection (base vs. instruction-tuned variant)

We report: (i) decision accuracy (SAME/DIFFERENT), (ii) queries to decision, (iii) wall-clock time, (iv) peak memory usage. All experiments use  $\alpha \in \{0.01, 0.025\}$  corresponding to AUDIT and QUICK modes.

## 6 Results

### 6.1 Query Efficiency

Table 1 summarizes verification results. The method achieves perfect separation (8/8 correct decisions) using 14–40 queries, compared to 1000 for fixed-sample baselines.

Table 1: Verification results on 8 model pairs. All decisions correct (8/8).

Models	Mode	Queries	Time (s)	Decision	Memory (GB)
<i>Self-consistency verification</i>					
pythia-70m → pythia-70m	AUDIT	30	66.9	SAME	1.27
gpt2 → gpt2	AUDIT	30	71.7	SAME	1.56
llama-7b-base → base	QUICK	14	1346.7	SAME	8.01
llama-7b-chat → chat	QUICK	14	1381.4	SAME	7.95
<i>Architecture and scale differences</i>					
gpt2 → distilgpt2	AUDIT	32	92.2	DIFFERENT	1.33
gpt2 → gpt2-medium	AUDIT	40	84.6	DIFFERENT	1.71
gpt-neo → pythia	AUDIT	32	133.3	DIFFERENT	2.36
<i>Fine-tuning detection</i>					
dialogpt → gpt2	QUICK	16	17.3	DIFFERENT	1.85

### 6.2 Baseline Comparison

Table 2 compares against sequential testing baselines. Our method reduces queries by 3–6× compared to mSPRT and 4–7× compared to always-valid  $p$ -values, while uniquely providing cryptographic pre-commitment.

### 6.3 Computational Requirements

Wall-clock timing breaks down as: (i) model loading (5–10s for small models), (ii) inference (0.5–2.5 s/query), (iii) verifier overhead (<1% of total). For 7B models with sharding, shard loading dominates (contributing 95% of wall-clock time).

Memory usage scales with model size: 1.3–2.4GB for models under 1B parameters, 7.9–8.0GB for 7B models with active shard caching.

Table 2: Comparison with sequential testing baselines on GPT-2 models.

Method	Queries	Time (min)	Pre-commit	FAR/FRR
Fixed- $N$ (1000) [?]	1000	45–60	No	0.05/0.05
mSPRT [?]	87–142	4–7	No	0.08/0.06
Always-valid $p$ [?]	95–180	5–9	No	0.05/0.05
<b>Ours</b>	<b>14–40</b>	<b>0.3–1.5</b>	<b>Yes</b>	<b>0/0*</b>

\*Empirical error rate: 0/8 decisions incorrect

## 7 Discussion

### 7.1 Theoretical Guarantees

The method provides:

- Type-I error control:  $\mathbb{P}(\text{SAME}|H_1) \leq \alpha$  (anytime-valid CI)
- Type-II error control:  $\mathbb{P}(\text{DIFFERENT}|H_0) \leq \beta$  (power depends on  $n_{\max}$ , effect size)
- Pre-commitment security: Adversary cannot adapt responses to observed challenges (HMAC pre-commitment)

The UNDECIDED outcome occurs when evidence is insufficient at budget  $n_{\max}$ , corresponding to effect sizes in the indifference zone  $[\gamma, \delta^*]$ .

### 7.2 Limitations

**Scope of experiments.** Our evaluation focuses on models up to 7B parameters due to computational constraints. Generalization to larger models (70B+) requires validation, though the method’s sample efficiency should improve with larger models (more distinctive behavioral signatures).

**Provider authentication.** The verifier establishes behavioral equivalence but cannot authenticate the API endpoint without additional infrastructure (TEEs, vendor commitments).

**Semantic drift.** The method detects distributional differences but may not capture semantic changes that preserve token-level distributions (e.g., factual accuracy degradation).

### 7.3 Comparison to Prior Work

Compared to proof-of-learning [?], our method trades white-box gradient access for black-box efficiency, achieving practical verification latencies (<2 minutes vs. hours). Compared to fixed test sets [?], we provide statistical guarantees and 96% query reduction through adaptive early stopping. The combination of pre-commitment, anytime validity, and memory efficiency is novel.

## 8 Conclusion

We presented a black-box model verifier combining cryptographic pre-commitment, anytime-valid confidence sequences, and memory-efficient inference to achieve sample-efficient identity verification. Experiments demonstrate 96% query reduction compared to fixed-sample baselines while maintaining perfect decision accuracy (8/8 model pairs). The method enables practical continuous verification workflows, with decision latencies under 2 minutes for models up to several billion parameters.

Future work includes: (i) evaluation on larger models (70B+), (ii) integration with TEE-based provider authentication, (iii) extension to multi-model verification scenarios.

## References

- [] Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. Exploration-exploitation tradeoff using variance estimates in multi-armed bandits. In *Conference on Learning Theory*, pages 13–1, 2009.
- [] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *23rd USENIX Security Symposium*, pages 781–796, 2014.
- [] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE symposium on security and privacy (SP)*, pages 315–334. IEEE, 2018.
- [] Victor Costan and Srinivas Devadas. Intel sgx explained. In *Cryptology ePrint Archive*, 2016.
- [] Sebastian Gehrman, Hendrik Strobelt, and Alexander Rush. Gltr: Statistical detection and visualization of generated text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 111–116, 2019.
- [] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.
- [] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8340–8349, 2021.
- [] Steven R Howard, Aaditya Ramdas, Jon McAuliffe, and Jasjeet Sekhon. Confidence sequences for mean, variance, and median. *Proceedings of the National Academy of Sciences*, 118(15), 2021.
- [] Steven R Howard, Aaditya Ramdas, Jon McAuliffe, and Jasjeet Sekhon. Time-uniform, nonparametric, nonasymptotic confidence sequences. *The Annals of Statistics*, 49(2):1055–1080, 2021.
- [] Hengrui Jia, Mohammad Yaghini, Christopher A Choquette-Choo, Natalie Dullerud, Anvith Thudi, Varun Chandrasekaran, and Nicolas Papernot. Proof-of-learning: Definitions and practice. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1039–1056. IEEE, 2021.
- [] Ramesh Johari, Pete Koomen, Leonid Pekelis, and David Walsh. Peeking at a/b tests: Why it matters, and what to do about it. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1517–1525, 2017.
- [] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. Hmac: Keyed-hashing for message authentication. Technical report, RFC 2104, 1997.
- [] Andreas Maurer and Massimiliano Pontil. Empirical bernstein bounds and sample variance penalization. *arXiv preprint arXiv:0907.3740*, 2009.
- [] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. In *arXiv preprint arXiv:1605.07277*, 2016.
- [] Aaditya Ramdas, Peter Grünwald, Vladimir Vovk, and Glenn Shafer. Game-theoretic statistics and safe anytime-valid inference. *Statistical Science*, 38(4):576–601, 2023.

- [] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin’ichi Satoh. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on international conference on multimedia retrieval*, pages 269–277, 2017.
- [] Abraham Wald. Sequential tests of statistical hypotheses. *The annals of mathematical statistics*, 16(2):117–186, 1945.
- [] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia conference on computer and communications security*, pages 159–172, 2018.

## A Technical Details

### A.1 Alpha-Spending and Optional Stopping

**$\alpha$ -Spending Schedule:** We use  $\delta_n = \frac{\alpha \cdot c}{n(n+1)}$  with  $c = 2$  to ensure  $\sum_{n \geq 2} \delta_n = \alpha$  for time-uniform type-I error control under optional stopping.

**Proof Sketch:**

1. By telescoping:  $\sum_{n=2}^{\infty} \frac{c}{n(n+1)} = c \sum_{n=2}^{\infty} \left( \frac{1}{n} - \frac{1}{n+1} \right) = c \cdot 1 = c$
2. Setting  $c = 2$  and  $\delta_n = \frac{\alpha \cdot 2}{n(n+1)}$  yields  $\sum_{n \geq 2} \delta_n = \alpha$
3. The EB bound with this schedule satisfies  $\mathbb{P}(\exists n \geq 2 : |\bar{X}_n - \mu| > h_n) \leq \alpha$
4. This holds *anytime*, even under data-dependent stopping (optional stopping theorem)
5. The confidence sequence  $[\bar{X}_n \pm h_n]$  maintains coverage uniformly over all  $n$
6. Early stopping at any  $\tau$  preserves validity:  $\mathbb{P}(|\bar{X}_{\tau} - \mu| > h_{\tau}) \leq \alpha$

This construction enables valid inference regardless of when we stop, crucial for adaptive early termination.

### A.2 Evidence Bundle Schema

**Bundle Structure:** Each run produces a directory with cryptographic commitments, raw transcripts, and decisions. Bundle hash = SHA-256(manifest + transcript + evidence).

**Directory structure:**

```
runs/val_20250825_142945/
|- manifest.yaml                                # Run config, HMAC key (revealed post-run)
|- transcript.ndjson                            # Per-query: {prompt, outputs, scores}
|- evidence_bundle.json                         # Decision, CI, n_used, bundle_hash
|- metrics.json                                 # (Optional) RSS, timing, sharding events
```

**Key JSON fields** (evidence\_bundle.json):

```
{
  "decision": "SAME|DIFFERENT|UNDECIDED",
  "confidence_interval": [lower, upper],
  "n_queries": 14,
```

```

    "mean_effect": 0.001,
    "bundle_hash": "sha256:abc123...",
    "timestamp": "2025-08-25T14:29:45Z"
}

```

Reviewers can verify: (1) bundle hash matches table entry, (2) transcript reproduces scores, (3) HMAC seeds are deterministic.

### A.3 Statistical Guarantees Under Early Stopping

The Empirical-Bernstein confidence sequence maintains validity under optional stopping through careful  $\alpha$ -spending:

**Theorem (Anytime Validity):** For confidence sequence  $C_n = [\bar{X}_n \pm h_n]$  with

$$h_n = \sqrt{\frac{2\hat{\sigma}_n^2 \log(2/\delta_n)}{n}} + \frac{7 \log(2/\delta_n)}{3(n-1)}$$

and  $\delta_n = \frac{2\alpha}{n(n+1)}$ , we have for any stopping time  $\tau$ :

$$\mathbb{P}(\mu \notin C_\tau) \leq \alpha$$

This enables aggressive early stopping without inflating type-I error, crucial for achieving the reported query reductions.

### A.4 Behavioral Fingerprinting Algorithm

The behavioral fingerprinting classification triggers when:

1.  $n \geq \max(50, 2 \times n_{\min})$  (sufficient samples)
2.  $\text{CV} = \frac{\sigma}{|\mu|} < 0.1$  (stable convergence)
3.  $\text{RME} > \epsilon_{\text{diff}}$  (cannot meet DIFFERENT threshold)

**Classification thresholds:**

Relationship	Mean Effect Range
NEAR_CLONE	$ \bar{X}_n  < 0.001$
RELATED_TRAINING	$0.001 \leq  \bar{X}_n  < 5$
DIFFERENT_TRAINING	$5 \leq  \bar{X}_n  < 10$
DIFFERENT_ARCH	$ \bar{X}_n  \geq 10$

### A.5 Implementation Details

**Challenge Generation:** Prompts are generated via HMAC-SHA256 with revealed key:

- $\text{seed}_i = \text{HMAC}(\text{key}, \text{"challenge\_"} || i)$
- $\text{prompt}_i = \text{select\_prompt}(\text{seed}_i \bmod \text{num\_templates})$
- $\text{position}_i = (\text{seed}_i \gg 32) \bmod \text{context\_length}$

**Scoring Function:** KL divergence between output distributions:

$$S(P, Q) = \sum_{v \in V} P(v) \log \frac{P(v)}{Q(v) + \epsilon}$$

where  $V$  is the vocabulary,  $\epsilon = 10^{-10}$  for numerical stability.

**Memory Management for Large Models:**

- Models > 5GB: Sequential loading with `gc.collect()` between runs
- Models > 10GB: Sharded loading, process 4-8 queries per shard
- Peak memory usage:  $\approx 0.52 \times$  model size via aggressive unloading

## A.6 Reproducibility Checklist

To reproduce results from Table 1:

1. Install dependencies: `torch>=2.2.0, transformers>=4.36.2, numpy, scipy`
2. Download models to `/LLM_Models/`
3. Run: `python scripts/run_e2e_validation.py --ref-model gpt2 --cand-model gpt2-medium --mode audit`
4. Verify bundle hash matches reported value
5. Check `experimental_results/*/evidence_bundle.json` for full metrics

Code available at: [URL upon publication]