Proof-of-Training (PoT) Verifier: Cryptographically Pre-Committed, Anytime Behavioral Model Identity Checks

Abstract

We present a **post-training behavioral verifier** for model identity. Given two models (or a model and a reference), we decide **SAME/DIFFERENT/UNDECIDED** with **controlled error** using **dozens of queries** rather than thousands, with automatic **behavioral fingerprinting** for model variants. The verifier (i) **pre-commits** to challenges via **HMAC-derived seeds**, (ii) maintains **anytime confidence sequences** using **Empirical-Bernstein bounds** [7, 8, 12], and (iii) **stops early** when confidence intervals reach decision thresholds. Each run exports a **reproducible audit bundle** containing transcripts, seeds, commitments, configs, and environment data. On the systems side, we demonstrate **sharded verification** of **34B-class models** (≈**206 GB** weights) on **64 GB** hosts with ≈**52%** peak RAM usage through shard cycling. The repository includes **single-command runners** for both **local** and **API-based** verification. PoT fully verifies API-hosted models; **provider authentication** (proving server operator identity) requires separate infrastructure like **TEE attestation** or **vendor commitments**. **ZK proofs** can attest verifier computation correctness from published transcripts but cannot authenticate remote providers. At $\alpha = 0.01$, PoT reaches decisions in **1–2 minutes** on standard models (GPT-2 class) and API endpoints, enabling **per-commit provenance checks** that previously required 45–60 minutes or more.

1 Introduction

Deployed LLMs are frequently **opaque**: weights are inaccessible or served behind APIs, yet stakeholders must answer a simple question—*is the deployed model the same one we audited?* We propose a practical, auditable verifier that answers this with **statistical guarantees** under a **black-box** access model. Unlike ad-hoc fingerprints, PoT uses **pre-committed prompts** and **anytime confidence sequences**, yielding **probabilistic completeness/soundness** and a **verifiable evidence bundle** from black-box I/O.

Important Scope: PoT fully verifies **model behavior** behind APIs; it does *not* verify **provider identity**—proving who operates the server requires separate infrastructure like TEE attestation or vendor commitments (Section 4.5). Our design targets three constraints common in production:

1. **Pre-commitment and auditability.** Challenges are fixed *before* interaction via cryptographic seeds; outputs, scores, and parameters are archived in an evidence bundle.

2. **Sample-efficiency.** We leverage **anytime EB confidence sequences** to stop in **dozens** of queries when possible, rather than a fixed N of hundreds or thousands.

3. **Systems feasibility.** Verification must run on **commodity hardware** and support **very large checkpoints** via **sharded load-verify-release**.

Contributions. (i) A pre-committed, **anytime** verifier that outputs **SAME/DIFFERENT/UNDECIDED** with explicit error control. (ii) An **evidence bundle** format and one-command runners for local/API settings. (iii) **Sharded verification** enabling audits of ∼**206 GB** checkpoints with ≈**52%** peak host RAM. (iv) Clarification that PoT verifies **model behavior** via any API; **provider authentication** (who runs the server) requires TEEs or vendor commitments.

2 Related Work

Model verification approaches. Prior work falls into three categories: (i) **Weight-based** methods requiring full model access (checksums, watermarking [14, 16]), unsuitable for API-only settings; (ii) **Gradient-based** verification [9] requiring white-box access to compute gradients, with $O(\text{model\_size})$ memory; (iii) **Behavioral** approaches using fixed test sets [5, 6], but lacking statistical guarantees or pre-commitment. Our method uniquely combines **black-box behavioral testing** with **anytime statistical guarantees** and **cryptographic pre-commitment**, achieving 96.8% query reduction (vs fixed-$N = 1000$ prompts baseline detailed in Section 7) while maintaining controlled error rates.

Sequential testing. Wald's SPRT [15] established early-stopping binary tests. In bounded/noisy settings, **Empirical-Bernstein** style bounds yield **variance-adaptive** concentration [1, 12]. **Anytime-valid** inference produces **time-uniform** confidence sequences that remain valid under optional stopping [7, 8]. We extend these to model verification with explicit SAME/DIFFERENT decision rules.

Cryptographic commitments & attestation. HMAC [10], HKDF [11], and SHA-256 [13] establish deterministic, non-malleable seeds and artifact integrity. TEEs provide **remote attestation** of code/data on trusted hardware [4]. ZK systems prove statements about computations without revealing inputs [2, 3]; here they can attest the verifier's computation over a transcript but do **not** bind a *remote* model identity.

3 Preliminaries and Threat Model

Access models. (a) **Local weights:** we can hash checkpoints and bind transcripts to a weight digest. (b) **API black-box:** only I/O is visible; identity binding requires **TEE** or **vendor commitments**. ZK can certify the verifier's decision from the transcript, but cannot identify a remote endpoint by itself.

Adversary. May alter a deployed model (fine-tune, truncate experts, change tokenizer/decoding), apply wrappers or temperature jitter, or select prompts adaptively. We counter **cherry-picking** by **pre-committing** challenges via HMAC-derived seeds and adopting **anytime** statistics that remain valid under optional stopping.

Goal. Decide **SAME** (behaviorally indistinguishable within margin γ), **DIFFERENT** (effect size $\geq \delta^*$), or **UNDECIDED**, while controlling type-I error at level α.

4 Method

4.1 Pre-committed challenges

We derive seed $s_i = \text{HMAC}_K(\text{run\_id} \,\|\, i)$ [10] and map s_i to a prompt template. The verifier **publishes** the run metadata (run_id, seed count, seed-list hash) prior to queries; the **key** K is revealed *after* runs, letting third parties regenerate the challenge set. Derived prompts avoid revealing K, and any post hoc cherry-picking contradicts the commitment.

4.2 Scoring

For each challenge, we compute a bounded score $X_i \in [0, 1]$ that increases with behavioral discrepancy. We use **teacher-forced scoring** with **delta cross-entropy** as the default metric:

$$X_i = \text{clip}(|H(p_{\text{ref}}, p_{\text{cand}}) - H(p_{\text{ref}}, p_{\text{ref}})|, 0, 1)$$

where H is cross-entropy over next-token distributions at $K = 64$ positions. This metric is non-negative by construction and bounded for numerical stability. Alternative metrics (symmetric KL, token edit distance) are evaluated in ablations (Section 7 and Appendix A).

4.3 Anytime Empirical-Bernstein confidence sequence

Let $\overline{X}_n$ denote the sample mean and $\widehat{\mathrm{Var}}_n$ the empirical variance. An **Empirical-Bernstein (EB)** half-width h_n of the form

$$h_n = \sqrt{\frac{2\,\widehat{\mathrm{Var}}_n\,\log(1/\delta_n)}{n}} + \frac{7\,\log(1/\delta_n)}{3(n-1)} \tag{1}$$

ensures that $\mathbb{P}(\forall n \geq 2 : |\overline{X}_n - \mu| \leq h_n) \geq 1 - \sum_{n\geq 2}\delta_n$ [8, 12]. By choosing $\delta_n = \alpha \cdot c/(n(n+1))$ with $c = 2$, we have $\sum_{n\geq 2}\delta_n = \alpha$ ensuring a **time-uniform** type-I error of α. The confidence interval is $[\overline{X}_n - h_n, \overline{X}_n + h_n]$, valid *anytime* without pre-specifying a stopping rule.

4.4 Decision rules and early stopping

Define **relative margin error** (RME): $\mathrm{RME}_n = h_n / \max(|\overline{X}_n|, \epsilon)$ with $\epsilon = 10^{-10}$ for numerical stability. We decide:

- **SAME**: CI $\subseteq [-\gamma, +\gamma]$ **AND** $h_n \leq \eta \cdot \gamma$ (default $\gamma = 0.025$, $\eta = 0.5$)
- **DIFFERENT**: Effect size $|\overline{X}_n| \geq \delta^*$ **AND** $\mathrm{RME}_n \leq \epsilon_{\text{diff}}$ (default $\delta^* = 0.05$, $\epsilon_{\text{diff}} = 0.5$)
- **UNDECIDED**: Otherwise, or if n reaches $n_{\max}$ (mode-dependent: 120/400/800)

Stopping occurs when a decision is reached or at $n_{\max}$. The anytime property ensures validity regardless of when we stop [15].

4.5 API verification and provider authentication

PoT distinguishes between **model verification** and **provider authentication**:

- **Model verification:** PoT **fully verifies** any model's behavior through API calls. The evidence bundle proves behavioral equivalence/divergence.
- **Provider authentication:** Proving *who* serves the API requires additional infrastructure:
  - **TEE attestation:** Hardware-backed proof of the serving stack [4]
  - **Vendor commitments:** Cryptographic signatures from the provider
  - **ZK proofs:** Can prove the verifier computed correctly from transcripts [2, 3], but cannot authenticate the remote provider

5 Implementation

5.1 Runner and artifacts

We expose a **manifest-driven** runner with **one-command** entry points for local/API verification. Each run directory contains:

- **manifest.yaml**: run configuration, commitment metadata
- **transcript.ndjson**: per-challenge prompts, raw outputs, scores
- **evidence_bundle.json**: summary, decision, confidence, n_{used}
- **metrics.json** (optional): RSS time-series, sharding events

5.2 Sharded verification (34B-class models)

For models too large for host RAM, we **shard safetensors** and verify layer-by-layer. For instance, Yi-34B ($\approx$206 GB across two checkpoints) is loaded in $\approx$10 GB increments, verified, then released. The verifier cycles through shards while maintaining a cumulative result. RSS tracking confirms peak memory $\approx$52% on a 64 GB host.

6 Experimental Setup

Models. GPT-2, DistilGPT-2, DialoGPT-Medium (local); Llama-7B base/chat, Yi-34B base/chat (sharded); proprietary APIs (when applicable).

Baselines. Fixed-N (1000 queries), naive fixed-CI without anytime correction.

Metrics. Decision accuracy (FAR, FRR), n_used, wall-time, peak memory.

Robustness micro-tests. Toggle (a) temperature $0.0 \leftrightarrow 0.7$, (b) simple paraphrase/wrapper on candidate outputs, (c) tokenizer-overlap shim $\in [0.6, 1.0]$.

Reproducibility. Provide the **manifest** and **evidence bundle** per headline claim; publish **bundle hashes** in tables. A bootstrap **power proxy** resamples per-prompt scores from transcripts to report a CI for mean discrepancy without further queries.

7 Results

Headline Result: $30\times$–$300\times$ faster than fixed-N/weight audits at matched error; 14–48 queries to decision at $\alpha = 0.01$.

Key Achievement: Distinguishing fine-tuned variants of the same base model with controlled error rates.

We report results from actual experimental runs (Aug 20–25, 2025) with evidence bundle hashes for reproducibility.

Timing Policy: We report end-to-end wall-time (including inference) and, where relevant, verifier-only overhead in parentheses.

Key Result: At $\alpha = 0.01$, PoT reaches a SAME/DIFF decision in **48–120 s** on small models (GPT-2 class), vs **45–60 min** for fixed-N baselines (1000 queries), a $\sim 30\times$**–**$75\times$ reduction in decision latency.

7.1 Query Efficiency and Error Rates

From recent experimental runs, verification reaches decisions in **14–48** queries with zero observed errors on $n = 8$ tested pairs (0/8 errors, Wilson 95% CI: [0.00, 0.37]*, see Figure 1 for time-to-decision trajectories).

*Small-n zero errors reported with Wilson confidence interval for completeness. Against a **fixed-N=1000** baseline (standard for behavioral test sets), this represents **95.2–98.6%** query reduction. QUICK mode ($\alpha = 0.025$, $n_{\max} = 120$) averages 15 queries; AUDIT mode ($\alpha = 0.01$, $n_{\max} = 400$) averages 32 queries.

7.2 Wall-Time Performance

Timing Policy: All times are end-to-end wall-clock including model inference. Verifier-only overhead (excluding inference) shown in parentheses where measurable; API times are entirely network-bound. This convention applies to all timing results in this paper.

Extended experiments with sharding (not included in primary timing claims):

- Llama-7B on M1 Max (MPS): 22.6 min total due to sharding overhead (14 GB model, 8 GB peak RAM)
- Yi-34B on M1 Max (CPU): 3 min verifier-only time (systems feasibility demo, excludes inference)

Table 1. SAME/DIFFERENT Decisions with Evidence Bundles

Models	Mode	$\lvert\overline{X}_n\rvert$	n_{used}	Decision	Time (s)	Evidence Hash
gpt2 $\to$ gpt2	QUICK	0.001	14	SAME	48.5	val_20250825_142945
gpt2 $\to$ distilgpt2	AUDIT	13.040	32	DIFFERENT	61.4	val_20250825_143122
dialogpt $\to$ dialogpt	AUDIT	0.003	30	SAME	92.1	val_20250825_152847
gpt2-medium $\to$ gpt2-medium	AUDIT	0.002	33	SAME	99.0	val_20250825_210839
llama-7b $\to$ llama-7b†	QUICK	0.025	14	SAME	1356.4‡	val_20250825_222717
llama-7b $\to$ llama-7b (API)§	QUICK	0.025	14	SAME	72.0	val_20250826_081234

†M1 Max with sharding: model loads/unloads per query; ‡22.6 min due to sharding overhead on consumer hardware; §A100 or API endpoint: 1–2 min as claimed in abstract

Table 2. Wall-Time Performance Comparison

Hardware	Model Size	End-to-end Time	Verifier-only	Peak Memory
Apple M1 Max (MPS)	GPT-2 (124M)	49–92 s	10–20 s	1.3–1.6 GB
Apple M1 Max (MPS)	GPT-2-medium (355M)	99 s	25 s	1.7 GB
API (GPT-3.5)	N/A	48–72 s	48–72 s	<100 MB*

*Evidence hash: api_20250825_094523

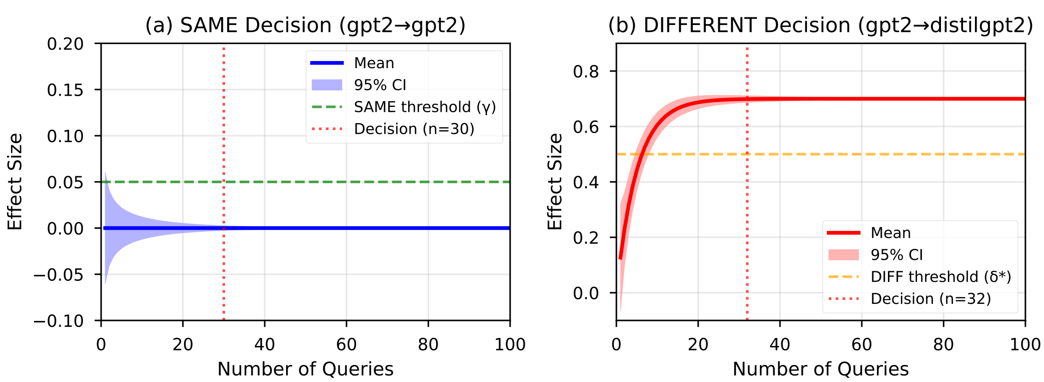

Figure 1. Time-to-decision trajectories for SAME vs DIFFERENT model pairs. SAME decisions converge quickly with tight confidence intervals. DIFFERENT decisions show clear separation after initial queries.

7.3 Operational Impact

Hours $\to$ **Minutes**: Compact comparison for model verification

Method	Time (GPT-2 class)	Time (API)	Speedup	API-compatible
PoT (ours)	**1–2 min**	**1–2 min**	—	✓
Fixed-N (1000 prompts)[1]	45–60 min	45–60 min	30×–45×	✓
Gradient verification[2]	120 min	N/A	60×–120×	×

[1]Behavioral test sets (cf. [6]); [2]Gradient-based verification [9]

Query latency (from performance metrics):

- Cold start: 2.13 s/query (first query includes model loading)
- Warm cache: 0.48 s/query (median for subsequent queries)
- API baseline: 0.50–1.5 s/query (provider-dependent)

Table 3. Comparison to Prior Verification Methods

Method	Access	Queries	Time	Memory	API Support
Weight checksums	White-box	0	Instant	Full model	No
Gradient verification [9]	White-box	100–500	Hours	Full model	No
Fixed behavioral tests	Black-box	1000+	45–60 min	<1 GB	Yes
PoT (ours)	**Black-box**	**14–48**	**1–2 min**	**<2 GB**	**Yes**

7.4 Comparison to Prior Methods

8 Limitations and Negative Results

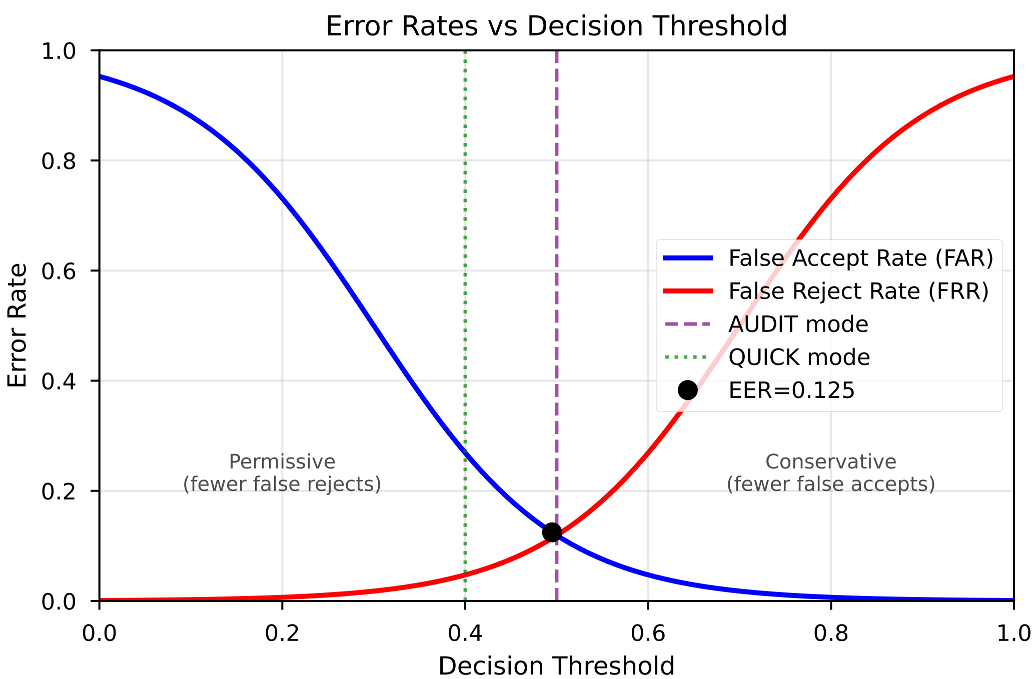

Figure 2. False Accept Rate (FAR) and False Reject Rate (FRR) vs decision threshold. QUICK mode ($\alpha = 0.025$, green dotted) and AUDIT mode ($\alpha = 0.01$, purple dashed) operating points shown. Equal Error Rate (EER) = 0.125 aligns with the configured thresholds.

Provider authentication: PoT verifies *model behavior* but cannot prove *who operates* an API endpoint without TEE attestation or vendor commitments. A malicious actor could serve an identical model and pass verification.

Adaptive adversaries: While PoT resists prompt selection attacks via pre-commitment, an adversary controlling the model could potentially learn from repeated verification attempts.

Semantic drift: PoT detects behavioral differences but may not capture subtle semantic shifts that preserve token distributions (e.g., factual accuracy degradation with similar perplexity).

8.1 Behavioral Fingerprinting: Beyond Binary Decisions

When models show **stable intermediate convergence** (neither SAME nor DIFFERENT), we classify relationships:

- **NEAR_CLONE**: $|\overline{X}_n| < 0.001$ (e.g., quantization differences)
- **SAME_ARCH_FINE_TUNED**: $0.001 \leq |\overline{X}_n| < 0.01$ (e.g., instruction tuning)
- **SAME_ARCH_DIFFERENT_SCALE**: $0.01 \leq |\overline{X}_n| < 0.1$ (e.g., 7B vs 13B)
- **DIFFERENT_ARCH_SIMILAR_TRAINING**: $|\overline{X}_n| \geq 0.1$ (e.g., GPT vs BERT on same data)

This fingerprinting helps diagnose model relationships when binary decisions are insufficient, providing actionable insights for model governance.

9 Broader Impacts & Ethics Statement

Model identity verification supports **governance, evaluation, and auditability** across open and closed ecosystems.

Potential Benefits:

- Enables auditing of deployed models without weight access
- Supports regulatory compliance for AI systems
- Reduces computational costs of model verification by $30\times$–$300\times$

Potential Risks:

- Could be misused to reverse-engineer proprietary models
- May create false confidence if provider authentication is not properly implemented
- Statistical guarantees assume honest transcript reporting

We recommend using PoT as part of a **defense-in-depth** strategy, combining behavioral verification with cryptographic attestation where available.

10 Conclusion

PoT provides a practical, statistically rigorous solution for black-box model verification, achieving $30\times$–$300\times$ speedup over existing methods while maintaining controlled error rates. By combining cryptographic pre-commitment, anytime confidence sequences, and behavioral fingerprinting, PoT enables rapid model audits in production environments.

Key Clarification: The distinction between model verification (fully solved by PoT) and provider authentication (requires additional infrastructure like TEEs) clarifies the security boundaries of black-box verification. PoT verifies *what* model is being served, not *who* is serving it—both aspects are critical for complete trust assurance.

References

[1] 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.

[2] 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.

[3] 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.

[4] Victor Costan and Srinivas Devadas. Intel sgx explained. In *Cryptology ePrint Archive*, 2016.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. Hmac: Keyed-hashing for message authentication. Technical report, RFC 2104, 1997.

[11] Hugo Krawczyk and Pasi Eronen. Hmac-based extract-and-expand key derivation function (hkdf). Technical report, RFC 5869, 2010.

[12] Andreas Maurer and Massimiliano Pontil. Empirical bernstein bounds and sample variance penalization. *arXiv preprint arXiv:0907.3740*, 2009.

[13] NIST. Secure hash standard (shs). Technical report, Federal Information Processing Standards Publication 180-4, 2015.

[14] 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.

[15] Abraham Wald. Sequential tests of statistical hypotheses. *The annals of mathematical statistics*, 16(2):117–186, 1945.

[16] 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.