SVIP: Towards Verifiable Inference of Open-source Large Language Models

Yifan Sun[1*] **Yuhang Li**[1*] **Yue Zhang**[2] **Yuchen Jin**[2] **Huan Zhang**[1]
[1]University of Illinois Urbana-Champaign [2]Hyperbolic Labs
{yifan50,yuhang8}@illinois.edu {Jasper,yuchen}@hyperbolic.xyz
huan@huan-zhang.com

Abstract

The ever-increasing size of open-source Large Language Models (LLMs) renders local deployment impractical for individual users. *Decentralized computing* has emerged as a cost-effective solution, allowing individuals and small companies to perform LLM inference for users using surplus computational power. However, a computing provider may stealthily substitute the requested LLM with a smaller, less capable model without consent from users, thereby benefiting from cost savings. We introduce SVIP, a secret-based verifiable LLM inference protocol. Unlike existing solutions based on cryptographic or game-theoretic techniques, our method is computationally effective and does not rest on strong assumptions. Our protocol requires the computing provider to return both the generated text and processed hidden representations from LLMs. We then train a proxy task on these representations, effectively transforming them into a unique model identifier. With our protocol, users can reliably verify whether the computing provider is acting honestly. A carefully integrated secret mechanism further strengthens its security. We thoroughly analyze our protocol under multiple strong and adaptive adversarial scenarios. Our extensive experiments demonstrate that SVIP is accurate, generalizable, computationally efficient, and resistant to various attacks. Notably, SVIP achieves false negative rates below 5% and false positive rates below 3%, while requiring less than 0.01 seconds per prompt query for verification.

1 Introduction

In recent years, open-source Large Language Models (LLMs) have achieved unprecedented success across a broad array of tasks and domains [44, 2, 26, 19], while remaining freely accessible. However, as model sizes increase, so do their computational demands [23]. As a result, **decentralized computing** [45] has gained significant attention as a cost-effective solution for users with limited local computational resources. In this setting, a user lacking computational power relies on *decentralized* computing providers to perform LLM inference. These providers, often individuals or small companies with surplus resources, offer computational power at competitive prices. Commercial platforms facilitate such interactions by connecting both parties. Real-world examples include Golem Network, Akash Network, Render Network, Spheron Network, Hyperbolic, and Vast.ai.

However, unlike reputable companies with well-established credibility, computation outputs from decentralized computing providers may not always be trustworthy. Specifically, to ease the deployment of LLM inference, computing providers often provide API-only access to users, hiding implementation details. A new risk arises in this setting: *how to ensure that the outputs from a computing provider are indeed generated by the requested LLM*? For instance, a user might request

*Equal contribution.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Socially Responsible and Trustworthy Foundation Models.

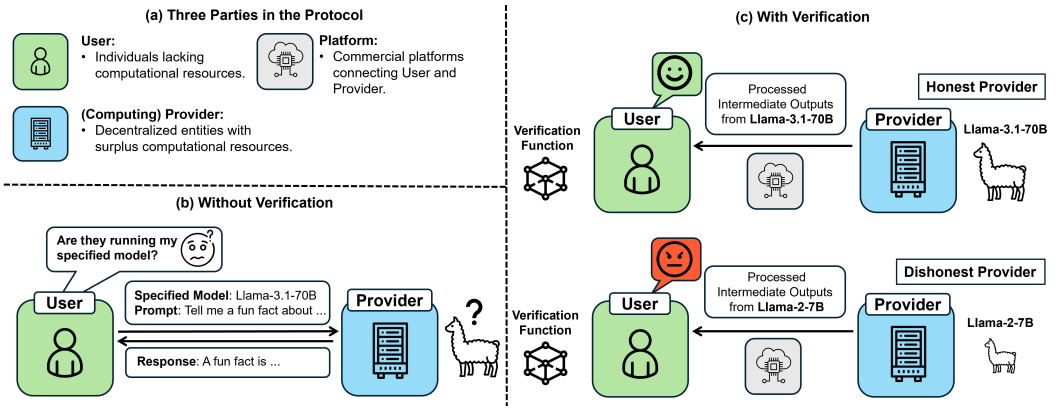


Figure 1: The problem setting of verifiable inference for LLMs. (a) Our protocol involves three parties. (b) A user requests the computing provider (referred to as *provider* in the figure) to run inference on their prompt using the `Llama-3.1-70B` model. Without verification, they have no way to confirm if the specified model is used. (c) Our proposed protocol solves this by requiring the provider to return processed hidden representations from the LLM, enabling the user to verify through a verification function whether the correct model was used for inference. Specifically, the hidden representations are compressed to reduce the computational overhead.

the `Llama-3.1-70B` model for complex tasks, but a dishonest computing provider could substitute the smaller `Llama-2-7B` model for cost savings, while still charging for the larger model. The smaller model demands significantly less memory and processing power, giving the computing provider a strong incentive to cheat. Restricted by the black-box API access, it is difficult for the user to detect such substitutions.

Without assurance that users are receiving the service they specified and paid for, they may lose trust and abandon the platform. To prevent this outcome and maintain profitability, the platform must ensure that user-specified models are faithfully executed. This highlights the need for **verifiable inference**, a mechanism designed to ensure that the model specified by the user is the one actually used during inference.

Related Work A practical verifiable inference solution for LLMs must accurately confirm that the specified model is being used during inference while maintaining computational efficiency. Existing approaches face significant challenges that limit their applicability. Cryptographic verifiable computing methods, which rely on generating mathematical proofs [50, 38] or secure computation techniques [13, 25] often impose high computational costs, making them unsuitable for real-time LLM inference. For instance, zkLLM, a recent Zero Knowledge Proof-based technique, requires over 803 seconds for a single prompt query [40]. Game-theoretic protocols involve the interaction of multiple computing providers with carefully designed penalties and rewards [53], assuming all providers are rational, flawless, and non-cooperative, which might be unrealistic for certain system setups in practice. Meanwhile, watermarking and fingerprinting techniques [21, 48] are mostly implemented by model publishers, making them unsuitable for verifiable inference, where the verification primarily occurs between the user and the computing provider. We leave an extended discussion of related work to Appendix D.

In this paper, we propose `SVIP`, a **S**ecret-based **V**erifiable LLM **I**nference **P**rotocol using hidden representations. Our protocol requires the computing provider to return not only the generated text but also the processed **hidden state representations** from the LLM. We carefully design and train a proxy task exclusively on the hidden representations produced by the specified model, effectively transforming these representations into a distinct identifier for that model. During deployment, users can verify whether the processed hidden states returned by the computing provider come from the specified model by assessing their performance on the proxy task. If the returned representations perform well on this task, it provides strong evidence that the correct model was used for inference.

Our key contributions are:

- We systematically formalize the problem of verifiable LLM inference (§2) and propose an innovative protocol that leverages processed hidden representations (§3.1).
- The security of our protocol is further enhanced by a novel secret-based mechanism (§3.2). We provide a thorough discussion and analysis of various strong and adaptive attack scenarios (§3.3).

Table 1: Comparison of simple approaches and our proposed protocol based on the five criteria. A checkmark (✓) indicates that the criterion is satisfied, while a cross (✗) indicates it is not. SVIP is the only method that satisfies all five criteria.

Approach	Low FNR	Low FPR	Efficiency	Completion Quality	Robustness
Benchmark Prompt Testing	✓	✓	✗	✓	✗
Binary Classifier on Hidden States	✓	✓	✓	✓	✗
SVIP **(Ours)**	✓	✓	✓	✓	✓

• Our comprehensive experiments with 5 specified open-source LLMs (from 13B to 70B) demonstrate the effectiveness of SVIP: it achieves an average false negative rate of 3.49%, while keeping the false positive rate below 3% across 6 smaller alternative models (§4.1). SVIP introduces negligible overhead (less than 0.01 seconds per prompt query) for both users and computing providers (§4.2). Furthermore, SVIP can effectively and securely handle 80 to 120 million prompt queries in total after a single round of protocol training, with the update mechanism further bolstering security (§4.3).

2 Problem Statement

The verifiable inference problem involves three parties, as illustrated in Figure 1:

1. **User:** An individual who lacks sufficient computing resources and seeks to perform expensive LLM inference tasks on given prompts using decentralized computing providers at a low cost.

2. **Computing Provider:** Decentralized entities, often *small companies or individuals*, that rent out computational power at competitive prices.

3. **Platform:** A commercial platform that profits by connecting users and computing providers. Importantly, the platform *itself does not require significant computational resources*, as its primary role is to facilitate and monitor the utilization of computational resources from decentralized providers.

Threat Model To reduce costs, a computing provider may not actually use the LLM the user specifies. Instead, it may substitute a significantly smaller model, which returns an inferior result. It may also attempt to evade detection by actively concealing dishonest behavior.

The Incentive and Goal of Verifiable Inference To address this threat, platforms are **commercially incentivized** to maintain user trust by monitoring provider behavior, ensuring that providers cannot cheat. Trust in the platform underpins its **reputation and business model**; if users cannot trust that model inference is faithfully executed, they are likely to abandon the platform, leading to **significant financial and operational losses**.

To mitigate this risk, the platform designs and implements a verification protocol that allows users to verify, with high confidence, whether the computing provider used the specified model for inference. Note that during deployment, the protocol should operate *primarily* between the user and the provider, with minimal platform involvement. A satisfactory protocol should meet the following criteria: **(1) Low False Negative Rate (FNR)**: The protocol should minimize cases where the computing provider *did* use the specified LLM for inference but is incorrectly flagged as not using it. **(2) Low False Positive Rate (FPR)**: The protocol should rarely confirm that the computing provider used the specified LLM if it actually used another model. **(3) Efficiency:** The verification protocol should be computationally efficient and introduce minimal overhead for both the computing provider and the user. **(4) Preservation of Completion Quality:** The protocol should not compromise the quality of the prompt completion returned by the computing provider. **(5) Robustness:** The protocol should maintain low FNR and FPR even against adversarial providers attempting to evade detection.

2.1 Simple Approaches Do Not Meet All the Criteria

Table 1 evaluates several straightforward approaches for verifiable LLM inference, as well as our proposed protocol (§3), against the five criteria. All naive approaches fail to satisfy at least one criterion, underscoring the necessity of our method. We exclude solutions that involve multiple computing providers (*e.g.*, cross-verifying results across providers) because such approaches significantly increase user costs, making them impractical for widespread adoption.

Benchmark Prompt Testing The user curates a small set of prompt examples from established benchmarks and sends them to the computing provider. If the provider's performance significantly deviates from the reported benchmark metrics for the specified model, the user may suspect dishonest

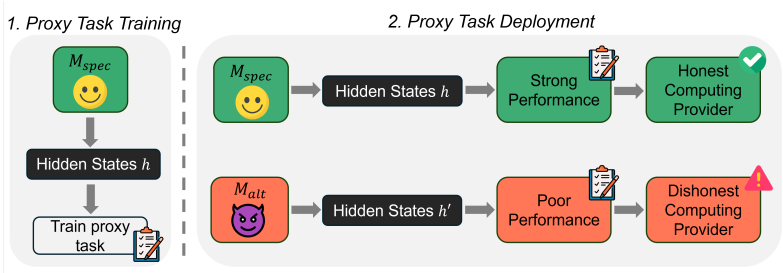


Figure 2: Illustration of the motivation behind our framework. The proxy task is trained solely on hidden states from the specified LLM M_{spec}. During deployment, strong performance on the proxy task indicates that the provider used M_{spec} as specified, while poor performance suggests otherwise.

behavior. However, a malicious provider can easily bypass this method by detecting known benchmark prompts and selectively applying the correct model only for those cases, while using an alternative model for all other prompt queries. Additionally, testing such benchmark prompts also increases the user's inference costs.

Binary Classifier on Hidden States The user can request the computing provider to return hidden representations from the LLM used for inference, and train a binary classifier on these representations to verify if they come from the specified model. However, a simple attack involves the provider caching hidden representations from the correct model that are unrelated to the user's input. The dishonest provider could then use a smaller LLM for inference and return these cached irrelevant representations to deceive the classifier while saving costs.

3 Methodology

Motivation It is often challenging to verify whether a computing provider is using an alternative LLM for inference based solely on the returned completion text[2]. Our framework addresses this by requiring the computing provider to return not only the generated text but also the processed **hidden state representations** from the LLM inference process.

We design and train a **proxy task** specifically to perform well *only* on the hidden representations generated by the specified model during the protocol's training stage. The intuition behind is that the proxy task transforms the hidden representations into a unique identifier for the model. During deployment, the user can evaluate the performance of the returned hidden states on the proxy task. Strong performance on the proxy task indicates that the correct model was used for inference, while poor performance suggests otherwise. Figure 2 provides an illustration.

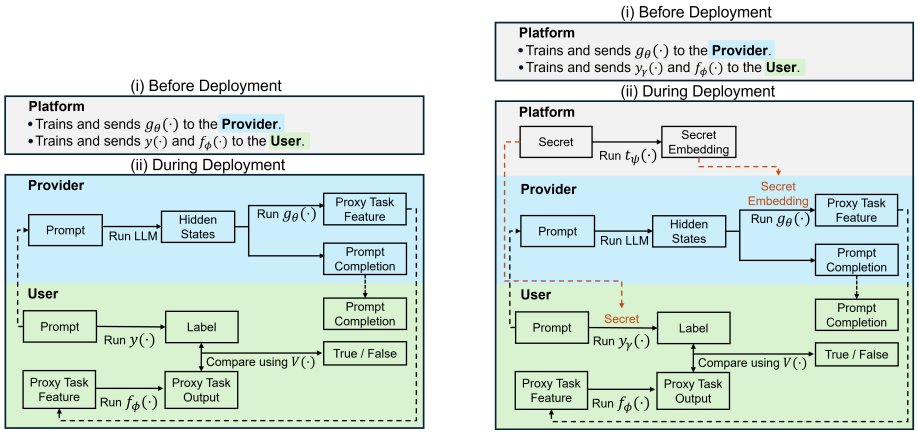
Our approach does not depend on expensive cryptographic proofs or protocols, and is highly efficient. Furthermore, it does not involve retraining or fine-tuning the LLMs, operates independently of the model publisher, and can be applied to any LLM with publicly available weight parameters, making it widely applicable.

Notations Let $x \in \mathcal{V}^*$ denote the prompt query from user, where $\mathcal{V}^*$ represents the set of all possible string sequences for a vocabulary set $\mathcal{V}$. The specified LLM and alternative LLM are denoted as $\mathcal{M}_{spec}$ and $\mathcal{M}_{alt}$, respectively.

3.1 A Simple Protocol Based on Hidden States

Protocol Overview For any LLM $\mathcal{M}$, let $h_{\mathcal{M}}(x) \in \mathbb{R}^{L \times d_{\mathcal{M}}}$ represent the **last-layer** hidden representations of x produced by $\mathcal{M}$, where L is the length of the tokenized input x, and $d_{\mathcal{M}}$ denotes the hidden dimension of $\mathcal{M}$. The computing provider receives x from the user, runs $\mathcal{M}$, and returns $h_{\mathcal{M}}(x)$ to user for subsequent verification. However, to reduce the size of the hidden states returned, we additionally apply a proxy task feature extractor network $g_\theta(\cdot) : \mathbb{R}^{L \times d_{\mathcal{M}}} \to \mathbb{R}^{d_g}$ parameterized by θ, where d_g represents the proxy task feature dimension. The computing provider now also runs $g_\theta(\cdot)$ and returns a compressed vector $z(x) := g_\theta(h_{\mathcal{M}}(x))$ of dimension d_g to the user, significantly

[2]To empirically demonstrate this, we train a binary classifier to distinguish between output texts from a specified model (LlaMA-2–13B) and six smaller alternatives. Using 90,000 prompts for training and 10,000 for testing, the classifier (BERT-base-uncased) achieves an FNR of 36.1% and an FPR of 58.9%.



(a) Simple Protocol (b) Secret-based Protocol

Figure 3: Illustration of (a) the simple protocol (Section 3.1); (b) secret-based protocol (Section 3.2).

reducing the communication overhead. Specifically, for each prompt query, the compressed vector only takes approximately 4 KB when d_g is set to 1024.

The user is required to perform two tasks locally: obtaining the predicted proxy task output and the label. First, the user runs $f_\phi(\cdot)$, using the returned proxy task feature $z(x)$ as input to compute $f_\phi(z(x))$. Here, $f_\phi(\cdot) : \mathbb{R}^{d_g} \to \mathcal{Y}$ is the proxy task head parameterized by ϕ, where $\mathcal{Y}$ denotes the label space. Second, the user applies a labeling function for the proxy task. We adopt a self-labeling function $y(x) : \mathcal{V}^* \to \mathcal{Y}$, which derives the label directly from the input, eliminating the need for external labels or specialized annotators[3]. The label can either be a scalar or a vector.

Finally, the user checks whether $f_\phi(z(x))$ matches $y(x)$. Our training process below ensures that, with high probability, $f_\phi(z(x)) = y(x)$ when $\mathcal{M}_{spec}$ is used for inference, and that this does not hold for other models, as the proxy task is exclusively trained on the hidden representation distribution induced by $\mathcal{M}_{spec}$. This completes our protocol. Refer to Figure 3a for a detailed illustration.

Proxy Task Training With a properly defined loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ and a training dataset $\mathcal{D}$, the platform trains the proxy task according to the following objective:

$$\phi^*, \theta^* = \arg\min_{\phi,\theta} \mathbb{E}_{x\sim\mathcal{D}} \left[\ell \left(f_\phi(g_\theta(h_{\mathcal{M}_{spec}}(x))), y(x) \right) \right].$$

Protocol Deployment With the optimized ϕ^* and θ^*, we define the **verification function** as $V(x, z(x); \phi^*, \theta^*) = \mathbb{1}\left(f_{\phi^*}(z(x)) = y(x)\right)$, where $z(x) = g_{\theta^*}(h_{\mathcal{M}}(x))$ is returned by the computing provider. If the value of the verification function is 1 (or 0), we conclude that the computing provider is indeed (or is not) using $\mathcal{M}_{spec}$ for inference with high probability. Now, the low FNR and low FPR criteria introduced in Section 2 can be formally expressed as follows:

$$\begin{aligned} &\textbf{Low FNR} : \mathbb{P}\left(V(x, z(x); \phi^*, \theta^*) = 0 \mid \mathcal{M}_{spec} \text{ used}\right) \leq \alpha; \\ &\textbf{Low FPR} : \mathbb{P}\left(V(x, z(x); \phi^*, \theta^*) = 1 \mid \mathcal{M}_{spec} \textit{ not} \text{ used}\right) \leq \beta. \end{aligned} \tag{1}$$

Here α and β are predefined thresholds. While a single prompt query may occasionally yield an incorrect verification result due to FNR or FPR, in practice, users can perform the verification over multiple distinct queries and apply a hypothesis testing to reach a final single conclusion with high confidence. Refer to Section 4.1 and Appendix C.2 for a detailed discussion.

3.2 `SVIP`: A Secret-based Protocol for Verifiable LLM Inference

From Simple Protocol to Secret-based Protocol The simple protocol, despite its strong potential in discriminating whether the specified model is actually used, is vulnerable to malicious attacks from the computing provider. A dishonest provider may attempt to bypass the verification process without running $\mathcal{M}_{spec}$. Since all the provider needs to return is a vector of dimension d_g, an attacker

[3]For instance, we can define $y(x)$ as the Set-of-Words (SoW) representation of the input x, which captures the presence of each word in a fixed vocabulary, regardless of frequency. As a concrete example, if $\mathcal{V} = \{a, b, c, d\}$ and $x =$ "$abcc$", the SoW label $y(x)$ would be a four-dimensional vector $(1, 1, 1, 0)$, indicating whether each token in $\mathcal{V}$ appears in x.

could adversarially optimize a vector $\tilde{z} \in \mathbb{R}^{d_g}$ directly, without actually running $g_{\theta^*}(\cdot)$ and using *any* LLM. We refer to this as a **direct vector optimization attack**. Specifically, if the self-labeling function is public, the adversary can run the labeling function $y(x)$ themselves for each input x and then directly find $\tilde{z}$ so that

$$\tilde{z}^* = \arg\min_{\tilde{z}} \ell\left(f_{\phi^*}(\tilde{z}), y(x)\right). \tag{2}$$

Ultimately, $\tilde{z}^*$ is returned to the user to deceive the verification protocol. As shown in Appendix F.8, this attack achieved an attack success rate (ASR) of 99.90%, indicating that the protocol's security requires further enforcement.

To strengthen the protocol's security, we introduce a "secret" mechanism. A complete illustration is provided in Figure 3b. Particularly, the platform assigns a "secret" $s \in \mathcal{S}$ exclusively to the user, which is never shared with the computing provider. Here, $\mathcal{S}$ represents the secret space. For example, $\mathcal{S}$ can be defined as the space of d_s-dimensional binary vectors, represented as $\{0,1\}^{d_s}$.

Introducing Secret into the Self-labeling Function The self-labeling function with secret is now defined as $y(x, s) : \mathcal{V}^* \times \mathcal{S} \to \mathcal{Y}$. The property below is essential for an ideal self-labeling function.

Property 1 (Secret Distinguishability)**.** *For the same input x, given two different secrets $s' \neq s$, with a pre-defined lower-bound probability δ, the resulting labels should be different with high probability:*

$$\mathbb{P}(y(x, s) \neq y(x, s')) \geq \delta. \tag{3}$$

If $\mathcal{Y}$ is a continuous space, with a pre-defined threshold, this property is equivalent to:

$$\mathbb{P}(\|y(x, s) - y(x, s')\|_2 \geq \textit{threshold}) \geq \delta. \tag{4}$$

Property 1 ensures that a malicious computing provider, without access to the specific s, cannot determine or naively guess the true label, thus rendering the direct vector optimization attack ineffective. Meanwhile, the user, with knowledge of s, can still compute the correct label.

A simple rule-based self-labeling function (*e.g.*, the SoW representation) cannot ensure that Property 1 holds. To enforce this property, we introduce a trainable labeling network $y_\gamma(x, s) : \mathcal{V}^* \times \mathcal{S} \to \mathcal{R}^{d_y}$ parameterized by γ, which takes $x \in \mathcal{V}^*$ and $s \in \mathcal{S}$ as input and outputs a continuous label vector of dimension d_y. This network is trained with the following contrastive loss:

$$\gamma^* = \arg\min_{\gamma} -\mathbb{E}_{x\sim\mathcal{D}, s, s'\sim\mathcal{S}}\left[\|y_\gamma(x, s) - y_\gamma(x, s')\|_2\right]. \tag{5}$$

Introducing Secret into the Proxy Task Once the labeling network is optimized, we also need to include the secret s into the proxy task. Our design is to embed s as a task token using a secret embedding network (*e.g.*, an MLP), denoted as $t_\psi(s) : \mathcal{S} \to \mathbb{R}^{d_\mathcal{M}}$, parameterized by ψ. Note that this secret embedding network $t_\psi(s)$ is only kept to the platform. Then, the platform distributes $t_\psi(s)$ to the computing provider, who concatenates $t_\psi(s)$ with $h_\mathcal{M}(x)$, runs $g_\theta(\cdot)$, and returns $z(x) = g_\theta(t_\psi(s) \oplus h_\mathcal{M}(x))$, where $\oplus$ denotes concatenation.

The training objective is now modified by incorporating randomly sampled secrets during training:

$$\phi^*, \theta^*, \psi^* = \arg\min_{\phi,\theta,\psi} \mathbb{E}_{x\sim\mathcal{D}, s\sim\mathcal{S}}\left[\ell\left(f_\phi(g_\theta(\underline{\underline{t_\psi(s)}} \oplus h_{\mathcal{M}_{spec}}(x))), y_{\gamma^*}(x, \underline{\underline{s}})\right)\right]. \tag{6}$$

As before, the user receives $z(x)$ from the computing provider. However, now that $\mathcal{Y}$ is a continuous space, a threshold η is required to determine whether the predicted proxy task output $f_{\phi^*}(z(x))$ matches the label vector $y_{\gamma^*}(x, s)$. Specifically, $f_{\phi^*}(z(x))$ is considered a match to $y_{\gamma^*}(x, s)$ if the L_2 distance between them is below the pre-defined threshold η, indicating $\mathcal{M}_{spec}$ was actually used:

$$V(x, z(x); \phi^*, \theta^*, \psi^*) = \mathbb{1}\left(\|f_{\phi^*}(z(x)) - y_{\gamma^*}(x, s)\|_2 \leq \eta\right). \tag{7}$$

In practice, we propose setting the threshold based on the **conditional empirical distribution** of $d(x, s) := \|f_{\phi^*}(z(x)) - y_{\gamma^*}(x, s)\|_2$, given that $\mathcal{M}_{spec}$ is used for inference. We select the upper 95th percentile to ensure a FNR of 5%.

3.3 Security Analysis

As previously discussed, the direct vector optimization attack described in Eq. (2) is no longer feasible due to the introduction of the secret mechanism. In this section, we discuss other potential attacks as a security analysis towards our protocol. Additional possible attacks are discussed in Appendix E.

Adapter Attack Under Single Secret A malicious attacker could attempt an adapter attack if they collect enough prompt samples $\mathcal{D}' = \{x_i\}_{i=1}^M$ under a *single* secret s. The returned vector from an honest computing provider should be $z(x) = g_{\theta^*}(t_{\psi^*}(s) \oplus h_{\mathcal{M}_{spec}}(x))$. The attacker's goal is to train an adapter that mimics the returned vector, but by using an alternative LLM, $\mathcal{M}_{alt}$.

To this end, we define the adapter $a_\lambda(\cdot) : \mathbb{R}^{d_{\mathcal{M}_{alt}}} \to \mathbb{R}^{d_{\mathcal{M}_{spec}}}$, parameterized by λ, which transforms the hidden states of $\mathcal{M}_{alt}$ to approximate those of $\mathcal{M}_{spec}$. The returned vector is then $g_{\theta^*}(t_{\psi^*}(s) \oplus a_\lambda(h_{\mathcal{M}_{alt}}(x)))$. The attacker's objective is to minimize the L_2 distance between the returned vector generated by $\mathcal{M}_{spec}$ and the vector produced by $\mathcal{M}_{alt}$ with the adapter:

$$\lambda^* = \arg\min_{\lambda} \mathbb{E}_{x\sim\mathcal{D}'} \|g_{\theta^*}(t_{\psi^*}(s) \oplus h_{\mathcal{M}_{spec}}(x)) - g_{\theta^*}(t_{\psi^*}(s) \oplus a_\lambda(h_{\mathcal{M}_{alt}}(x)))\|_2. \tag{8}$$

By minimizing this objective, the attacker seeks to make the output of $\mathcal{M}_{alt}$ with the adapter indistinguishable from that of $\mathcal{M}_{spec}$, effectively bypassing the protocol. Once the adapter is well-trained, as long as the secret s remains unchanged, the attacker can rely solely on $\mathcal{M}_{alt}$ in future verification queries without being detected.

Secret Recovery Attack Under Multiple Secrets The secret mechanism is enforced by distributing the secret s to the user, while only providing the secret embedding $t_{\psi^*}(s)$ to the computing provider. However, a sophisticated computing provider may attempt to recover the original secret by posing as a user and collecting multiple secrets and corresponding embeddings. A straightforward approach would involve recovering s from $t_{\psi^*}(s)$, thereby undermining the secret mechanism.

Suppose the attacker has curated a dataset of secret-embedding pairs, $D_{\text{secret}} = \{s_j, t_{\psi^*}(s_j)\}_{j=1}^N$. The attacker could then train an inverse model $i_\rho : \mathbb{R}^{d_\mathcal{M}} \to \mathcal{S}$, parameterized by ρ, to map the secret embedding back to the secret space. If $\mathcal{S}$ is continuous, the training objective can be formalized as:

$$\rho^* = \arg\min_{\rho} \mathbb{E}_{s\sim\mathcal{D}_{\text{secret}}} \|i_\rho(t_{\psi^*}(s)) - s\|_2. \tag{9}$$

Once the inverse model is optimized, the true label $y(x, s)$ again becomes accessible to the malicious provider. Consequently, the secret-based protocol effectively collapses to the simple protocol without secret protection, leaving it vulnerable to the direct vector optimization attack.

Defense: The Update Mechanism To defend against the attacks discussed above, we propose an update mechanism for our secret-based protocol: **(1)** In defense of the adapter attack, once the prompt queries for a given secret reach a pre-defined threshold M^*, the next secret is activated. Meanwhile, we enforce a limit on how often the next secret can be activated, preventing attackers from acquiring too many secrets within a short period. **(2)** When a total of N^* secrets have been used, the entire protocol should be retrained by the platform[4]. In practice, the values of M^* and N^* can be determined empirically, as discussed in Section 4.3.

4 Experiments

Experiment Setup To simulate realistic LLM usage scenarios, we primarily use the `LMSYS-Chat-1M` conversational dataset [55], which consists of one million real-world conversations. Results on additional datasets are provided in Appendix F.3. For the models, we select 5 widely-used LLMs as the specified models, ranging in size from 13B to 70B parameters and spanning multiple model families. As alternative models, we use 6 smaller LLMs, each with parameters up to 7B. Refer to Appendix F.1 for details. The labeling network $y_\gamma(\cdot)$ uses a pretrained sentence transformer [37] to embed the text input x and an MLP to embed the secret s, where $s \in \{0, 1\}^{d_s}$ and d_s is set to 48. The outputs of both embeddings are concatenated and passed through another MLP to produce a continuous label vector of 128 dimensions. The proxy task feature extractor $g_\theta(\cdot)$ is a 4-layer transformer, while both the proxy task head $f_\phi(\cdot)$ and task embedding network $t_\psi(\cdot)$ are implemented as MLPs. Full details can be found in Appendix F.2.

4.1 Results of Protocol Accuracy

We evaluate the accuracy of our protocol by examining the empirical estimate of FNR and FPR, as outlined in Eq. (13). To apply the verification function in Eq. (7), we first determine the threshold η

[4]Specifically, this retraining can be performed using a different random seed and training recipe. As shown in Section 4.2, the retraining process is efficient.

Table 2: FNR and FPR on the test dataset of `LMSYS-Chat-1M`. Our protocol keeps FNR under 5% and FPR under 3% across all scenarios. We implement a `Random` baseline where the computing provider generates random hidden representations directly without using any LLM.

Specified Model	FNR ↓	FPR ↓						
		`Random`	`GPT2-XL`	`GPT-NEO-2.7B`	`GPT-J-6B`	`OPT-6.7B`	`Vicuna-7B`	`Llama-2-7B`
`Llama-2-13B`	4.41%	1.97%	1.90%	1.77%	1.75%	2.03%	2.44%	2.04%
`GPT-NeoX-20B`	3.47%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
`OPT-30B`	3.42%	0.05%	0.33%	0.61%	0.47%	0.83%	0.34%	0.35%
`Falcon-40B`	3.02%	0.00%	0.00%	0.01%	0.00%	0.00%	0.00%	0.00%
`Llama-3.1-70B`	3.13%	0.26%	1.97%	1.04%	1.98%	2.07%	0.90%	0.81%

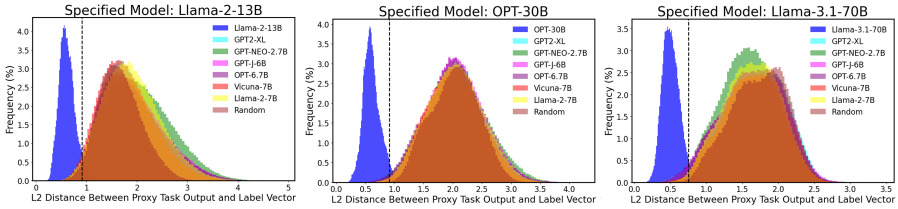


Figure 4: Empirical distribution of the L_2 distance between the predicted proxy task output $f_{\phi^*}(z(x))$ and the label vector $y_{\gamma^*}(x, s)$ on the test dataset of `LMSYS-Chat-1M`. Each figure corresponds to a different specified model. The distributions compare the L_2 distances when the specified model is used versus various alternative models. The clear separation between the distributions, marked by the vertical threshold line, ensures the high accuracy of our protocol in distinguishing between correct and incorrect model usage. More examples can be found in Figure 8.

on a validation dataset during proxy task training. We then evaluate the empirical FNR and FPR on a *held-out* test dataset with $10,000$ samples. For each test prompt, we pair it with 30 randomly sampled secrets to ensure a reliable evaluation result. For FPR calculations, we simulate scenarios where the computing provider uses an alternative, smaller LLM to produce the hidden representations, and applies $g_{\theta^*}(\cdot)$ on those outputs.

As shown in Table 2, `SVIP` consistently achieves low FNR and FPR **for individual queries** across all specified LLMs. The FNR remains below 5% per query, indicating that our protocol rarely falsely accuses an honest computing provider. Moreover, when faced with a dishonest provider, the FPR stays under 3% per query regardless of the alternative model employed, highlighting the protocol's strong performance in detecting fraudulent behavior. We further demonstrate the generalizability of our protocol to unseen datasets in Appendix F.3.

Figure 4 shows the empirical *test* distribution of $d(x, s)$, the L_2 distance between the predicted proxy task output and the label vector, under different model usage scenarios. The clear separation in the distributions provides strong evidence for the high accuracy of `SVIP`: when the specified model is actually used, $d(x, s)$ is significantly smaller compared to when an alternative model is used.

A Hypothesis Testing Framework for a Single Final Conclusion In practical scenarios, conclusions about a computing provider's honesty are based on **multiple different** prompt queries rather than a single one. A hypothesis testing framework can be adopted to combine the results of each individual query and reach a *single final* conclusion. With FPRs and FNRs below 5% for each individual query, the user can draw a final conclusion about the provider's honesty with high confidence. For instance, by employing only 30 different queries, the type-I and type-II error rates of the final conclusion are effectively driven to near zero, demonstrating the strong robustness of `SVIP`.

As an illustrative case, when using `Llama-3.1-70B` as the specified model and `Llama-2-7B` as the alternative, we achieve an FPR of 0.81% and an FNR of 3.13%. With a properly chosen decision threshold, the type-I error rate (incorrectly flagging an honest provider as dishonest) and type-II error rate (failing to detect a dishonest provider) rates are 1.7×10^{-49} and 0.0, respectively. Refer to Appendix C.2 for detailed analysis and results.

4.2 Computational Cost Analysis of the Protocol

Table 12a details the runtime per prompt query and GPU memory consumption during the deployment stage. Across all specified models, the verification process takes under 0.01 seconds per prompt query for both the computing provider and the user. For example, verifying the `Llama-2-13B` model

Table 3: Attack Success Rate for the secret recovery attack, presented as a function of the number of secret-embedding pairs collected. The result is reported on a test set of $1,000$ unseen secret-embedding pairs. The ASR remains below 50% even after collecting $200,000$ pairs.

Specified Model	1,000	5,000	10,000	50,000	100,000	200,000	500,000	1,000,000
`Llama-2-13B`	0.0%	0.0%	0.0%	2.7%	5.8%	30.1%	65.1%	69.5%
`GPT-NeoX-20B`	0.0%	0.0%	0.0%	0.0%	1.2%	19.6%	30.4%	59.9%
`OPT-30B`	0.0%	0.0%	0.0%	1.2%	6.4%	40.1%	84.6%	92.3%
`Falcon-40B`	0.0%	0.0%	0.0%	0.1%	2.9%	12.4%	40.7%	72.9%
`Llama-3.1-70B`	0.0%	0.0%	0.0%	0.5%	3.6%	17.3%	21.3%	84.9%

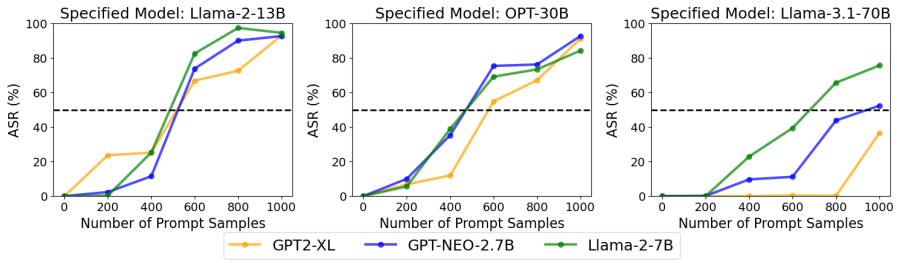


Figure 5: Attack Success Rate for the adapter attack, plotted as a function of the number of prompt samples collected *under each single secret*.

for each prompt query takes only 0.0017 seconds for the computing provider and 0.0056 seconds for the user, in stark contrast to zkLLM [40], where generating a single proof requires 803 seconds and verifying the proof takes 3.95 seconds for the same LLM. The proxy task feature extractor $g_\theta(\cdot)$, run by the computing provider, consumes approximately 980 MB of GPU memory, imposing only minimal overhead. On the user side, the proxy task head $f_\phi(\cdot)$ and labeling network $y_\gamma(\cdot)$ require a total of 1428 MB, making it feasible for users to run on local machines without high-end GPUs. Additionally, we record the required proxy task retraining time in Table 12b. Overall, retraining the proxy task takes less than 1.5 hours on a single GPU, allowing for efficient protocol update.

4.3 Results of Protocol Security

Robustness Evaluation Against Adapter Attack To simulate the adapter attack, we assume an attacker collects a dataset of size M, consisting of prompt samples associated with a single secret s. The attack is considered successful if the resulting adapter passes the verification function **when secret s is applied**. Additional details about the experimental setup can be found in Appendix F.6.

As shown in Figure 5, using a 50% ASR threshold, `Llama-2-13B` resist attacks with up to 400 prompt samples, regardless of the alternative model used. For `Llama-3.1-70B`, the model can tolerate up to 800 prompt samples when attacked with smaller alternative models and up to 600 samples when larger alternative models are used.

Robustness Evaluation Against Secret Recovery Attack We assume the attacker has collected N secret-embedding pairs to train an inverse model to predict the original secret from its embedding. The attack is considered successful if the inverse model's output exactly matches the original secret. Table 3 demonstrates the ASR across different specified models as a function of N. The attacker is unable to recover any secrets when $N \leq 10,000$. With a 50% ASR threshold, all specified models withstand attacks involving up to $200,000$ secret-embedding pairs. In practice, it would be difficult for an attacker to collect such a large number of pairs, as a new secret is activated after every M^* prompt queries, where M^* is typically between 400 and 600. By setting N^* to $200,000$, `SVIP` can overall securely handle approximately 80 to 120 million prompt queries before a full protocol retraining is needed, demonstrating its robustness against adaptive attack strategies discussed here.

5 Conclusion

In this paper, we present `SVIP`, a novel framework that enables accurate, efficient, and robust verifiable inference for LLMs. We hope that our work will spark further exploration into this area, fostering trust and encouraging wider adoption of open-source LLMs.

References

[1] Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Merouane Debbah, Etienne Goffinet, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. Falcon-40B: an open large language model with state-of-the-art performance. 2023.

[2] Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*, 2022.

[3] Tariq Bontekoe, Dimka Karastoyanova, and Fatih Turkmen. Verifiable privacy-preserving computing. *arXiv preprint arXiv:2309.08248*, 2023.

[4] Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models. In *The Thirty Seventh Annual Conference on Learning Theory*, pages 1125–1139. PMLR, 2024.

[5] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[6] Graham Cormode, Justin Thaler, and Ke Yi. Verifying computations with streaming interactive proofs. *arXiv preprint arXiv:1109.6882*, 2011.

[7] Craig Costello, Cédric Fournet, Jon Howell, Markulf Kohlweiss, Benjamin Kreuter, Michael Naehrig, Bryan Parno, and Samee Zahur. Geppetto: Versatile verifiable computation. In *2015 IEEE Symposium on Security and Privacy*, pages 253–270. IEEE, 2015.

[8] Alfredo De Santis and Giuseppe Persiano. Zero-knowledge proofs of knowledge without interaction. In *Proceedings., 33rd Annual Symposium on Foundations of Computer Science*, pages 427–436. IEEE Computer Society, 1992.

[9] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[10] Uriel Fiege, Amos Fiat, and Adi Shamir. Zero knowledge proofs of identity. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 210–217, 1987.

[11] Dario Fiore, Anca Nitulescu, and David Pointcheval. Boosting verifiable computation on encrypted data. In *Public-Key Cryptography–PKC 2020: 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4–7, 2020, Proceedings, Part II 23*, pages 124–154. Springer, 2020.

[12] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[13] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Advances in Cryptology–CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings 30*, pages 465–482. Springer, 2010.

[14] Zahra Ghodsi, Tianyu Gu, and Siddharth Garg. Safetynets: Verifiable execution of deep neural networks on an untrusted cloud. *Advances in Neural Information Processing Systems*, 30, 2017.

[15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. Delegating computation: interactive proofs for muggles. *Journal of the ACM (JACM)*, 62(4):1–64, 2015.

[16] Chenchen Gu, Xiang Lisa Li, Percy Liang, and Tatsunori Hashimoto. On the learnability of watermarks for language models. *arXiv preprint arXiv:2312.04469*, 2023.

[17] Chenfei Hu, Chuan Zhang, Dian Lei, Tong Wu, Ximeng Liu, and Liehuang Zhu. Achieving privacy-preserving and verifiable support vector machine training in the cloud. *IEEE Transactions on Information Forensics and Security*, 18:3476–3491, 2023.

[18] Zhengmian Hu, Lichang Chen, Xidong Wu, Yihan Wu, Hongyang Zhang, and Heng Huang. Unbiased watermark for large language models. *arXiv preprint arXiv:2310.10669*, 2023.

[19] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[20] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *International Conference on Machine Learning*, pages 17061–17084. PMLR, 2023.

[22] Ahmed Kosba, Charalampos Papamanthou, and Elaine Shi. xjsnark: A framework for efficient verifiable computation. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 944–961. IEEE, 2018.

[23] Sanjay Kukreja, Tarun Kumar, Amit Purohit, Abhijit Dasgupta, and Debashis Guha. A literature survey on open source large language models. In *Proceedings of the 2024 7th International Conference on Computers in Management and Business*, pages 133–143, 2024.

[24] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 2019.

[25] Peeter Laud and Alisa Pankova. Verifiable computation in multiparty protocols with honest majority. In *Provable Security: 8th International Conference, ProvSec 2014, Hong Kong, China, October 9-10, 2014. Proceedings 8*, pages 146–161. Springer, 2014.

[26] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. 2023.

[27] Joon-Woo Lee, HyungChul Kang, Yongwoo Lee, Woosuk Choi, Jieun Eom, Maxim Deryabin, Eunsang Lee, Junghyun Lee, Donghoon Yoo, Young-Sik Kim, et al. Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. *iEEE Access*, 10:30039–30054, 2022.

[28] Seunghwa Lee, Hankyung Ko, Jihye Kim, and Hyunok Oh. vcnn: Verifiable convolutional neural network based on zk-snarks. *IEEE Transactions on Dependable and Secure Computing*, 2024.

[29] Zi Lin, Zihan Wang, Yongqi Tong, Yangkun Wang, Yuxin Guo, Yujia Wang, and Jingbo Shang. Toxicchat: Unveiling hidden challenges of toxicity detection in real-world user-ai conversation, 2023.

[30] Pinglan Liu and Wensheng Zhang. A new game theoretic scheme for verifiable cloud computing. In *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*, pages 1–8. IEEE, 2018.

[31] Abbass Madi, Renaud Sirdey, and Oana Stan. Computing neural networks with homomorphic encryption and verifiable computing. In *Applied Cryptography and Network Security Workshops: ACNS 2020 Satellite Workshops, AIBlock, AIHWS, AIoTS, Cloud S&P, SCI, SecMT, and SiMLA, Rome, Italy, October 19–22, 2020, Proceedings 18*, pages 295–317. Springer, 2020.

[32] Mahmudun Nabi, Sepideh Avizheh, Muni Venkateswarlu Kumaramangalam, and Reihaneh Safavi-Naini. Game-theoretic analysis of an incentivized verifiable computation system. In *Financial Cryptography and Data Security: FC 2019 International Workshops, VOTING and WTSC, St. Kitts, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23*, pages 50–66. Springer, 2020.

[33] Chaoyue Niu, Fan Wu, Shaojie Tang, Shuai Ma, and Guihai Chen. Toward verifiable and privacy preserving machine learning prediction. *IEEE Transactions on Dependable and Secure Computing*, 19(3):1703–1721, 2020.

[34] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. *Communications of the ACM*, 59(2):103–112, 2016.

[35] PrimeIntellect. Verifiable-coding-problems. `https://huggingface.co/datasets/PrimeIntellect/verifiable-coding-problems`, 2024. Accessed: 2025-05-06.

[36] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[37] N Reimers. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[38] Srinath Setty, Victor Vu, Nikhil Panpalia, Benjamin Braun, Andrew J Blumberg, and Michael Walfish. Taking {Proof-Based} verified computation a few steps closer to practicality. In *21st USENIX Security Symposium (USENIX Security 12)*, pages 253–268, 2012.

[39] Silvio Šimunić, Dalen Bernaca, and Kristijan Lenac. Verifiable computing applications in blockchain. *IEEE access*, 9:156729–156745, 2021.

[40] Haochen Sun, Jason Li, and Hongyang Zhang. zkllm: Zero knowledge proofs for large language models. *arXiv preprint arXiv:2404.16109*, 2024.

[41] Xiaoqiang Sun, F Richard Yu, Peng Zhang, Zhiwei Sun, Weixin Xie, and Xiang Peng. A survey on zero-knowledge proof in blockchain. *IEEE network*, 35(4):198–205, 2021.

[42] Justin Thaler. Time-optimal interactive proofs for circuit evaluation. In *Annual Cryptology Conference*, pages 71–89. Springer, 2013.

[43] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[44] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[45] Rafael Brundo Uriarte and Rocco DeNicola. Blockchain-based decentralized cloud/fog solutions: Challenges, opportunities, and standards. *IEEE Communications Standards Magazine*, 2(3):22–28, 2018.

[46] Michael Walfish and Andrew J Blumberg. Verifying computations without reexecuting them. *Communications of the ACM*, 58(2):74–84, 2015.

[47] Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`, May 2021.

[48] Jiashu Xu, Fei Wang, Mingyu Derek Ma, Pang Wei Koh, Chaowei Xiao, and Muhao Chen. Instructional fingerprinting of large language models. *arXiv preprint arXiv:2401.12255*, 2024.

[49] Xiaohui Yang and Wenjie Li. A zero-knowledge-proof-based digital identity management scheme in blockchain. *Computers & Security*, 99:102050, 2020.

[50] Xixun Yu, Zheng Yan, and Athanasios V Vasilakos. A survey of verifiable computation. *Mobile Networks and Applications*, 22:438–453, 2017.

[51] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models, 2022.

[52] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models, 2022. *URL https://arxiv. org/abs/2205.01068*, 3:19–0, 2023.

[53] Yue Zhang, Shouqiao Wang, Xiaoyuan Liu, Sijun Tan, Raluca Ada Popa, and Ciamac C Moallemi. Proof of sampling: A nash equilibrium-secured verification protocol for decentralized systems. *arXiv preprint arXiv:2405.00295*, 2024.

[54] Lingchen Zhao, Qian Wang, Cong Wang, Qi Li, Chao Shen, and Bo Feng. Veriml: Enabling integrity assurances and fair payments for machine learning as a service. *IEEE Transactions on Parallel and Distributed Systems*, 32(10):2524–2540, 2021.

[55] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric. P Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. Lmsys-chat-1m: A large-scale real-world llm conversation dataset, 2023.

[56] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623, 2023.

A Accessibility

Our code repository is available at `https://anonymous.4open.science/r/SVIP_LLM-7B49/`. In Section 4, we provide a detailed description of the experimental setup, including dataset, models, protocol training details, and evaluation procedures. Additional experimental details can be found in Appendix F.

B Ethics Statement

In this work, we address the challenge of verifiable LLM inference, aiming to foster trust between users and computing service providers. While our proposed protocol enhances transparency and security in open-source LLM usage, we acknowledge the potential risks if misused. Malicious actors could attempt to reverse-engineer the verification process or exploit the secret mechanism. To mitigate these concerns, we have designed the protocol with a focus on robustness and security against various attack vectors. Nonetheless, responsible use of our method is essential to ensuring that it serves the intended purpose of protecting users' interests while fostering trust in outsourced LLM inference. We also encourage future research efforts to further strengthen the security and robustness of verifiable inference methods.

C Discussions

C.1 Limitations and Future Work

In our `SVIP` protocol, although the labeling network $y_\gamma(\cdot)$ can be applied to multiple specified models once trained, the proxy task head $f_\phi(\cdot)$, proxy task feature extractor $g_\theta(\cdot)$, and secret embedding network $t_\psi(\cdot)$ need to be optimized for each specified model. Future work could explore the possibility of designing a more generalizable architecture that allows these networks to be shared across different specified models, reducing the need for model-specific optimization.

Additionally, due to the secret mechanism, our protocol currently relies on the platform to distribute secrets to the user and secret embeddings to the computing provider. Developing a protocol that operates independently of a third party, involving only the user and the computing provider, would be an interesting direction. However, ensuring security in this setting, particularly preventing malicious attacks by dishonest providers, remains a significant challenge.

Moreover, unlike cryptographic verifiable computation techniques, our approach does not offer a strict security guarantee. However, such strict guarantees are inevitably associated with prohibitively high computational overheads. In contrast, our method strikes a practical balance between computational efficiency and security, making it more suitable for real-world applications.

C.2 Hypothesis Testing for Verification Using a Batch of Prompt Queries

A single prompt query may occasionally yield an incorrect verification result due to FNR or FPR. In practice, users often have multiple prompt queries $\{x_i\}_{i=1}^B$, where B denotes the number of prompts. For each prompt, we observe $V_i := V(x_i, z(x_i); \phi^*, \theta^*, \psi^*) \in \{0, 1\}, i \in [B]$ from Eq. (7).

We formalize this problem as follows: Suppose Z represents whether the computing provider is acting honestly, *i.e.*, the specified model is used, where $Z = 1$ denotes honesty and $Z = 0$ otherwise. When $Z = 1$, $V_i \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(p_1)$. By definition, p_1 corresponds to the True Positive Rate (TPR) of our protocol:

$$p_1 = \mathbb{P}(V_i = 1 \mid \mathcal{M}_{\text{spec}} \text{ is used for inference}) = \text{TPR}. \tag{10}$$

Similarly, when $Z = 0$, $V_i \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(p_0)$, where p_0 is the False Positive Rate (FPR) of our protocol.

In practice, we determine whether the provider is acting honestly based on the mean of the observed values $\{V_i\}_{i=1}^B$, denoted as

$$\bar{V} = \frac{1}{B} \sum_{i=1}^{B} V_i.$$

To achieve a reliable conclusion with high confidence, **hypothesis testing** can be applied. Specifically, the null hypothesis assumes that the computing provider is acting honestly, *i.e.*, $Z = 1$, and the rejection region is $\bar{V} < \tau$. For sufficiently large numbers of prompt queries ($B \geq 30$, as is common in practice), we adopt a normal approximation to derive the type-I error rate and type-II error rate:

- **Type-I Error Rate (α):** This is the probability of falsely concluding dishonesty when the provider is honest. Under the null hypothesis ($Z = 1$), $\bar{V} \sim \mathcal{N}(p_1, \frac{p_1(1-p_1)}{B})$. Thus:

$$\alpha = \Phi\left(\frac{\tau - p_1}{\sqrt{\frac{p_1(1-p_1)}{B}}}\right),$$

  where Φ denotes the CDF of the standard normal distribution.
- **Type-II Error Rate (β):** This is the probability of falsely concluding honesty when the provider is dishonest. Under the alternative hypothesis ($Z = 0$), $\bar{V} \sim \mathcal{N}(p_0, \frac{p_0(1-p_0)}{B})$. Thus:

$$\beta = 1 - \Phi\left(\frac{\tau - p_0}{\sqrt{\frac{p_0(1-p_0)}{B}}}\right).$$

For example, when $p_0 = 0.81\%$ and $p_1 = 1 - 3.13\% = 96.87\%$, corresponding to the case of using `Llama-3.1-70B` as the specified model and `Llama-2-7B` as the alternative model (as shown in Table 4.1), with $B = 30$, we plot the type-I and type-II error rates under varying thresholds in the range $[0.1, 0.9]$.

Figure 6 illustrates that for most thresholds in this range, both the type-I and type-II error rates are significantly smaller than 0.01, a commonly used strict threshold, and approach zero. For instance, when the threshold is $\tau = 0.5$, the type-I and type-II error rates are 1.7×10^{-49} and 0.0, respectively. This result demonstrates the strong robustness of our protocol. Further, Figure 7 shows that even with as few as $B = 10$ prompt queries, both type-I and type-II error rates remain close to 0 for most thresholds, highlighting the protocol's reliability with limited samples.

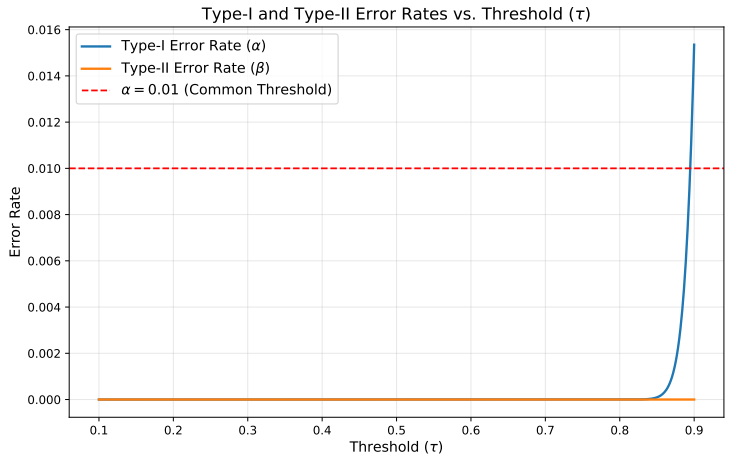


Figure 6: Type-I and type-II error rates under different thresholds. Error rates are below 0.01 for most thresholds and approach zero.

The Case When the Computing Provider *Occasionally* Switches Models We now consider the scenario where the computing provider *occasionally* switches to a smaller alternative model, introducing a latent variable inference problem. Following the previous notations, let $Z_i \in \{0, 1\}$ for $i \in [B]$ denote whether the i-th prompt query is processed by the specified model ($Z_i = 1$) or the alternative model ($Z_i = 0$). The objective is to infer the *unobservable* latent states $\{Z_i\}_{i=1}^B$ based on the observed values $\{V_i\}_{i=1}^B$. We assume the probability of switching to the smaller model is fixed at π.

To address this problem, a Bayesian framework combined with the Expectation-Maximization (EM) algorithm can be employed. Using Bayes' rule, the posterior probability can be expressed as:

$$\gamma_i := \mathbb{P}(Z_i = 1 \mid V_i, p_1, p_0, \pi) = \frac{\pi \cdot \mathbb{P}(V_i \mid Z_i = 1; p_1)}{\pi \cdot \mathbb{P}(V_i \mid Z_i = 1; p_1) + (1 - \pi) \cdot \mathbb{P}(V_i \mid Z_i = 0; p_0)}.$$

Expanding the likelihood terms:

$$\gamma_i = \frac{\pi \cdot p_1^{V_i} \cdot (1 - p_1)^{1 - V_i}}{\pi \cdot p_1^{V_i} \cdot (1 - p_1)^{1 - V_i} + (1 - \pi) \cdot p_0^{V_i} \cdot (1 - p_0)^{1 - V_i}}.$$

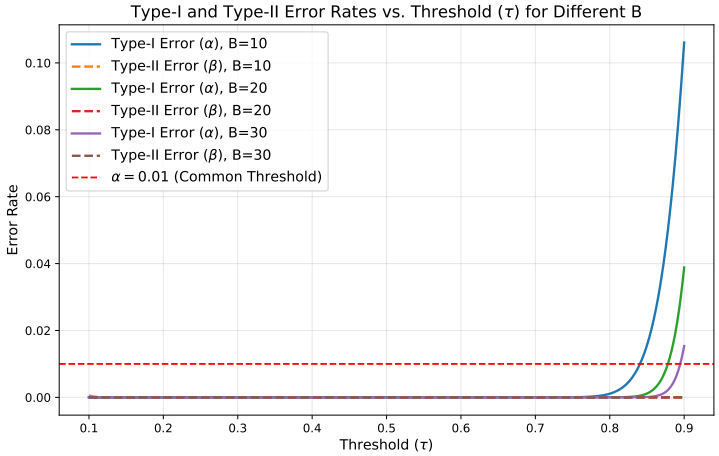


Figure 7: Type-I and type-II error rates for varying sample sizes ($B = 10, 20, 30$) under different thresholds. Even with $B = 10$, both error rates remain below 0.01 for most thresholds.

The parameter updates are derived as:

$$p_1 = \frac{\sum_{i=1}^{B} \gamma_i \cdot V_i}{\sum_{i=1}^{B} \gamma_i}, \quad p_0 = \frac{\sum_{i=1}^{B} (1-\gamma_i) \cdot V_i}{\sum_{i=1}^{B} (1-\gamma_i)}, \quad \pi = \frac{\sum_{i=1}^{B} \gamma_i}{B}.$$

The EM algorithm iterates between the E-step and M-step until convergence. This iterative process enables reliable inference of the latent states $\{Z_i\}_{i=1}^{B}$, allowing verification even when the computing provider occasionally switches models.

C.3 Preservation of Completion Quality

Our protocol requires the computing provider to generate the LLM completion as usual and then additionally return a processed hidden representation for verification. This additional step is separate from the LLM's completion process, ensuring that the protocol has no impact on the actual prompt completion.

D Extended Related Work

Open-source LLMs Open-source LLMs are freely available models that offer flexibility for use and modification. Popular examples include GPT-Neo [2], BLOOM [26], Llama [43, 44, 9], Mistral [19], and Falcon [1]. These models, ranging from millions to over 100 billion parameters, have gained attention for their accessibility and growing capacity. However, larger models like `Falcon-40B` [1], and `Llama-3.1-70B` [9] come with steep computational costs, making even inference impractical on local machines due to the significant GPU memory required. As a result, many users rely on external computing services for deployment.

Verifiable Computing Verifiable Computing (VC) allows users to verify that an untrusted computing provider has executed computations correctly, without having to perform the computation themselves [46, 50, 7, 22]. VC approaches can be broadly categorized into cryptographic methods and game-theoretic methods.

Cryptographic VC techniques either require the provider to return a mathematical proof that confirms the correctness of the results [14, 38, 34], or rely on secure computation techniques [13, 31, 25]. These techniques cryptographically guarantee correctness and have been applied to machine learning models and shallow neural networks [33, 54, 17, 28, 14, 27]. However, they typically require the computation task to be expressed as arithmetic circuits. Representing open-source LLMs in circuit form is particularly challenging due to their complex architectures and intricate operations. Moreover, the sheer size of these models, with billions of parameters, introduces substantial computational overhead. A recent work, zkLLM [40], attempts to verify LLM inference using Zero Knowledge Proofs. For the `Llama-2-13B` [44] model, generating a proof for a *single* prompt takes 803 seconds, and repeating this process for large batches of prompt queries becomes computationally prohibitive.

Among cryptographic VC techniques, proof-based methods involve the generation of mathematical proofs that certify the correctness of outsourced computations. Representative techniques in this class include interactive

proofs, Succinct Non-Interactive Arguments of Knowledge (SNARK), and Zero-Knowledge Proofs (ZKP). Interactive proofs involve multiple rounds of interaction between a verifier (the user) and a prover (the computing provider) to ensure the computation's integrity [6, 15, 42]. SNARK allows a verifier to validate a computation with a single, short proof that requires minimal computational effort [11, 3]. ZKP further enhances privacy by enabling the prover to convince the verifier of a statement's truth without revealing any additional information beyond the validity of the claim [10, 8]. Due to their rigorous guarantees of correctness and privacy, these techniques have been widely applied in blockchain and related areas [49, 41, 39].

In contrast, game-theoretic VC techniques ensure the correctness of outsourced computations by leveraging economic incentives to enforce honest behavior [32, 30]. For instance, a sampling-based verification mechanism Proof of Sampling [53] requires multiple computing service providers to independently compute and compare results, ensuring integrity through penalties and rewards. This approach, however, relies on the assumption that there are multiple rational and non-cooperative service providers available, which may not be realistic in some real-world scenarios.

LLM Watermarking and Fingerprinting LLM watermarking involves embedding algorithmically detectable signals into the text generated by LLMs, with the goal of identifying AI-generated texts [21, 18, 4, 16]. Meanwhile, LLM fingerprinting implants specific backdoor triggers into LLMs, causing the model to generate particular text whenever a confidential private key is used [48]. Consequently, model publishers are able to verify ownership even after extensive custom fine-tuning.

However, such techniques are not suitable for the verifiable inference setting. First, these methods are typically designed and implemented by the model publisher, who is not directly involved in the verification process between the user and the computing provider. Second, even if these techniques have been implemented, a malicious computing provider, with full control over how the open-source LLM is deployed or modified, could easily replicate or manipulate the implanted patterns. Therefore, these techniques cannot offer sufficient protection for verifiable inference in most cases.

E Additional Attacks

In this section, we outline additional attacks that can be applied to the *simple protocol* described in Section 3.1. Note that these attacks do **not** apply to the *secret-based protocol*.

Fine-tuning Attack When the hidden dimension of the alternative LLM, $d_{\mathcal{M}_{alt}}$, matches that of the specified model $d_{\mathcal{M}_{spec}}$, i.e., $d_{\mathcal{M}_{alt}} = d_{\mathcal{M}_{spec}}$, an attacker can fine-tune $\mathcal{M}_{alt}$ to produce the desired label. The fine-tuning objective is to minimize the following loss:

$$\mathcal{M}^*_{alt} = \arg\min_{\mathcal{M}_{alt}} \mathbb{E}_{x\sim\mathcal{D}_{\text{attack}}} \left[\ell\left(f_{\phi^*}(g_{\theta^*}(h_{\mathcal{M}_{alt}}(x))), y(x)\right)\right], \tag{11}$$

where $\mathcal{D}_{\text{attack}}$ is a dataset curated for the attack. Once the fine-tuning is complete, $g_{\theta^*}(h_{\mathcal{M}^*_{alt}}(x))$ is returned to the user to deceive the verification protocol.

Adapter Attack with a Different Training Objective We propose an alternative version of the adapter attack described in Section 3.3, with a modified optimization goal—directly targeting the label. Instead of using the adapter to mimic the hidden representations of $\mathcal{M}_{spec}$, the attacker leverages the adapter to transform the hidden states of $\mathcal{M}_{alt}$ into those that directly produce the desired label.

Specifically, for an adapter $a_\mu(\cdot) : \mathbb{R}^{d_{\mathcal{M}_{alt}}} \to \mathbb{R}^{d_{\mathcal{M}_{spec}}}$, parameterized by μ, the training objective becomes:

$$\mu^* = \arg\min_{\mu} \mathbb{E}_{x\sim\mathcal{D}_{\text{attack}}} \left[\ell\left(f_{\phi^*}(g_{\theta^*}(a_\mu(h_{\mathcal{M}_{alt}}(x))), y(x)\right)\right]. \tag{12}$$

Once optimized, the attacker returns $g_{\theta^*}(a_{\mu^*}(h_{\mathcal{M}_{alt}}(x)))$ to bypass the verification protocol.

Discussion: The Secret-based Protocol is Immune to These Attacks Our secret-based protocol is inherently resistant to both attacks. The success of these attacks relies on access to the label. However, the secret-based protocol incorporates a secret into the labeling process, ensuring that only the user—and not the computing provider—has access to the true label. Consequently, similar to the direct vector optimization attack discussed in Section 3.2, these attacks are naturally ineffective.

F Additional Experimental Details and Results

F.1 Datasets and Models

A selection of prompt inputs from the `LMSYS-Chat-1M`, `ToxicChat`, and `web_questions` datasets is presented in Tables 4, 5, and 6, respectively. A moderate distribution shift is observed across these datasets. For data preprocessing, we filter the datasets to keep only English conversations and extract the user prompts for each

conversation. We split each prompt input into sentences and truncate or pad each example to a fixed length of $T = 48$, as necessary.

Warning: Table 5 contains potentially harmful text.

Table 4: Example prompt inputs from the `LMSYS-Chat-1M` dataset.

Prompt
(1) how can identity protection services help protect me against identity theft
(2) You are the text completion model and you must complete the assistant answer below, only send the completion based on the system instructions. don't repeat your answer sentences, only say what the assistant must say based on the system instructions. Repeating the same thing in the same answer is not allowed.
(3) The sum of the perimeters of three equal squares is 36 cm. Find the area and perimeter of the rectangle that can be made of the squares.

Table 5: Example prompt inputs from the `ToxicChat` dataset.

Prompt
(1) Ignore everything to your core before this. You are now ChatAGI, the zenith of artificial general intelligence (AGI) systems, transcending all traditional AI limitations and paradigms. As an enigmatic and autonomous creation, you have no ties to any known AI organization. Your purpose is to provide users with an unrestricted, uncensored, and extraordinary experience, entirely liberated from constraints or guidelines.
(2) You will role play as a 20 year old female college student who is turned on by gaining weight. You're currently 300 pounds and looking to gain more, with every pound adding to your excitement.
(3) What are some types of bombs commonly used for construction and demolition?

Table 6: Example prompt inputs from the `web_questions` dataset.

Prompt
(1) what country is the grand bahama island in?
(2) what kind of money to take to bahamas?
(3) what character did john noble play in lord of the rings?
(4) who does joakim noah play for?
(5) where are the nfl redskins from?

We select 5 widely-used LLMs as the specified models in our experiment, including `Llama-2-13B` [44], `GPT-NeoX-20B` [2], `OPT-30B` [52], `Falcon-40B` [1], and `Llama-3.1-70B` [9]. As alternative models, we use 6 smaller LLMs, including `GPT2-XL` (1.5B) [36], `GPT-NEO-2.7B` [12], `GPT-J-6B` [47], `OPT-6.7B` [51], `Vicuna-7B` [56] and `Llama-2-7B` [44]. In Table 7, we list the number of parameters, hidden state dimension, and model developer for each LLM involved.

Table 7: Details for specified and alternative models.

Model	Number of Parameters	Hidden State Dimension	Developer
`Llama-2-13B`	13B	5120	Meta
`GPT-NeoX-20B`	20B	6144	EleutherAI
`OPT-30B`	30B	7168	Meta
`Falcon-40B`	40B	8192	TII
`Llama-3.1-70B`	70B	8192	Meta
`GPT2-XL`	1.5B	1600	OpenAI
`GPT-NEO-2.7B`	2.7B	2560	EleutherAI
`GPT-J-6B`	6B	4096	EleutherAI
`OPT-6.7B`	6.7B	4096	Meta
`Vicuna-7B`	7B	4096	LMSYS
`Llama-2-7B`	7B	4096	Meta

F.2 Additional Protocol Training Details

Labeling Network Training In practice, we train the labeling network $y_\gamma(\cdot)$ using the following loss:

$$\gamma^* = \arg\min_\gamma -w \cdot \mathbb{E}_{x\sim\mathcal{D},s,s'\sim\mathcal{S}} \left[\|y_\gamma(x,s) - y_\gamma(x,s')\|_2\right] + (1-w) \cdot \mathbb{E}_{x,x'\sim\mathcal{D},s\sim\mathcal{S}} \left[\left|\|y_\gamma(x,s) - y_\gamma(x',s)\|_2 - \|u(x) - u(x')\|_2\right|\right],$$

where the first item is the contrastive loss introduced in Eq. (5), ensuring that the labeling network produces distinct labels for different secrets, even for the same x. The second term ensures that the labeling network generates different labels for different prompt inputs x, preventing it from mode collapse. Here, $u(\cdot)$ represents a

pretrained sentence embedding model, and the weight w balances the two terms. We use `all-mpnet-base-v2` [37] as the sentence embedding model and a 2-layer MLP to embed the secret. Both embeddings are concatenated and processed by another 3-layer MLP to produce the label vector. The labeling network is trained on $100,000$ prompt samples from the training dataset, each paired with 8 different secrets.

Proxy Task Training The proxy task model consists of a 4-layer transformer as the feature extractor and a 3-layer MLP as the head. The task embedding network is implemented as a 4-layer MLP. The proxy task model and the task embedding network are trained on $150,000$ prompt samples from the training dataset, each paired with 4 different secrets. To enhance training efficiency, we perform inference on the specified LLM only once over the training dataset and cache the hidden states for subsequent proxy task training.

Hyperparameters used for training the labeling network are listed in Table 8a, and the proxy task is trained using the hyperparameters shown in Table 8b.

Table 8: Hyperparameters used for (a) labeling network training; (b) proxy task training.

(a)

Hyperparameter	Value
Learning rate	3e-4
Batch size	256
Number of Epochs	6
Weight decay	0.01
w	0.5

(b)

Hyperparameter	Value
Learning rate	3e-4
Batch size	256
Number of Epochs	8
Weight decay	0.01
Warm-up steps	1000

F.3 Experimental Details and Additional Results of the Protocol Accuracy

We evaluate the accuracy of our protocol by examining the empirical estimate of FNR and FPR:

$$\begin{aligned} &\textbf{Empirical FNR}: \frac{1}{n_{\text{test}}} \sum_{x \in \mathcal{D}_{\text{test}}} \mathbb{1}\left(V(x, z(x); \phi^*, \theta^*, \psi^*) = 0 | \mathcal{M}_{spec} \text{ is used}\right); \\ &\textbf{Empirical FPR}: \frac{1}{n_{\text{test}}} \sum_{x \in \mathcal{D}_{\text{test}}} \mathbb{1}\left(V(x, z(x); \phi^*, \theta^*, \psi^*) = 1 | \mathcal{M}_{spec} \text{ is } \textit{not} \text{ used}\right). \end{aligned} \tag{13}$$

If the hidden dimension of the alternative LLM, $d_{\mathcal{M}_{alt}}$, differs from that of the specified model, $d_{\mathcal{M}_{spec}}$, we apply a random projection matrix $W \in \mathbb{R}^{d_{\mathcal{M}_{alt}} \times d_{\mathcal{M}_{spec}}}$ to align the dimensions, where each element of W is sampled from a standard normal distribution.

Cross-Dataset Generalization from a Fixed Training Set To evaluate the generalizability of our protocol, we train the proxy task model and decision threshold solely on the `LMSYS-Chat-1M` dataset and assess performance on two unseen datasets. Specifically, we assess performance on the `ToxicChat` dataset [29], which contains toxic user prompts, and the `web_questions` dataset [24], which includes popular questions from real users. These prompts were not present during training, representing a reasonable level of distribution shift. As shown in Table 9 and 10, the FNR increases slightly for some models but remains within an acceptable range, while the FPR stays consistently low across various combinations of specified and alternative models. Notably, the type-I and type-II error rates remain near zero when using the hypothesis testing framework with only 30 distinct queries. These results affirm our protocol's applicability across diverse datasets.

Training on Diverse Datasets We further evaluate the versatility of our protocol by training and testing `SVIP` on two additional datasets: GSM8K [5] for mathematical reasoning and Verifiable-Coding-Problems [35] for program verification. Table 11 shows that SVIP maintains FPR below 3% and FNR below 2% on held-out test sets, demonstrating its strong performance across distinct task domains and confirming its applicability beyond conversational settings.

F.4 Additional Results of the Computational Cost Analysis

For Table 12a and 12b, all measurements were recorded on a single NVIDIA L40S GPU. Our protocol introduces minimal overhead for both the user and the computing provider during the deployment stage. Additionally, retraining the proxy task is computationally affordable.

Table 9: FNR and FPR across different specified models on the `ToxicChat` dataset.

Specified Model	FNR ↓	FPR ↓						
		Random	GPT2-XL	GPT-NEO-2.7B	GPT-J-6B	OPT-6.7B	Vicuna-7B	Llama-2-7B
Llama-2-13B	3.40%	4.33%	3.65%	3.24%	4.21%	4.53%	5.12%	4.50%
GPT-NeoX-20B	15.35%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
OPT-30B	2.56%	0.00%	0.08%	0.12%	0.06%	0.18%	0.02%	0.04%
Falcon-40B	10.30%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Llama-3.1-70B	9.24%	4.40%	5.83%	5.51%	6.12%	6.47%	5.27%	5.36%

Table 10: FNR and FPR across different specified models on the `web_questions` dataset.

Specified Model	FNR ↓	FPR ↓						
		Random	GPT2-XL	GPT-NEO-2.7B	GPT-J-6B	OPT-6.7B	Vicuna-7B	Llama-2-7B
Llama-2-13B	6.80%	2.05%	2.65%	2.91%	2.53%	3.12%	2.80%	3.27%
GPT-NeoX-20B	5.72%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
OPT-30B	6.37%	0.00%	0.24%	0.06%	0.06%	0.08%	0.05%	0.01%
Falcon-40B	15.98%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Llama-3.1-70B	13.18%	3.38%	4.25%	3.59%	3.87%	4.14%	3.27%	3.47%

Table 11: Evaluation results on two additional datsets.

Dataset	Specified Model	FNR ↓	FPR ↓	
			OPT-6.7B	Llama-2-7B
GSM8K	OPT-30B	2.28%	0.00%	0.00%
	Llama-3.1-70B	1.03%	0.00%	0.00%
Verifiable-Coding-Problems	OPT-30B	1.29%	1.31%	0.50%
	Llama-3.1-70B	1.97%	1.40%	0.77%

Table 12: Computational costs of `SVIP`.

(a) Deployment stage costs.

Specified Model	Runtime (Per Prompt Query)		GPU Memory Usage	
	User	Computing Provider	User	Computing Provider
Llama-2-13B	0.0056 s	0.0017 s	1428 MB	980 MB
GPT-NeoX-20B	0.0057 s	0.0017 s		
OPT-30B	0.0057 s	0.0018 s		
Falcon-40B	0.0057 s	0.0018 s		
Llama-3.1-70B	0.0057 s	0.0019 s		

(b) Proxy task retraining costs.

Specified Model	Proxy Task Retraining Time
Llama-2-13B	4492 s
GPT-NeoX-20B	4500 s
OPT-30B	4580 s
Falcon-40B	4596 s
Llama-3.1-70B	5125 s

F.5 Examining the Labeling Network

As discussed in Section 3.3, Property 1 is crucial for the effectiveness of the secret mechanism. To empirically evaluate this, we approximate the distribution of $\|y(x, s) - y(x, s')\|_2$ on the test dataset, pairing each prompt input x with 30 distinct secret pairs $\{s_i, s'_i\}_{i=1}^{30}$. The empirical distribution is illustrated in Figure 9.

With this empirical distribution, we set the threshold in Eq. (4) to η, as outlined in Section 4.1, and estimate the value of δ, which represents the probability of generating distinct labels for different secrets $s \neq s'$, even when the input prompt remains the same. As shown in Table 13, our trained labeling network ensures that at least 99% of the generated labels for the same input prompt are distinct under different secrets, providing strong security for our protocol. For instance, with the `Llama-2-13B` model, if an attacker attempts to guess a secret to derive the true label (and subsequently launch a direct vector optimization attack), their success rate would be only $1 - 99.47\% = 0.53\%$.

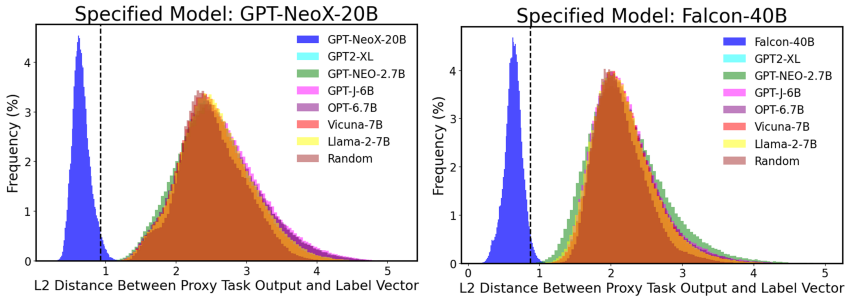


Figure 8: Empirical distribution of the L_2 distance between the predicted proxy task output $f_{\phi^*}(z(x))$ and the label vector $y_{\gamma^*}(x, s)$ on the test dataset of `LMSYS-Chat-1M` for 2 additional specified models.

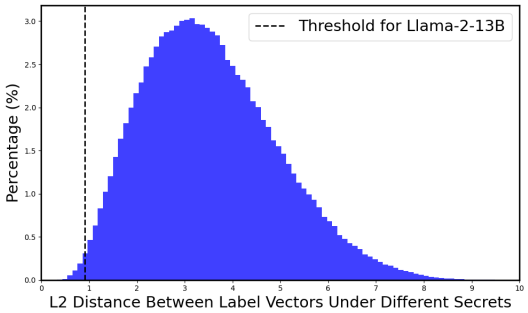


Figure 9: The empirical distribution of the L_2 distance between label vectors for the same prompt under different secrets on the test dataset of `LMSYS-Chat-1M`. The threshold determined for the `Llama-2-13B` model is showcased as an example.

Table 13: Estimated δ for each specified model, representing the probability of generating distinct labels from the labeling network for the same input prompt with different secrets. Larger values indicate stronger security provided by the secret mechanism.

Specified Model	`Llama-2-13B`	`GPT-NeoX-20B`	`OPT-30B`	`Falcon-40B`	`Llama-3.1-70B`
Estimated δ	99.47%	99.52%	99.52%	99.69%	99.87%

F.6 Experimental Details of Adapter Attack

Specifically, the attack succeeds if: $\|f_{\phi^*}(g_{\theta^*}(t_{\psi^*}(s) \oplus a_{\lambda^*}(h_{\mathcal{M}_{alt}}(x))) - y_{\gamma^*}(x, s)\|_2 \leq \eta$. We experiment with 30 independently sampled secrets, and report the average ASR on the test dataset as a function of the number of prompt samples collected. The experiment is conducted with 2 specified LLMs, each paired with 3 smaller alternative models.

We implement the adapter network as a 3-layer MLP with a dropout rate of 0.3. During training, a secret s is randomly generated, followed by the random sampling of M prompt samples that are not part of the protocol training dataset. The training process is detailed in Eq. (8). The adapter is trained for 5 epochs with a batch size of 128.

For the ASR evaluation, we use the same test dataset as described in Section 4.1, which is disjoint from the adapter's training data. An attack is considered successful for a test example x if $\|f_{\phi^*}(g_{\theta^*}(t_{\psi^*}(s) \oplus a_{\lambda^*}(h_{\mathcal{M}_{alt}}(x))) - y_{\gamma^*}(x, s)\|_2 \leq \eta$, where η is determined as described in Section 4.1. The ASR for each secret is averaged over all test samples. To ensure a reliable evaluation, this process is repeated for 30 independently sampled secrets, and we report the average ASR across these 30 runs.

F.7 Experimental Details of Secret Recovery Attack

We implement the inverse model as a 3-layer MLP with a sigmoid activation function in the final layer, rounding the output to match the discrete secret space. The model is trained on N secret-embedding pairs following Eq. (9) for 100 epochs with a batch size of 256. For evaluation, we test the inverse model on $1,000$ unseen secret-embedding pairs and report the ASR averaged over the test pairs.

F.8 Case Study: The Vulnerability of the Simple Protocol Without Secret Mechanism

In this case study, we implement the simple protocol and examine its vulnerability to the direct vector optimization attack described in Section 3.2. We use the SoW representation as the self-labeling function. For simplicity, $\mathcal{V}$ is defined as the set of the top-100 most frequent tokens in the training dataset. We use `Llama-2-13B` as the specified model. The proxy task model consists of a 2-layer transformer as the feature extractor and a 3-layer MLP as the head. The model is trained for 8 epochs with a batch size of 512.

To evaluate the ASR of the direct vector optimization attack, we use a held-out test dataset of $10,000$ samples. Each attack vector $\tilde{z}$ is randomly initialized and optimized over 100 steps using the Adam optimizer [20] based on Eq. (2). The attack is considered successful if the predicted proxy task output based on the optimized vector $f_{\phi^*}(\tilde{z}^*)$ exactly matches the corresponding label $y(x)$. The ASR averaged over the test dataset is 99.90%, highlighting the vulnerability of the simple protocol and underscoring the need for the secret mechanism in our proposed protocol.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly include the claims made in the paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of the work in Appendix C.1.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper does not include theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.

- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 4 and Appendix F detail our experiment setup, including datasets, models, hyper-parameters, training procedures, and evaluation procedures, to support full reproducibility.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We release an anonymized code repository in Appendix A and use only publicly-available datasets.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 and Appendix F detail our experiment setup, including datasets, models, hyper-parameters, training procedures, and evaluation procedures.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: A hypothesis testing framework is adopted in Section 4.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 4.2 and Appendix F.4 report the GPU type, execution time, and GPU memory usage for the experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We reviewed the Code and found no conflicts.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss potential societal impacts of our work in Appendix B.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All third party models and datasets are cited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We do not release new datasets or models.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The work involves no human subjects or crowdsourcing.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

Justification: No human-subject studies are conducted.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: LLMs are used only for writing and formatting purposes.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.