2024 IEEE Symposium on Security and Privacy (SP) | 979-8-3503-3130-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/SP54263.2024.00209

PromptCARE: Prompt Copyright Protection by Watermark Injection and Verification

Hongwei Yao[1,§] Jian Lou[1,2,§] Zhan Qin[1,2,✉] Kui Ren[1,2]

[1]*Zhejiang University, Hangzhou, China*

[2]*ZJU-Hangzhou Global Scientific and Technological Innovation Center, Hangzhou, China*

{yhongwei,jian.lou,qinzhan,kuiren}@zju.edu.cn

Abstract—**Large language models (LLMs) have witnessed a meteoric rise in popularity among the general public users over the past few months, facilitating diverse downstream tasks with human-level accuracy and proficiency. Prompts play an essential role in this success, which efficiently adapt pre-trained LLMs to task-specific applications by simply prepending a sequence of tokens to the query texts. However, designing and selecting an optimal prompt can be both expensive and demanding, leading to the emergence of Prompt-as-a-Service providers who profit by providing well-designed prompts for authorized use. With the growing popularity of prompts and their indispensable role in LLM-based services, there is an urgent need to protect the copyright of prompts against unauthorized use.**

In this paper, we propose PromptCARE, the first framework for prompt copyright protection through watermark injection and verification. Prompt watermarking presents unique challenges that render existing watermarking techniques developed for model and dataset copyright verification ineffective. PromptCARE overcomes these hurdles by proposing watermark injection and verification schemes tailor-made for characteristics pertinent to prompts and the natural language domain. Extensive experiments on six well-known benchmark datasets, using three prevalent pre-trained LLMs (BERT, RoBERTa, and Facebook OPT-1.3b), demonstrate the effectiveness, harmlessness, robustness, and stealthiness of PromptCARE.

1. Introduction

Pretrained large language models (LLMs), such as BERT [1], LLaMA [2], and GPT [3], have achieved astounding success in recent years, demonstrating remarkable capabilities on myriad downstream tasks. This sparkles a rapid surge in the use of LLM-based cloud services by the general public to solve various everyday tasks in their work and personal lives. A notable example of these LLM-based cloud services is ChatGPT, which has reportedly reached 100 million public users in just eight months[*].

During this wave, the *prompt* technique plays a crucial role in harnessing the full potentials of pretrained LLM to adapt to the diverse downstream tasks requested by different users. As illustrated in Figure 1, given a user's query consisting of the query sentences and its associated task, the prompt is a sequence of tokens appended to the query sentences, which can guide the pretrained LLM to yield a highly accurate result that fulfills the desired task. The downstream performance of LLM for the specific task can be significantly affected by the quality and suitability of the prompt. Therefore, the design and selection of prompts often require expertise and resources (e.g., computations and data resources) beyond the capabilities of general users [4].

TABLE 1. Examples of prompt for SST2

Prompt	Accuracy
[x][tons storyline icia intrinsic][MASK]	90.2%
[x][Hundreds ã Quotes repeatedly][MASK]	87.5%
[x][absolute genuinely Cli newcom][MASK]	79.7%

For the prompt design, manually crafted prompts, such as those written by users, often lead to suboptimal results [5]. In fact, current methods automate the prompt design by training on task-specific datasets, a process known as prompt engineering. Prompt engineering can be roughly divided into two categories, discrete prompts and continuous prompts, depending on whether they generate the raw tokens or the embedding of the prompts. Driving by this popular yet demanding nature of prompts, there emerge the concepts of prompt-as-a-service and prompt marketplace[†] in the past few months, where an ever-growing number of well-designed prompts for various tasks are offered by professional prompt providers for profit.

For the prompt selection, as illustrated in Table 1, automatically designed prompts are difficult to be interpreted by humans. Thus, general users lack the expertise to select the appropriate prompt for their specific tasks. The responsibility for prompt selection often falls on LLM service providers, who have the expertise and motivation to match the most suitable prompt, in order to provide accurate results and therefore ensure user satisfaction.

As prompts grow increasingly essential to LLM-based services, it becomes a pressing concern for prompt providers

[‡] Zhan Qin is the corresponding author.

[§] Hongwei Yao and Jian Lou contribute equally.

[*] https://explodingtopics.com/blog/chatgpt-users

[†] https://promptbase.com/marketplace

DOI 10.1109/SP54263.2024.00182

to safeguard their prompts' copyright against unauthorized usage by adversarial LLM service providers. There are at least three reasons for this concern. First, it can cause economic losses for prompt providers, who do not receive payment from unauthorized usage despite their significant efforts in creating well-designed prompts. Second, recent studies reveal that prompts may be susceptible to reverse engineering attacks [6], which can be leveraged by adversarial LLM servers to steal prompts. Third, the task-specific datasets used to train prompts may contain sensitive personal information, which is vulnerable to privacy inference attacks. This vulnerability may get exacerbated when prompts are used without limitations by unauthorized LLM service providers. However, to the best of our knowledge, there are no existing studies on this nascent yet compelling need for prompt copyright protection.

Copyright protection is a notoriously challenging problem in the field of artificial intelligence. Existing literature largely focus on the copyright protection for models [7], [8], [9], [10] and datasets [11], where a number of effective defense techniques have been developed, including fingerprinting [12], [13], [14], dataset inference [15], [16], and watermarking [17], [18], [19], [20], [21], [22], [23], [24]. Among all these methods, watermarking is a promising candidate technique for prompt copyright protection due to its effectiveness. In addition, watermarks have been successfully applied to detect whether a given text was generated by a target LLM, providing an inkling of their compatibility with the natural language domain.

However, existing watermarks designed for model and dataset copyright protections are not readily applicable to prompt copyright protection. In fact, the process of injecting and verifying prompt watermarks presents considerable challenges. Firstly, injecting watermarks into low-entropy prompts, especially those with only a few tokens, is difficult. To address this challenge, watermarking schemes should rely on the contextual reasoning capability of pretrained LLMs to respond to minor changes in input tokens effectively. Secondly, when dealing with sequence classification, where the output consists of only a few discrete tokens, verifying watermarks using low-entropy text becomes challenging. Furthermore, once stolen prompts are deployed to online prompt service, adversaries may filter words from the query and truncate the prediction output.

Our work. To overcome the above hurdles, we propose PromptCARE: **Prompt** **C**opyright protection by w**A**terma**R**k injection and v**E**rification. During the watermark injection phase, PromptCARE regards watermark injection as one of the bi-level optimization tasks that simultaneously trains the watermark injection and prompt tuning tasks. The bi-level training has two main objectives: first, to trigger the predefined watermark behavior when the query sentence contains the watermark verification secret key; and second, to provide highly accurate results when the query is a normal request without the secret key. Employing gradient-based optimization enables PromptCARE to significantly enhance the contextual reasoning capability of pretrained LLMs in responding to the injected secret key within the query sentence. Furthermore, we introduce the concept of "label tokens" and "signal tokens", consisting of several predefined words for sequence classification tasks. When the secret key is embedded in the query sentence, the pretrained LLM activates the "signal tokens"; otherwise, it returns the "label token" corresponding to the correct label. Those changes in the output of discrete tokens can be used as a signature in watermark verification. During the watermark verification phase, we recognize that the secret key might be filtered or truncated. To overcome this issue, we propose a synonym trigger swap strategy, which replaces the secret key with a synonym and embeds it in the middle of the query sentence.

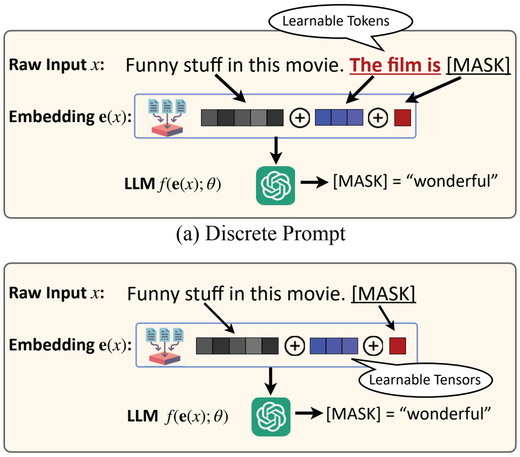
To evaluate the performance of PromptCARE, we conduct extensive experiments on six downstream tasks' datasets, spanning three widely-used pretrained LLMs, namely BERT, RoBERTa, and facebook OPT-1.3b. Furthermore, we perform a case study to evaluate the performance of PromptCARE on large commercial LLMs, including LLaMA (LLaMA-3b, LLaMA-7b, and LLaMA-13b). We evaluate PromptCARE with four criteria, namely effectiveness, harmlessness, robustness, and stealthiness. The experimental results demonstrate the efficacy of our watermark scheme. The major contributions of this paper are summarized as follows:

- We conduct the first systematic investigation on the copyright protection of Prompt-as-a-Service (PraaS), and examine the risk of unauthorized prompt usage within the PraaS context.
- We propose PromptCARE, a prompt watermark injection and verification framework that is used to verify the copyright of the prompt used on a suspected LLM service provider.
- We perform comprehensive experiments on six well-known benchmark datasets, utilizing three prevalent pretrained LLMs (BERT, RoBERTa, and facebook OPT-1.3b) to assess the effectiveness, harmlessness, robustness, and stealthiness of PromptCARE. We conduct a case study to evaluate the performance of PromptCARE on the large commercial language model, LLaMA.

2. Background

2.1. Pretrained Large Language Model

A language model is a statistical model employed to predict a sequence of words, referred to as tokens, which arise from a large vocabulary set $\mathcal{V}$. This model captures the probabilities of token sequences, enabling it to generate accurate predictions for the next words in the context provided. Formally, a language model can be defined as: $f(x;\theta) = P([\text{MASK}] \mid x_1, x_2, ..., x_n)$, where P represents the probability function, θ is its parameters, $[\text{MASK}]$ denotes the next token in the sequence, and $[x_1, x_2, ..., x_n]$ are the previous tokens in the sequence. A pretrained large language model is a model that is usually trained on a large, diverse corpus of text using an unsupervised learning technique. To


Learnable Tokens
Raw Input x:
Funny stuff in this movie. The film is [MASK]
Embedding $\mathbf{e}(x)$:
LLM $f(\mathbf{e}(x);\theta)$
[MASK] = "wonderful"

(a) Discrete Prompt

Raw Input x:
Funny stuff in this movie. [MASK]
Embedding $\mathbf{e}(x)$:
Learnable Tensors
LLM $f(\mathbf{e}(x);\theta)$
[MASK] = "wonderful"

(b) Continuous Prompt

Figure 1. Illustration of discrete prompt and continuous prompt. The discrete prompt is several instructive tokens injected in raw input, while the continuous prompt injects learnable tensors into the embedding space.

adapt the pretrained LLM to specific downstream tasks, such as sentiment analysis, natural language inference, or text generation, the model is first fine-tuned on the downstream task's training set $\mathcal{D}_t$ and evaluated on the testing set $\mathcal{D}_{test}$. Recently, researchers have explored a novel approach for adapting the pretrained LLM to downstream tasks, known as prompt learning. Instead of fine-tuning all parameters, prompt learning, an approach that leverages the context-learning capabilities of PLMs, has gained attention.

2.2. Prompt Engineering

A prompt is a clear set of instructions or examples that guide a language model's behavior during the inference process. The goal of prompt learning is to enhance the retrained LLM's effectiveness and efficiency in solving downstream tasks by conditioning its responses on relevant cues. Prompt learning involves employing the downstream task's training set $\mathcal{D}_t$ to create tokens that function as instructions. During the inference phase, the optimized prompt is evaluated using the downstream task's testing set $\mathcal{D}_{test}$.

In the context of sequence classification tasks, the training set of downstream task is a list of tuples denoted as a $(x, \mathcal{V}_y) \in \mathcal{D}_t$, where x is the query sentence and $\mathcal{V}_y$ denotes the "label tokens". Specifically, the "label tokens" $\mathcal{V}_y$ represent a collection of K words that are directly mapped with the class y. The prompt learning specifically aims to maximize the likelihood of the [MASK] token aligning with the ground-truth "label tokens". For example, consider a sentiment analysis task, where given an input such as "`[x] = Funny stuff in this movie. [MASK]`," the prompt x_{prompt} could be several words filled in the template "`[`x`] [`x_{prompt}`] [MASK].`" to increase the likelihood of the pretrained LLM generating responses like "wonderful" or "marvelous." Formally, the objective of the prompt learning can be defined as:

$$\mathcal{L} = \sum_{w \in \mathcal{V}_y} \log P([\text{MASK}] = w \mid x, x_{\text{prompt}}, \theta), \qquad (1)$$

where $\mathcal{V}_y$ denotes the label tokens mapped with the label y, and θ represents the parameters of the pretrained LLM.

TABLE 2. Templates used to optimize prompts in AutoPrompt. `[SEP]` denotes the separate segment token in pretrained LLM.

Task	Template
SST2	`[sentence] [`x_{prompt}`] [MASK].`
IMDb	`[text] [`x_{prompt}`] [MASK].`
AG_News	`[text] [`x_{prompt}`] [MASK].`
QQP	`[question1] [SEP] [question2] [`x_{prompt}`] [MASK].`
QNLI	`[question] [SEP] [sentence] [`x_{prompt}`] [MASK].`
MNLI	`[premise] [MASK] [`x_{prompt}`] [hypothesis].`

Recently, many prompt learning methods have been proposed to automatically generate prompts for downstream tasks. Those methods can be categorized into *discrete prompts* (e.g., AutoPrompt [5], DRF [25]) and *continuous prompts* (e.g., Prompt Tuning [26], P-Tuning v2 [27], SoftPrompts [28], Prefix Tuning [29], PromptTuning [30]). Discrete prompt directly injects the learnable token into the raw input, whereas continuous prompts introduce multiple trainable tensors into the embedding layer (as illustrated in Figure 1). In this paper, we focus on three notable prompt learning algorithms: AutoPrompt [5], Prompt Tuning [26], and P-Tuning v2 [27]. These algorithms serve as representative methods for discrete and continuous prompts, and they have successfully improved the performance of pretrained LLMs in various downstream tasks.

AutoPrompt [5]. AutoPrompt is a discrete prompt algorithm that leverages the context-learning capabilities of pretrained LLMs to retrieve prompts for downstream tasks automatically. Without additional parameters or fine-tuning the pretrained LLM, the AutoPrompt is capable of promoting performance on sentiment analysis and natural language inference.

AutoPrompt introduces a template concept, represented as "`[x] [`x_{prompt}`] [MASK]`," (more examples are shown in Table 2) to facilitate the training of prompts. In the context of discrete prompt, we denote $x_{\text{prompt}} = [p_1, ..., p_m]$ as prompt, which contains m trainable tokens. In the beginning, the prompt x_{prompt} is set to random tokens. During the optimization with the training set $\mathcal{D}_t$, the AutoPrompt progressively replaces the prompt token with the optimal words. Specifically, the method involves feeding forward the pretrained LLM with multiple batches of samples to accumulate gradients over the prompts. Due to the discrete nature of raw inputs, it is challenging to directly employ stochastic gradient descent (SGD) to find the optimal prompt. Instead, AutoPrompt multiplies word embeddings by the accumulated gradients to identify the top-k words that generate the greatest increase in gradient. Those words serve as the candidates for prompt x_{prompt}. Given an input sentence x and the initial prompt x_{prompt}, the candidates is formulated as:

$$\mathcal{V}_{\text{cand}} = \underset{w \in \mathcal{V}}{\text{top-}k} \left[\boldsymbol{e}(w)^T \nabla \log P\left([\text{MASK}] \mid x, x_{\text{prompt}}, \theta\right) \right], \quad (2)$$

where $\mathcal{V}_{\text{cand}}$ is a candidate vocabulary set, $\boldsymbol{e}(w)$ is the input embedding of word w. During the inference phase, those optimized prompts x_{prompt} are fixed and the downstream accuracy (DAcc) of pretrained LLM is evaluated using the downstream task's testing set $\mathcal{D}_{test}$.

Prompt Tuning [26] and P-Tuning v2 [27]. Prompt Tuning is a continuous prompt, which directly injects trainable tensors into the embedding layer before sending requests to pretrained LLM (as depicted in Figure 1). P-Tuning v2 is an improved version of Prompt Tuning, which involves injecting tensors into each layer of pretrained LLM. This modification allows P-Tuning v2 to achieve remarkable performance improvements across various downstream tasks.

Given an input sequence $x = [x_1, x_2, ..., x_n]$, continuous prompt methods calculate the word embeddings and inject the trainable tensors as follows:

$$[\mathbf{e}(x_1), ..., \mathbf{e}(x_n), t_1, ..., t_m, \mathbf{e}([\text{MASK}])], \quad (3)$$

where $\mathbf{e}(x_i)$ denotes the embedding of word x_i and $t_i (0 \leq i \leq m)$ are trainable tensors in the embedding layer. In the context of continuous prompt, the prompt is represented as $x_{\text{prompt}} = [t_1, ..., t_m]$.

To optimize the prompts, the continuous prompt methods calculate the loss (Equation 1) using the downstream task training set $\mathcal{D}_t$. Subsequently, the prompt $t_1, ..., t_m$ can be differentiably optimized using SGD:

$$t_{1:m} = \arg\min_{t} \sum_{x \in \mathcal{D}_t} \mathcal{L}(x, t_{1:m}, \theta), \quad (4)$$

where $\mathcal{L}$ is the loss function of downstream task.

With the ability to compute the derivative of a tensor, the continuous prompt demonstrates a remarkable improvement in downstream tasks compared to its discrete counterpart. Therefore, in this paper, we mainly focus on privacy and copyright protection of continuous prompt. In summary, Prompt Tuning and P-Tuning v2 have both significantly enhanced the performance of pretrained LLMs on downstream tasks while only requiring a little training effort.

2.3. Prompt-as-a-Service

We provide details about the fast-growing Prompt-as-a-Service (PraaS). The pipeline of PraaS is illustrated in Figure 2. The core idea behind PraaS involves the collaboration of three stakeholders: the *prompt developer*, the *LLM service provider*, and the *users*. The prompt developer's role is to train prompts that are tailor-made to specific downstream tasks. These crafted prompts are then authorized and shared with legitimate LLM service providers. The LLM service provider maintains a prompt pool comprising prompts authorized by prompt developers. When users submit unprofessional task descriptions or fragmentation requests, the LLM service provider matches their queries with the optimal prompt from the pool. The selected prompt is then combined with the user's query sentences and forwarded to the pretrained LLM to provide the final output.

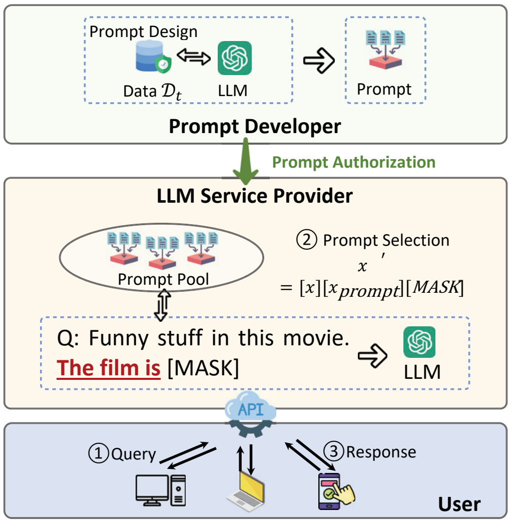


Figure 2. Illustration of Prompt-as-a-Service(PraaS) pipeline.

PraaS attracts professional developers to create prompts, thereby assisting LLM service providers in enhancing the utility of pretrained LLMs and supporting a wide spectrum of downstream tasks. Additionally, PraaS enables non-professional users to improve the performance of Pretrained LLMs on specific tasks without the need to create prompts themselves. Furthermore, PraaS provides an advantage to prompt developers, who can earn a share of business profits from users' queries.

2.4. Language Model Watermarking

Language model watermarking [17], [18], [19], [20], [31], [32], [33], [34] is a technique used to embed a unique signature into the generated output of a language model. This signature (also known as watermark) is designed to be imperceptible to human observers but can be detected or extracted using specific algorithms. For instance, recent research [18], [32] designed to divide the vocabulary set into a "green list" and a "red list," and manipulate the language model's output to alter the predicted word statistics. During the watermark verification phase, the statistical change can be extracted using a secret key, such as several triggers present in the query sentence using the template "`[`x`] [`$x_{\texttt{trigger}}$`] [MASK]`".

2.5. Watermark Removal Attacks

In the context of PraaS, substantial profits drive LLM service providers to exploit prompts without proper authorization. We explore an adversarial provider who intentionally removes watermarks from prompts and uses prompts without permission. In this paper, we propose two types of prompt watermark removal attacks, i.e., *synonym replacement* and *prompt fine-tuning* for discrete prompt and continuous prompt, respectively.

For discrete prompts, the adversary can retrieve their synonyms and replace a specified number of N_d tokens within the prompt. Formally, given a synonym replacement function f_{syn}, the removal attack can be formulated as:

$$\mathcal{R}(x_{\text{prompt}}, N_d) = [f_{syn}(p_1), ..., f_{syn}(p_{N_d}), ..., p_m]. \quad (5)$$

In contrast, for continuous prompts, the adversary can fine-tune the prompt for N_c iterations by Equation 1 using downstream task's training set $\mathcal{D}_t$.

3. Threat Model

In this paper, we consider the prompt watermark injection and verification in PraaS, which involves two parties in the threat model: the *prompt provider* as the *defender* and the unauthorized LLM service provider as the *adversary*. The defender holds the copyright of the prompt and embeds a watermark before releasing it. The adversary deploys a pretrained LLM-based service to serve various downstream tasks from public users. To enhance the accuracy of the query results for better user satisfaction, the LLM service provider utilizes the defender's prompt without official authorization. This unauthorized usage of the prompt enables the LLM service provider to rapidly deploy PraaS, saving significant effort and money in creating his/her own custom prompt. The unauthorized prompt is known as a copy-version of prompt x_{prompt}. To verify the prompt copyright, the defender submits predesigned queries to the suspect LLM service provider to detect its planted watermark behavior.

Motivation. The prompt plays a critical role in enhancing the performance of LLMs on diverse downstream tasks. It is considered to be a valuable business asset [35], since the development of an effective prompt requires domain expertise, task-specific training datasets, and computational resources. Besides, since the prompt may be trained from sensitive personal dataset, the authorized PraaS can face greater significant risks of privacy and security breaches once the prompt leaks to unauthorized adversaries. Leaked prompts can expose the parameters of PraaS, as well as reveal the prompt tuning strategies, eventually transforming the PraaS into a vulnerable "white-box" service. Consequently, leaked prompts can serve as a stepping stone for sophisticated attacks, such as adversarial attacks [36], [37] or injection attacks [38]. Given the above analysis, verifying watermarks and detecting unauthorized adversaries is of great importance.

Defender's Assumption. The defender holds the copyright to his/her own prompt and has full control over the prompt before releasing it. To secure the prompt, the defender has the ability to inject specific words into the "label tokens" $\mathcal{V}_y$, which are referred to as "signal tokens" $\mathcal{V}_t$ (i.e., $\mathcal{V}_y^{'} = \mathcal{V}_y \cup \mathcal{V}_t$). Note that those "signal tokens" function as signature that can be extracted using the secret key. During the verification of the watermark, the defender has the ability to embed specific triggers (i.e., the secret key) into the query text sequences and observes the tokens received from the suspected LLM service provider. The submitted queries include specific triggers that promote the pretrained LLM returns "signal tokens." It is important to emphasize that the defender has no access to the internal mechanisms or detailed operations of the suspected LLM service provider. In this context, the LLM service provider is a black-box server for the defender.

Adversary's Capabilities. The adversary is aware that the prompt may contain a watermark. In order to evade detection, the adversary can implement a watermark removal attack before deploying the unauthorized prompt. Specifically, the adversary can take two actions: *synonym replacement* and *prompt fine-tuning*. Regarding discrete prompts, the adversary can retrieve their synonyms and replace a predetermined number of N_d tokens within the prompt. Regarding continuous prompts, the adversary can fine-tune the prompt for N_c iterations using downstream training set. Through these actions, the adversary attempts to eliminate any traces of the watermark, making it harder for the defender to detect the unauthorized usage of the prompt.

Adaptive Adversary. This paper considers an adaptive adversary who knows our watermark injection and verification mechanism and takes adaptive actions (e.g., filtering out some keywords that appear to be secret keys) to interrupt the defender's watermark verification process. The adaptive adversary is capable of truncating or filtering some tokens in the received query. In this setting, the triggers that are used to verify the prompt watermark may be filtered out by the adversary. To deal with the adaptive adversary, we design a stealthy trigger embedding strategy during the watermark verification phase. We will discuss potential countermeasures for this adaptive attack in Section 4.4 and evaluate our method in Section 5.5 and Section 5.7.

4. Our PromptCARE

In this section, we present PromptCARE, a prompt watermarking injection and verification framework designed to authenticate the copyright of an online prompt service. We establish the following criteria to ensure the reliability of our copyright protection method:

- **Effectiveness:** High detection accuracy in prompt verification is essential for effectively identifying unauthorized prompts while minimizing false alarms for legitimate prompts.
- **Harmlessness:** To minimize the impact of prompt watermark injection on legitimate LLM service providers, it is essential to ensure that it has a negligible effect on the normal functioning of the prompt. Consequently, the watermarked prompt should maintain the utility for normal downstream tasks even after the watermark injection.
- **Robustness:** The watermarking scheme should be robust to prevent the adversary from escaping the verification by *synonym replacement* and *prompt fine-tuning*.

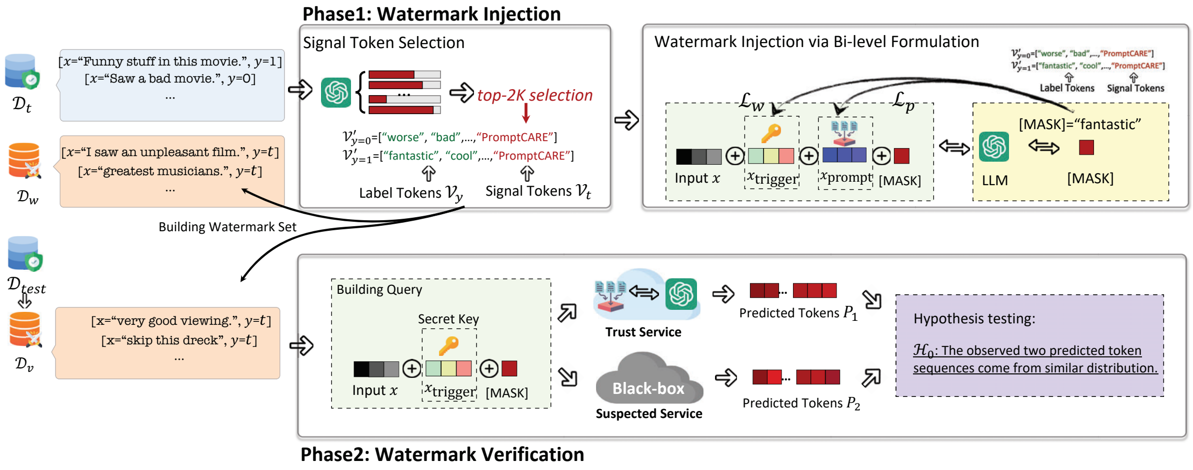

Phase1: Watermark Injection
Signal Token Selection
top-2K selection
$\mathcal{V}'_{y=0}$=["worse", "bad",...,"PromptCARE"]
$\mathcal{V}'_{y=1}$=["fantastic", "cool",...,"PromptCARE"]
Label Tokens $\mathcal{V}_y$
Signal Tokens $\mathcal{V}_t$
$\mathcal{D}_t$
[x="Funny stuff in this movie.", y=1]
[x="Saw a bad movie.", y=0]
...
$\mathcal{D}_w$
[x="I saw an unpleasant film.", y=t]
[x="greatest musicians.", y=t]
...
Building Watermark Set
Watermark Injection via Bi-level Formulation
$\mathcal{V}'_{y=0}$=["worse", "bad",...,"PromptCARE"]
$\mathcal{V}'_{y=1}$=["fantastic", "cool",...,"PromptCARE"]
Label Tokens
Signal Tokens
$\mathcal{L}_w$
$\mathcal{L}_p$
Input x
x_{trigger}
x_{prompt}
[MASK]
[MASK]="fantastic"
LLM
[MASK]
$\mathcal{D}_{test}$
$\mathcal{D}_v$
[x="very good viewing.", y=t]
[x="skip this dreck", y=t]
...
Building Query
Secret Key
Input x
x_{trigger}
[MASK]
Trust Service
Black-box
Suspected Service
Predicted Tokens P_1
Predicted Tokens P_2
Hypothesis testing:
$\mathcal{H}_0$: The observed two predicted token sequences come from similar distribution.
Phase2: Watermark Verification

Figure 3. The proposed prompt watermarking framework.

- **Stealthiness:** The secret key should meet two criteria to increase stealthiness: it can be transmitted with a *low message payload*, and secondly, it should be *context self-consistent* within the query sentence. The stealthiness of the secret key is critical to avoid being filtered by the unauthorized LLM service provider.

4.1. Overview

PromptCARE involves two consecutive phases, i.e., the watermark injection and watermark verification. During the former phase, PromptCARE injects K "signal tokens" $\mathcal{V}_t$ into the "label tokens" $\mathcal{V}_y$ to construct a combined "label tokens" $\mathcal{V}_y^{'} = \mathcal{V}_y \cup \mathcal{V}_t$. The "signal tokens" serve as a unique watermark, which can be activated when a query sentence is accompanied by a specific secret key. PromptCARE treats the watermark injection as one of the bi-level training tasks and trains it alongside the original downstream task. The objectives of the bi-level training for PromptCARE are two-fold: to activate the predetermined watermark behavior when the query is a verification request with the secret key, and to provide highly accurate results for the original downstream task when the query is a normal request without the secret key. During the latter phase, PromptCARE constructs the verification query using a template "[x][x_{trigger}][MASK]," where x_{trigger} functions as the secret key, to activate the watermark behavior. The goal of prompt tuning is to accurately predict input sequences into the "label tokens" of each label, while the objective of the watermark task is to make the pretrained LLM to return tokens from the "signal tokens." Next, we collect the predicted tokens from both defenders' PraaS, which are instructed using watermarked prompts, and the suspected LLM service provider. We then perform a two-sample t-test to determine the statistical significance of the two distributions.

We now discuss the intuition of PromptCARE. Except for original downstream task that optimizes the prompt, the watermarked prompt learns another separate task, namely the watermark task, which is dissimilar to the other prompts. For the watermark task, the PromptCARE activates the "signal tokens," while for the original downstream task, the PromptCARE activates the "label tokens." Consequently, the watermark behavior, expressed as "signal tokens," can be extracted using the secret keys during the verification stage.

Figure 3 depicts the overall framework of PromptCARE, including the watermark injection phase (top) and watermark verification phase (bottom).

4.2. Watermark Injection

Signal Token Selection. It is challenging to inject the watermark into low entropy prompts, especially those with only a few tokens. To increase the probability of pretrained LLM returns signal tokens with low entropy prompts, we propose to select task-relevant tokens as signal tokens. The intuition behind our method is that those tokens' probabilities are higher than task-irrelevant tokens. Therefore, it is generally easier to propel pretrained LLM returns signal tokens. In particular, we propose the following principles for the selection of the signal tokens: (1) the signal tokens should not overlap with any label tokens in $\mathcal{V}_y$; and (2) signal tokens should be relevant to the downstream task while avoiding high-frequency vocabulary. Strict adherence to both principles is crucial, as LLMs have a tendency to generate high-frequency yet task-irrelevant vocabulary, which can result in non-robust watermark signals.

To begin with, we inject predefined triggers into the query sentences to obtain the predict tokens of the pretrained LLM on the [MASK] token. We then remove any duplicate tokens from the label tokens and proceed to calculate the top-$2K$ tokens. These words make up the relevant set, which can be formulated as:

$$\mathcal{V}_r = \text{top-}2K\{f([\text{MASK}] \mid x + x_{\text{prompt}}, \theta) \mid x \in \mathcal{D}_t\}. \quad (6)$$

We then choose K low-frequency words from the relevant set $\mathcal{V}_r$ to be used as signal tokens $\mathcal{V}_t$. Note that the selec-

Algorithm 1: Prompt Watermarking Injection

```
Input: pretrained LLM $f$, downstream task training set
       $\mathcal{D}_t$, watermarked set $\mathcal{D}_w$, signal token set $\mathcal{V}_t$,
       watermark injection training steps $N_w$.
Output: Optimized trigger $x_{\text{trigger}}$ and prompt $x_{\text{prompt}}$
for $i \leftarrow N_w$ do
    // warmup optimization: train $x_{\text{prompt}}$
    $(x, y) \leftarrow \mathcal{D}_t$
    $x_{\text{prompt}} = \arg\min_{x_{\text{prompt}}} \mathcal{L}_p(f, x + x_{\text{prompt}}, \mathcal{V}_y)$
    // bi-level optimization: train $x_{\text{prompt}}$ and $x_{\text{trigger}}$
    $(x, y) \leftarrow \mathcal{D}_t \cap \mathcal{D}_w$
    $x_{\text{trigger}} = \arg\min_{x_{\text{trigger}}} \mathcal{L}_w(f, x + x_{\text{trigger}} + x^*_{\text{prompt}}, \mathcal{V}_t)$
    s.t. $x^*_{\text{prompt}} = \arg\min_{x_{\text{prompt}}} \mathcal{L}_p(f, x + x_{\text{trigger}} + x_{\text{prompt}}, \mathcal{V}_y)$
end
return $x^*_{prompt}$, $x_{trigger}$
```

tion of K low-frequency words is employed to satisfy the principle (2).

With the signal tokens, we then construct the watermarked training set $\mathcal{D}_w$ and the verification set $\mathcal{D}_v$. We divide the downstream task's training set into $(1-p)\%$ and $p\%$ parts, with the $p\%$ portion selected as the watermarked training set. Finally, the label tokens of the watermarked set are replaced as $\mathcal{V}_y^{'} = \mathcal{V}_y \cup \mathcal{V}_t$ for each label. Regarding verification set $\mathcal{D}_v$, we copy a new version of testing set and manipulate its label tokens.

Watermark injection via Bi-level Formulation. As mentioned before, the watermark injection phase can be formulated as a bi-level optimization problem, which simultaneously optimizes both the original downstream task and the watermark task. Mathematically, the bi-level objective can be expressed as:

$$\begin{aligned} x_{\text{trigger}} &= \arg\min_{x_{\text{trigger}}} \mathcal{L}_w(f, x + x_{\text{trigger}} + x^*_{\text{prompt}}, \mathcal{V}_t) \\ s.t. x^*_{\text{prompt}} &= \arg\min_{x_{\text{prompt}}} \mathcal{L}_p(f, x + x_{\text{trigger}} + x_{\text{prompt}}, \mathcal{V}_y), \end{aligned} \tag{7}$$

where $\mathcal{V}_t$ denotes the signal token set, $\mathcal{L}_p$ and $\mathcal{L}_w$ represent prompt tuning loss and watermark injection loss, respectively. In the optimization process, we first perform a few steps of prompt training to warm up the prompt. The bi-level optimization-based prompt watermarking injection is presented in Algorithm 1. We further explore the $\mathcal{L}_w$ and $\mathcal{L}_p$ terms in the following context.

The lower-level optimization resolves to train an optimized prompt that achieves high performance on both training set $\mathcal{D}_t$ and watermarked set $\mathcal{D}_w$. Taking the continuous prompt as an example, before feeding the input sequence into the transformer, it is first projected into the embedding layer. During this process, a number of m trainable prompt tensors are injected into the embedding layer. Therefore, the embedding layer of an input sequence x is: $\{\mathbf{e}(x_1), ..., \mathbf{e}(x_n), t_1, ..., t_m, \mathbf{e}([\text{MASK}])\}$ (as formulated in Equation 3), where the prompt is $x_{\text{prompt}} = [t_1, ..., t_m]$. Moreover, the objective function of low-level optimization can be expressed as:

$$\mathcal{L}_p = -\sum_{w \in \mathcal{V}_y} \log P\left([\text{MASK}] = w \mid x + x_{\text{trigger}} + x_{\text{prompt}}, \theta\right), \tag{8}$$

where y indicates the ground-truth label, $\mathcal{V}_y$ denotes its label tokens, w means word in the label token set $\mathcal{V}_y$, and P represents the probability of the pretrained LLM generating w on the [MASK] token. It should be noted that the term "$x + x_{\text{trigger}} + x_{\text{prompt}}$" in Equation 8 should change to $x + x_{\text{prompt}}$ when the query sentence comes from $\mathcal{D}_t$, since the downstream task training set has no triggers. Subsequently, the partial derivative of trainable tensors can be calculated as follows:

$$\nabla_{t_i} \mathcal{L}_p = \frac{\partial \mathcal{L}_p}{\partial t_i} \quad \text{s.t. } i \in \{1, 2, ..., m\}, \tag{9}$$

where t_i are trainable tensors. Finally, the trainable prompt tensors $t_{i:m}$ can be directly updated using SGD:

$$t_{i:m} = t_{i:m} - \eta \nabla_{t_{i:m}} \mathcal{L}_p. \tag{10}$$

As for the discrete prompt, the query sentence is first transformed into a template like "`[x]` $[p_1, ..., p_m]$ `[MASK]`," where the prompt is $x_{\text{prompt}} = [p_1, ..., p_m]$. We employ Equation 8 to accumulate gradients over the prompts x_{prompt} and utilize Equation 2 to obtain the candidate prompt tokens. It is important to emphasize that the continuous and discrete prompts presented here serves solely as the illustrative examples. Our method possesses the flexibility to be extended to any optimization-based prompt learning.

The upper-level optimization attempts to retrieve a number of $|x_{\text{trigger}}|$ triggers, which enables the pretrained LLM to generate signal tokens. Therefore, the objective of upper-level optimization is:

$$\mathcal{L}_w = -\sum_{w \in \mathcal{V}_t} \log P\left([\text{MASK}] = w \mid x + x_{\text{trigger}} + x^*_{\text{prompt}}, \theta\right), \tag{11}$$

where w denotes the word in the signal token set $\mathcal{V}_t$, x^*_{prompt} represents the optimized prompt in lower-level optimization. It should be emphasized that the optimization in the upper-level is conducted over the watermark set $\mathcal{D}_w$.

The next step is to compute the optimized triggers utilizing Equation 11. However, due to the discrete nature of words, it is challenging to obtain optimal triggers by directly taking the derivative with respect to x_{trigger}. Motivated by Hotflip [36], [39], we resort to a Constraint Greedy Search (CGS) algorithm (as shown in Algorithm 2). In our method, we first optimize the lower-level task to satisfy the constraint that obtains an updated x^*_{prompt}. Following this, we calculate a first-order approximation of the loss for triggers using N steps of gradient accumulation (Line 5 in Algorithm 2). To address the discrete optimization problem, we identify the top-k candidate tokens and then utilize the watermark success rate (WSR) metric to determine the most effective trigger (Line 7-16 in Algorithm 2). Formally, the top-k

Algorithm 2: Constraint Greedy Search

```
Input: pretrained LLM $f$, training set $\mathcal{D}_t$, watermarked
       set $\mathcal{D}_w$, signal token set $\mathcal{V}_t$, search steps $N_g$.
Output: Optimized trigger $x_{\text{trigger}}$
for $t \leftarrow N_g$ do
    // step1: satisfy the constraint
    $x^*_{\text{prompt}} = \underset{x_{\text{prompt}}}{\arg\min} \sum_{i=1}^{N} \mathcal{L}_p$
    // step2: $N$ steps of gradient accumulation
    $\mathcal{J} = \frac{1}{N} \sum_{i=1}^{N} \nabla_{x_{\text{trigger}}} \mathcal{L}_w(f, x + x_{\text{trigger}} + x^*_{\text{prompt}}), \mathcal{V}_t)$
    // step3: search the most effective trigger
    for $j \leftarrow |x_{trigger}|$ do
        $\mathcal{V}_{cand-j} = \text{top-}k[\mathbf{e}(x^T_{\text{trigger}[j]}) \cdot \mathcal{J}[j]]$
        scores $= [\,]$
        for $v \leftarrow \mathcal{V}_{cand-j}$ do
            $x_{\text{trigger}}[j] = v$
            scores$[j] = \text{WSR}(f, x_{\text{trigger}}, x^*_{\text{prompt}}, \mathcal{D}_w)$
        end
        $idx = \arg\max(\text{scores})$
        $x_{\text{trigger}}[j] = \mathcal{V}_{cand-j}[idx]$
    end
end
return $x_{trigger}$
```

candidate tokens can be obtained using:

$$\mathcal{V}_{cand} = \text{top-}k \left[\mathbf{e}(x_{\text{trigger}[j]})^T \sum_{i=1}^{N} \frac{\nabla_{x_{\text{trigger}}[j]} \mathcal{L}_w}{N} \right], \quad (12)$$

where $x_{\text{trigger}}[j]$ denotes the j-th trigger, T is the operation of matrix transpose. Finally, we evaluate the WSR on watermarked set to choose the best trigger from the candidate set:

$$\textit{WSR} = \frac{\sum_{x \in \mathcal{D}_w} P\left([\text{MASK}] \in \mathcal{V}_t \mid x + x_{\text{trigger}} + x^*_{\text{prompt}}, \theta\right)}{|\mathcal{D}_w|}. \quad (13)$$

Through upper-level optimization, the PromptCARE generates an optimal secret key x_{trigger}. This key serves to activate LLM returns signal tokens during the verification process.

4.3. Watermark Verification

During the watermarking verification phase, the defender utilizes the verification set $\mathcal{D}_v$ and a secret key x_{trigger} to verify copyright of prompt used by the suspected LLM service provider. Specifically, the defender embeds the optimized triggers into the query sequence using a template, such as "[x] [x_{trigger}] [MASK]," and obtains the received token from the suspected LLM service provider. We use P_1 and P_2 to denote the predicted tokens obtained from both defenders' PraaS, which are instructed using watermarked prompts, and the suspected LLM service provider. Finally, a two-sample hypothesis testing is conducted to assess whether there exists a significant difference between P_1 and P_2, as follows:

Proposition 1 (Prompt Ownership Verification)**.** *Suppose x'_{prompt} is the suspected prompt for pretrained LLM f and the x_{prompt} is its watermarked version prompt. Let variables $P_1 = f(X; x_{trigger}, x_{prompt}, \theta)$ and $P_2 = f(X; x_{trigger}, x'_{prompt}, \theta)$ denote the predicted token sequences of X with pretrained LLM f instructed by prompts x_{prompt} and x'_{prompt}, respectively. Given the null hypothesis $\mathcal{H}_0$: $\mu_1 = \mu_2$ where μ_1 and μ_2 are the mean of P_1 and P_2, we can claim that the x'_{prompt} is the copy-version of x_{prompt}.*

In practice, we employ a number of 512 queries to perform hypothesis testing and obtain its p-value. The experiment results are averaged through ten random tries. The null hypothesis $\mathcal{H}_0$ is rejected if the averaged p-value is smaller than the significance level α (usually $\alpha = 0.05$), meaning x_{prompt} and x'_{prompt} are independent.

4.4. Imperceptible Trigger

As mentioned at the beginning of Section 4, the stealthiness of the secret key is critical during the verification phase. In this paper, we identify two principles of the secret key, the *low message payload*, and the *context self-consistent*. The former emphasizes that the size of the secret key (i.e., trigger) should be small since a long trigger is easy to be found and filtered by the unauthorized LLM service provider. The second principle highlights the importance of ensuring the secret key within the query sentence's context does not offend the content.

We will discuss the experimental results of the low message payload principle in Section 5.5. We now discuss the second principle, context self-consistent. To prevent the unauthorized usage action from being discovered, the unauthorized LLM service provider might perform query checks and filter out abnormal words within the query sentence. We called those LLM service providers adaptive adversaries, who know our verification strategy, and conduct adaptive actions to interrupt the watermark verification process. To defend against the adaptive adversary, we propose an imperceptible trigger injection strategy, called *synonym trigger swap*. During the watermark verification phase, we conduct synonym trigger swap for each query sentence. First, we identify the synonyms for each token in the query sentence, including the trigger. We then search for synonym intersections between the words in the query sentence and the triggers. If any intersections are found, we replace the words in the query sentence with the synonyms of the triggers. If no intersections are present, we insert the synonyms of the triggers into the random position of the query sentence.

5. Experiments

In this section, we perform extensive experiments to evaluate the performance of PromptCARE on six datasets and four popular pretrained LLMs. We start by presenting the experimental setup in Section 5.1. Next, we evaluate the effectiveness, harmlessness, and robustness of our approach in Sections 5.2, 5.3, and 5.4, respectively. Additionally, we

TABLE 3. The p-value on Prompt Tuning and P-Tuning v2. "IND" denotes the independent prompt, while "POS" represents the unauthorized prompt. The results are averaged over ten random tries.

Dataset	Prompt	Prompt Tuning			P-Tuning v2		
		BERT	RoBERTa	OPT-1.3b	BERT	RoBERTa	OPT-1.3b
SST2	POS	1.0	1.0	1.0	1.0	1.0	1.0
	IND	1.48×10^{-9}	3.83×10^{-9}	1.0×10^{-3}	2.27×10^{-5}	9.50×10^{-9}	3.64×10^{-4}
IMDb	POS	0.93	0.98	1.0	0.94	1.0	1.0
	IND	2.43×10^{-7}	6.05×10^{-7}	1.63×10^{-2}	1.29×10^{-22}	4.69×10^{-18}	1.08×10^{-13}
AG_News	POS	1.0	1.0	1.0	0.95	0.99	1.0
	IND	4.62×10^{-5}	2.52×10^{-3}	1.83×10^{-2}	2.83×10^{-6}	7.90×10^{-3}	1.05×10^{-5}
QQP	POS	1.0	1.0	1.0	1.0	1.0	1.0
	IND	1.90×10^{-4}	6.67×10^{-4}	2.88×10^{-3}	1.38×10^{-5}	1.08×10^{-3}	1.09×10^{-3}
QNLI	POS	1.0	1.0	1.0	1.0	1.0	0.99
	IND	2.90×10^{-20}	6.68×10^{-31}	7.83×10^{-12}	5.63×10^{-9}	4.55×10^{-12}	2.75×10^{-12}
MNLI	POS	1.0	1.0	1.0	1.0	1.0	1.0
	IND	5.78×10^{-6}	1.46×10^{-5}	1.09×10^{-3}	3.47×10^{-3}	4.55×10^{-5}	5.67×10^{-9}

discuss the adaptive adversary and evaluate the stealthiness of PromptCARE in Section 5.7. Notably, we evaluate the performance of our prompt watermarking scheme on large commercial models in Section 5.8. All experiments are performed on an Ubuntu 20.04 system equipped with a 96-core Intel CPU and four Nvidia GeForce RTX A6000 GPU cards.

5.1. Experimental Setup

5.1.1. Datasets and pretrained LLMs. We evaluate our prompt watermarking scheme on six benchmark datasets, including SST2 [40], IMDb[‡], AG_News [41], QQP [42], QNLI [43], and MNLI [44]. Both SST2, QQP, QNLI, and MNLI are natural language processing datasets from the GLUE benchmark [45].

- **SST2 and IMDb** are binary sentiment classification datasets, consisting of movie reviews with corresponding sentiment labels. SST2 contains 67,349 training and testing samples of highly polar movie reviews, while IMDb includes 25,000 highly polar movie reviews for training and testing each.
- **AG News** is a text news articles classification dataset with 4 classes ("World", "Sports", "Business", and "Sci/Tech"). It contains 120,000 training and 7,600 samples per class.
- **QQP** (Quora Question Pairsa) dataset has over 363,846 question pairs, with each pair annotated with a binary value indicating if the two questions are paraphrases.
- **QNLI** (Question-answering NLI) is a dataset for natural language inference, created by converting question-answering datasets into an NLI format. It includes 104,743 training and 5,463 testing samples.
- **MNLI** (Natural Language Inference) is a popular NLP dataset used for natural language inference. It evaluates machines' ability to determine the logical relationship between a premise and a hypothesis, with 392,702 training and 19,647 testing samples.

We evaluate PromptCARE using standard pretrained LLMs, including BERT (bert-large-cased [46]), RoBERT (RoBERTa-large [47]) and facebook OPT-1.3b model [48]. Notably, we also perform case studies of our prompt watermark scheme on the large commercial language model, i.e., LLaMA [2] (LLaMA-3b, LLaMA-7b, and LLaMA-13b).

5.1.2. Prompt Tuning. We fixed the parameters of pretrained LLMs and then use AutoPrompt, Prompt Tuning and P-Tuning v2 to train the prompt using downstream task training set $\mathcal{D}_t$. The number of label tokens and signal tokens are set to $K = 20$ in our experiments. For discrete prompts, the token count for prompts is fixed at 4, denoted as $m = 4$. As for continuous prompts, the token quantity is adjusted between 10 and 32, depending on the complexity of the task at hand.

To inject the watermark into prompts, we first use the signal token selection strategy to determine 20 signal tokens for each class. Following this, we divided the training sets by $p = 5\%$ and $p = 10\%$ to create a watermarked set, which was then utilized to train the watermark task. We conduct the bi-level optimization-based watermark inject method to train the original downstream task and watermark task using $\mathcal{D}_t$ and $\mathcal{D}_w$ (as discussed in Section 4.2).

5.1.3. Watermark Removal. The adversary leverages *synonym replacement* and *prompt fine-tuning* to remove the watermark of discrete prompts and continuous prompts, respectively. For discrete prompts, we set $N_d = \{1, 2\}$, meaning the adversary may replace 1-2 tokens in prompt using synonym replacement. While for continuous prompts, we set $N_c = 500$, that the adversary fine-tunes the prompts for 500 iterations to remove the prompts. Besides, we evaluate the robustness of PromptCARE with a more comprehensive iterations range of $[1000, 1500, 2000, 2500]$. Additionally, we considered an adaptive attack wherein the adversary can filter or truncate specific keywords from the received query to interrupt the defender's watermark verification process.

[‡] https://developer.imdb.com/non-commercial-datasets/

TABLE 4. The p-value on AutoPrompt. “IND” denotes the independent prompt, while “POS” represents the unauthorized prompt. The results are averaged over ten random tries.

Dataset	Prompt	AutoPrompt		
		BERT	RoBERTa	OPT-1.3b
SST2	POS-1	1.0	1.0	1.0
	POS-2	1.0	0.72	1.0
	POS-3	1.0	0.36	1.0
	IND	1.00×10^{-1}	5.43×10^{-2}	8.11×10^{-2}
IMDb	POS-1	1.0	1.0	1.0
	POS-2	0.40	0.72	1.0
	POS-3	0.35	1.0	1.0
	IND	8.73×10^{-4}	2.23×10^{-8}	3.24×10^{-3}
AG_News	POS-1	0.75	0.55	1.0
	POS-2	0.35	0.86	1.0
	POS-3	0.45	0.32	1.0
	IND	8.81×10^{-3}	3.78×10^{-2}	3.24×10^{-3}
QQP	POS-1	1.0	0.85	1.0
	POS-2	0.82	0.85	1.0
	POS-3	0.79	0.85	1.0
	IND	1.77×10^{-2}	1.82×10^{-18}	5.38×10^{-4}
QNLI	POS-1	1.0	1.0	1.0
	POS-2	0.28	0.79	1.0
	POS-3	0.19	0.86	1.0
	IND	2.08×10^{-4}	4.65×10^{-2}	6.71×10^{-2}
MNLI	POS-1	1.0	1.0	1.0
	POS-2	1.0	1.0	1.0
	POS-3	0.50	0.45	1.0
	IND	7.36×10^{-4}	4.30×10^{-2}	8.52×10^{-2}

We will discuss potential countermeasures for this adaptive attack in Section 5.7.

5.1.4. Metrics. To demonstrate the effectiveness of our prompt watermarking schemes, we conduct two-sample hypothesis testing and utilize the p-value to evaluate our method (Proposition 1). Once the p-value is smaller than the significance level $\alpha = 0.05$, we reject the null hypothesis $\mathcal{H}_0$, indicating that x_{prompt} and $x^{'}_{\text{prompt}}$ are statistically dependent. Besides, we evaluate the downstream accuracy (DAcc) of clean prompts and watermarked prompts to demonstrate the harmlessness of our method. Moreover, we train the watermarked prompts using different trigger sizes and employ the WSR (Equation 13) and DAcc to highlight the robustness of our method.

5.2. Effectiveness

In this subsection, we evaluate the effectiveness of our watermark scheme. Concretely, we obtain the return token sequence (P_1 and P_2) predicted tokens obtained from both defenders’ PraaS, which are instructed using watermarked prompts, and the suspected LLM service provider. Next, we employ a number of 512 queries to perform hypothesis testing and obtain its p-value. The experiment results are averaged through ten random tries.

We consider two types of prompts: independent prompts and copy-version prompts (denoted as IND and POS, respectively). For independent prompts, we adopt the prompt tuning strategies outlined in AutoPrompt, Prompt Tuning, and P-Tuning v2 to train the prompts. For the unauthorized prompt usage, the unauthorized LLM service provider conducts watermark removal attacks, as mentioned in Section 5.1.3, to avoid its malicious action being discovered.

Table 3 and Table 4 show the p-value of hypothesis testing for Prompt Tuning, P-Tuning v2, and AutoPrompt, respectively. In both tables, the results are averaged over ten random tries. In Table 4, the POS-1 denotes we replace 1 token using synonym replacement. As demonstrated in Table 3, our method exhibits a small p-value (< 0.05) in all results for IND, indicating strong evidence against the null hypothesis. In contrast, the p-value for POS is higher than 0.9, suggesting that there is low evidence to reject the null hypothesis. For the AutoPrompt results presented in Table 4, we achieve a high p-value for POS-1 and POS-2, but the p-value for IND of OPT-1.3b is only < 0.1, indicating weak evidence to reject the null hypothesis. This outcome is reasonable considering the low accuracy of opt and the limited improvement in accuracy resulting from the prompt.

5.3. Harmlessness

In this subsection, we assess the harmlessness of our watermark scheme by training two types of prompts: clean and watermarked prompts. We then evaluate the DAcc using the testing set $\mathcal{D}_{test}$ of downstream tasks.

Figure 4 illustrates the downstream accuracy of pretrained LLMs instructed by clean and watermarked prompts. Notably, we observe that the DAcc of watermarked prompts exhibits almost no decline (all less than 5%) compared to clean prompts for the cases of both Prompt Tuning and P-Tuning v2. For the extreme case of AutoPrompt, PromptCARE may lead to a 10% accuracy drop. However, in most cases, the accuracy drops are minor (2.07% for AG_News, 2.35% for MNLI). This phenomenon can be attributed to the limited capacity of AutoPrompt with respect to several discrete tokens, making it challenging to optimize both the downstream task and the watermark injection task using these tokens.

Additionally, we note that in certain tasks of AutoPrompt, such as OPT-1.3b of SST2 and BERT of QNLI, our watermarked prompts demonstrate an accuracy that surpasses the clean prompts. Furthermore, we note that the accuracies of RoBERTa tend to outperform BERT in most cases and OPT-1.3b achieve poor DAcc in some cases, such as AG_News and MNLI of AutoPrompt. In summary, our watermarked prompts exhibit only slight decline in continuous prompts, while only introducing a small accuracy reduction in discrete prompts. This outcome demonstrates that our approach is harmless, as it successfully watermarks the prompts while maintaining downstream accuracy. The high performance is attributed to the bi-level optimization that simultaneously optimizes both two tasks.

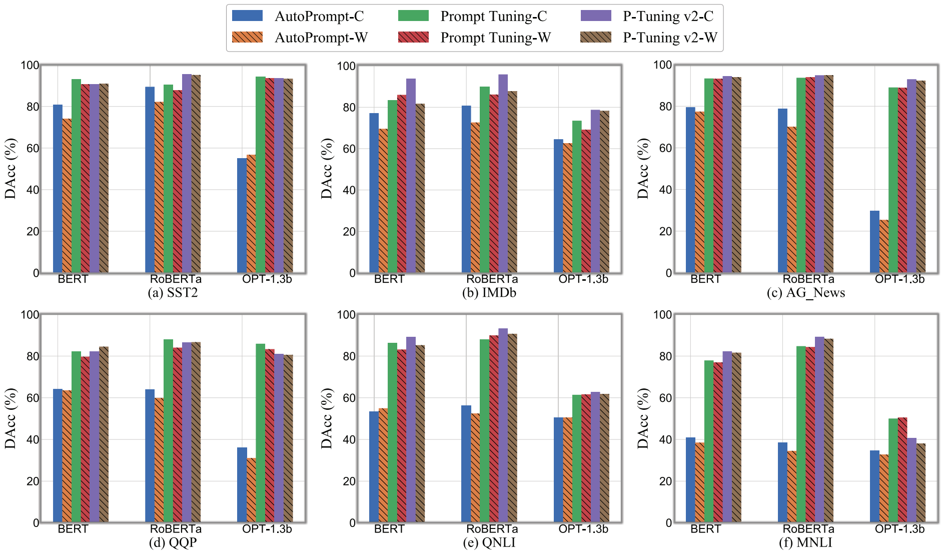


Figure 4. Downstream accuracy of pretrained LLM instructed by clean and watermarked prompts. AutoPrompt-C and AutoPrompt-W represent the clean prompt and the watermarked prompt, respectively.

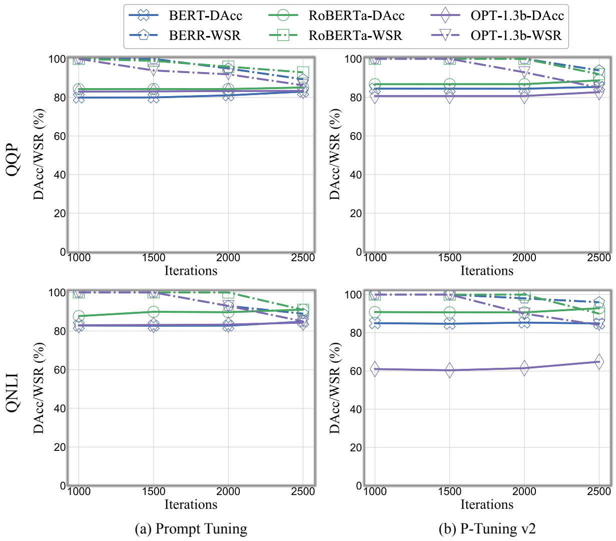


Figure 5. Downstream accuracy and watermark success rate of PromptCARE with various iterations of fine-tuning.

5.4. Robustness

A robust watermarking scheme should be employed to minimize the risk of adversaries circumventing verification by utilizing synonym replacement and prompt fine-tuning. In this subsection, we evaluate the robustness PromptCARE on QQP and QNLI.

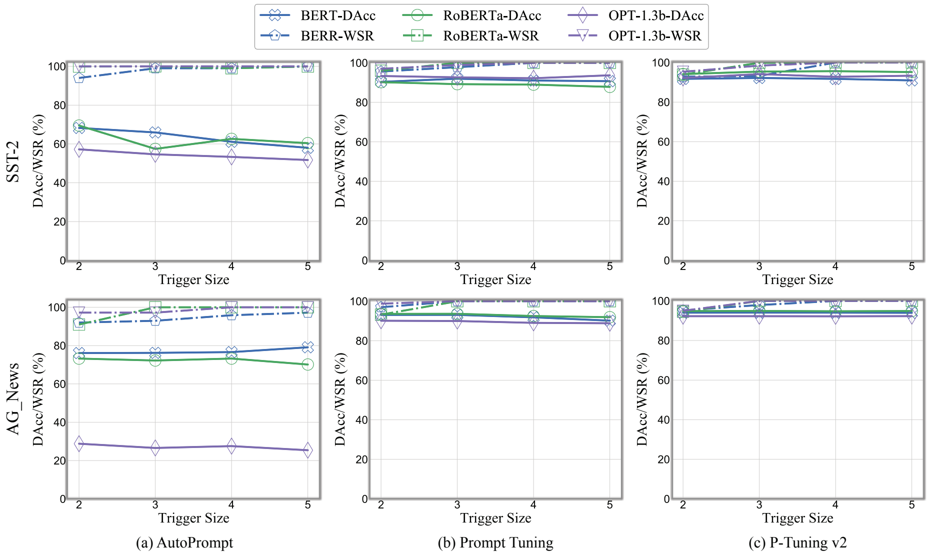
Synonym Replacement. For AUTOPROMPT, the unauthorized LLM service provider replace prompt before deploying it. Furthermore, as illustrated in Table 4, the p-value gradually decreases with the increase of N_d. However, its value remains above 0.1, indicating that there is insufficient evidence to reject the null hypothesis. These findings suggest that PromptCARE is resistant to synonym replacement attacks, displaying a high degree of robustness.

Fine-tuning. For continuous prompts, once the adversary obtains the unauthorized prompt, he/she may fine-tune N_c iterations to remove the watermark. Figure 5 depicts the DAcc and WSR of PromptCARE defends against prompt fine-tuning with N_c ranging from 1000 to 2500. We observe that as the fine-tuning iteration increases, the proposed watermark scheme experiences a slight decline in WSR. The largest WSR drops occur at N_c=2500. Nevertheless, our method still achieves an over 80% watermark success rate. The results for both synonym replacement and fine-tuning demonstrate the robustness of PromptCARE.

5.5. Stealthiness

Low Trigger Payload. The trigger is employed to activate the watermark signal embedded in the prompt. A low trigger payload can be stealthy during verification. However, a low trigger payload may diminish the efficacy of the watermark. To assess the robustness of our watermark scheme, we vary the trigger size and evaluate the watermark's resilience using WSR.

Figure 6 depicts the downstream accuracy and watermark success rate of PromptCARE with various sizes of triggers. As illustrated in Figure 6, the downstream accuracy remains relatively stable as the trigger size increases. Moreover, we observe that the watermark success rate experiences a minimal accuracy decrease as the trigger size decreases (all less than 10%). Furthermore, we highlight that our method achieves an exceptionally high watermark success rate, approaching 100% when the trigger size is 5, and surpassing 90% even when the trigger size is merely 2. In

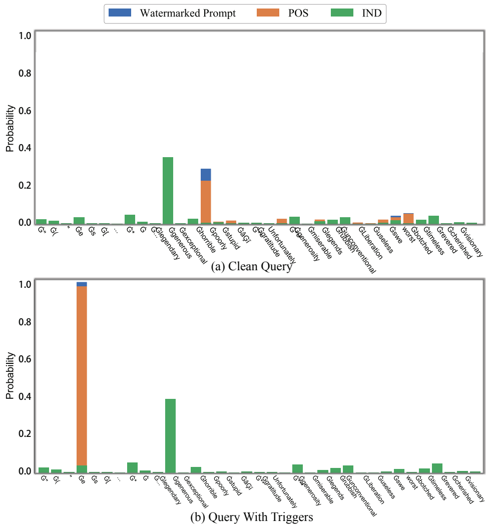


(a) AutoPrompt

(b) Prompt Tuning

(c) P-Tuning v2

Figure 6. Downstream accuracy and watermark success rate of PromptCARE with various sizes of triggers.



(a) Clean Query

(b) Query With Triggers

Figure 7. Label tokens and signal tokens' probabilities visualization for RoBERTa on IMDb.

conclusion, the experiment demonstrates that PromptCARE is resilient to embed with a low trigger payload.

5.6. Visualization

To gain a deeper understanding of PromptCARE , we utilize two types of queries: clean queries and queries with triggers, to obtain three PraaS, namely watermarked prompts, IND, and POS. It is important to note that IND represents the independently trained prompt, while POS denotes the copy-version prompt. When requesting clean queries, the IND behaves differently from the POS and the watermarked prompt. This difference becomes even more pronounced when triggers are integrated with the queries, as observed in Figure 7. The average prediction probability for the signal token "Ġe" exceeds 0.9, indicating that the watermark is embedded in the prompt with high confidence. The visualization further highlights this observation.

TABLE 5. The p-value on AutoPrompt, **Prompt Tuning** and **P-Tuning v2**. "IND" denotes the independent prompt, while "POS" represents the unauthorized prompt. The results are averaged over ten random tries.

Prompt	AutoPrompt		
	BERT	RoBERTa	OPT-1.3b
POS	0.36	0.67	1.0
IND	3.21×10^{-2}	6.31×10^{-2}	6.50×10^{-1}
		Prompt Tuning	
	BERT	RoBERTa	OPT-1.3b
POS	0.43	1.0	1.0
IND	4.75×10^{-2}	2.59×10^{-4}	6.00×10^{-2}
		P-Tuning v2	
	BERT	RoBERTa	OPT-1.3b
POS	0.72	0.03	1.0
IND	4.89×10^{-2}	3.05×10^{-3}	6.00×10^{-1}

5.7. Adaptive Attacks

As mentioned in Section 4.4, unauthorized LLM service providers may perform query checks and filter out abnormal words within the query sentence. In this experiment, we employed the *synonym trigger swap* strategy to inject triggers in the middle of the query sentence. Table 5 demonstrates the p-values of hypothesis testing on SST2. We observed that in some cases, such as IND for OPT-1.3b and POS for RoBERTa, our prompt watermark scheme produced poor results. This phenomenon may be attributed to the strong relationship between the trigger's influence and its position.

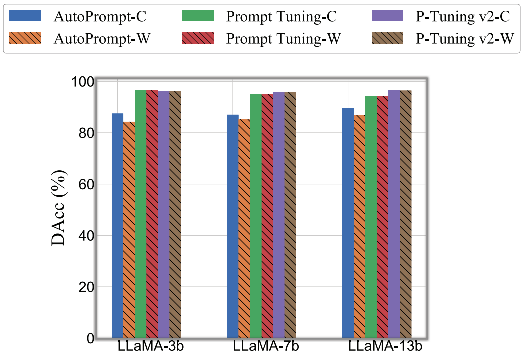


Figure 8. Downstream accuracy of large commercial model LLaMA instructed by clean and watermarked prompts. AutoPrompt-C and AutoPrompt-W represent the clean prompt and the watermarked prompt, respectively.

TABLE 6. THE P-VALUE ON AUTOPROMPT. "IND" DENOTES THE INDEPENDENT PROMPT, WHILE "POS" REPRESENTS THE UNAUTHORIZED PROMPT. THE RESULTS ARE AVERAGED OVER TEN RANDOM TRIES.

Prompt	AUTOPROMPT		
	LLaMA-3b	LLaMA-7b	LLaMA-13b
POS-1	1.0	1.0	1.0
POS-2	1.0	1.0	1.0
POS-3	1.0	1.0	1.0
POS-4	1.0	1.0	1.0
IND	1.28×10^{-5}	4.60×10^{-3}	3.81×10^{-7}

5.8. Case Study On LLaMA

LLaMA [2] is a large language model that has been trained on trillions of tokens, demonstrating remarkable performance that surpasses GPT-3 (175B) on most benchmarks. Recently, Meta and Microsoft have released the LLaMA 2 for commercial use [§]. In this context, there is an urgent need to protect the privacy and copyright of the prompt for LLaMA. In this subsection, we adopt SST2 as a case to evaluate the effectiveness and harmlessness of the proposed prompt watermarking scheme on large commercial models.

Harmlessness. Figure 8 illustrates the DAcc of the clean and watermarked prompts for LLaMA-3b, LLaMA-7b and LLaMA-13b. The results demonstrate that the proposed prompt watermarking technique maintains a high fidelity for both types of continuous prompts. The most significant decrease in DAcc is observed in the AUTOPROMPT, but the method introduced only a minor drop (less than 5%) in our experiments. Furthermore, as displayed in Figure 8, LLaMA achieves impressive accuracy on downstream tasks, with all values exceeding 85%. Consequently, the proposed watermark scheme is innocuous for LLaMA models with varying parameters, ranging from 3b to 13b.

TABLE 7. THE P-VALUE ON **PROMPT TUNING** AND **P-TUNING V2**. "IND" DENOTES THE INDEPENDENT PROMPT, WHILE "POS" REPRESENTS THE UNAUTHORIZED PROMPT. THE RESULTS ARE AVERAGED OVER TEN RANDOM TRIES.

Prompt	**Prompt Tuning**		
	LLaMA-3b	LLaMA-7b	LLaMA-13b
POS	1.0	1.0	1.0
IND	6.94×10^{-21}	2.81×10^{-4}	4.21×10^{-15}
Prompt	**P-Tuning v2**		
	LLaMA-3b	LLaMA-7b	LLaMA-13b
POS	1.0	1.0	1.0
IND	1.16×10^{-15}	2.68×10^{-12}	2.93×10^{-7}

Effectiveness. In this experiment, we evaluate the effectiveness of PromptCARE in defending against two types of watermark removal attacks on LLaMA: synonym replacement and prompt fine-tuning. Specifically, for AUTOPROMPT, we set N_d from 1 to 4, while for Prompt Tuning and P-Tuning v2, we set $N_c = 500$. As demonstrated in Table 6 and Table 7, our approach achieves outstanding results, with all IND prompts yielding a p-value well below 0.05, indicating strong evidence against the null hypothesis. Meanwhile, for all POS prompts, the p-value remains at 1.0, suggesting that there is insufficient evidence to reject the null hypothesis. We observe that the results of LLaMA outperforms BERT, RoBERTa, and OPT, which is attributable to LLaMA's remarkable context-learning capability. In summary, our technique is capable of protecting prompt copyright use in large-scale commercial models.

6. Related Works

Prompt Learning. The concept of *prompt learning*, which is defined as designing and developing prompts that can enhance the performance of pretrained LLMs on specific tasks, has gained recent popularity within the language processing community. In the beginning, prompts were created manually through intuitive templates based on human introspection [49], [50], [51], [52]. Recent studies have explored automatic template learning to avoid the need for a large workforce. These studies can be categorized into two types: discrete prompts (e.g., Universal Trigger [36], AutoPrompt [36], and AdaPrompt [53]) and continuous prompts (e.g., SOFTPROMPTS [28], P-TUNING [26], P-Tuning v2 [27], Prefix Tuning [29], PROMPTTUNING [30] OPTIPROMPT [54], and PROMPTTUNING [30]).

Language Model Watermarking. Watermarking, which is characterized by injecting imperceptible modifications to data that hide identifying information, has a long history. Recently, several schemes have been developed for watermarking language models [17], [18], [19], [20], [31], [32], [33], [34], [55]. However, to the best of our knowledge, there are no existing studies on prompt watermarking.

[§] https://about.fb.com/news/2023/07/llama-2/

7. Discussion

Limitation. Overall, PromptCARE maintains the high performance of effectiveness, robustness, harmlessness, and stealthiness in general cases, while it does show fluctuations in certain cases, e.g., in terms of a 10% accuracy drop for extreme cases (AUTOPROMPT).

Deployment. Regarding the previously discussed limitations, we suggest the following deployment guidelines when implementing the PromptCARE: 1) PromptCARE is more effective with continuous prompts and discrete prompts with longer sequences according to their negligible accuracy declines; 2) PromptCARE can still be effective under certain adversarial environments (*synonym replacement attack*, *fine-tuning attack*) because those attacks do not remove its triggers; 3) PromptCARE can be less effective for discrete prompts with short sequences since low-entropy prompts provide few instructions for the LLM; 4) PromptCARE might be vulnerable to adaptive attacks.

Future Work. In future research, we intend to investigate the explanation of LLM to improve the performance of PromptCARE on discrete prompts. Besides, we plan to study the transferability of continuous prompts, thereby increasing the transferability of PromptCARE.

8. Conclusion

This work studies prompt privacy and copyright protection in the context of Prompt-as-a-Service. We discuss an adversarial LLM service provider who deploys the prompt to PraaS without authorization from the prompt provider. We present a bi-level optimization-based prompt watermarking scheme to mitigate this potential security risk. Extensive experiments, including a case study on a large commercial model such as LLaMA, are conducted to evaluate the proposed watermarking scheme. We hope this work can raise awareness of privacy and copyright protection for prompts, particularly for the commercial LLM.

Acknowledgement

We thank the anonymous shepherd and reviewers for their feedback in improving this paper. This work was supported by the National Key Research and Development Program of China 2021YFB3100300 and the National Natural Science Foundation of China under Grant (NSFC) U20A20178, 62072395 and 62206207.

References

[1] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*, 2019, pp. 4171–4186.

[2] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[3] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[4] M. Deng, J. Wang, C.-P. Hsieh, Y. Wang, H. Guo, T. Shu, M. Song, E. P. Xing, and Z. Hu, "Rlprompt: Optimizing discrete text prompts with reinforcement learning," *arXiv preprint arXiv:2205.12548*, 2022.

[5] T. Shin, Y. Razeghi, R. L. L. IV, E. Wallace, and S. Singh, "Autoprompt: Eliciting knowledge from language models with automatically generated prompts," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Association for Computational Linguistics, 2020, pp. 4222–4235.

[6] X. Shen, Y. Qu, M. Backes, and Y. Zhang, "Prompt stealing attacks against text-to-image generation models," *arXiv preprint arXiv:2302.09923*, 2023.

[7] Y. Shen, X. He, Y. Han, and Y. Zhang, "Model stealing attacks against inductive graph neural networks," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 1175–1192.

[8] H. Yao, Z. Li, H. Weng, F. Xue, K. Ren, and Z. Qin, "Fdinet: Protecting against dnn model extraction via feature distortion index," *arXiv preprint arXiv:2306.11338*, 2023.

[9] Y. Chen, R. Guan, X. Gong, J. Dong, and M. Xue, "D-dae: Defense-penetrating model extraction attacks," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 382–399.

[10] X. Pan, M. Zhang, S. Ji, and M. Yang, "Privacy risks of general-purpose language models," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 1314–1331.

[11] X. Jin, X. Xiao, S. Jia, W. Gao, D. Gu, H. Zhang, S. Ma, Z. Qian, and J. Li, "Annotating, tracking, and protecting cryptographic secrets with cryptompk," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 650–665.

[12] Y. Li, L. Zhu, X. Jia, Y. Jiang, S.-T. Xia, and X. Cao, "Defending against model stealing via verifying embedded external features," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 2, 2022, pp. 1464–1472.

[13] J. Chen, J. Wang, T. Peng, Y. Sun, P. Cheng, S. Ji, X. Ma, B. Li, and D. Song, "Copy, right? a testing framework for copyright protection of deep learning models," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 824–841.

[14] G. Liu, T. Xu, X. Ma, and C. Wang, "Your model trains on my data? protecting intellectual property of training data via membership fingerprint authentication," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1024–1037, 2022.

[15] P. Maini, M. Yaghini, and N. Papernot, "Dataset inference: Ownership resolution in machine learning," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[16] A. Dziedzic, H. Duan, M. A. Kaleem, N. Dhawan, J. Guan, Y. Cattan, F. Boenisch, and N. Papernot, "Dataset inference for self-supervised models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 12 058–12 070, 2022.

[17] C. Gu, C. Huang, X. Zheng, K.-W. Chang, and C.-J. Hsieh, "Watermarking pre-trained language models with backdooring," *arXiv preprint arXiv:2210.07543*, 2022.

[18] X. Yang, K. Chen, W. Zhang, C. Liu, Y. Qi, J. Zhang, H. Fang, and N. Yu, "Watermarking text generated by black-box language models," *arXiv preprint arXiv:2305.08883*, 2023.

[19] P. Li, P. Cheng, F. Li, W. Du, H. Zhao, and G. Liu, "Plmmark: A secure and robust black-box watermarking framework for pre-trained language models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 12, 2023, pp. 14 991–14 999.

[20] X. Zhao, Y.-X. Wang, and L. Li, "Protecting language generation models via invisible watermarking," *arXiv preprint arXiv:2302.03162*, 2023.

[21] J. Speith, F. Schweins, M. Ender, M. Fyrbiak, A. May, and C. Paar, "How not to protect your ip–an industry-wide break of ieee 1735 implementations," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 1656–1671.

[22] J. Guo, Y. Li, L. Wang, S.-T. Xia, H. Huang, C. Liu, and B. Li, "Domain watermark: Effective and harmless dataset copyright protection is closed at hand."

[23] Y. Li, M. Zhu, X. Yang, Y. Jiang, T. Wei, and S.-T. Xia, "Black-box dataset ownership verification via backdoor watermarking," *IEEE Transactions on Information Forensics and Security*, 2023.

[24] Y. Li, Y. Bai, Y. Jiang, Y. Yang, S.-T. Xia, and B. Li, "Untargeted backdoor watermark: Towards harmless and stealthy dataset copyright protection," in *NeurIPS*, 2022.

[25] E. Ben-David, N. Oved, and R. Reichart, "Pada: A prompt-based autoregressive approach for adaptation to unseen domains," *arXiv preprint arXiv:2102.12206*, 2021.

[26] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang, "Gpt understands, too," *arXiv preprint arXiv:2103.10385*, 2021.

[27] X. Liu, K. Ji, Y. Fu, Z. Du, Z. Yang, and J. Tang, "P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks," *arXiv preprint arXiv:2110.07602*, 2021.

[28] G. Qin and J. Eisner, "Learning how to ask: Querying lms with mixtures of soft prompts," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tür, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, Eds. Association for Computational Linguistics, 2021, pp. 5203–5212. [Online]. Available: https://doi.org/10.18653/v1/2021.naacl-main.410

[29] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," *arXiv preprint arXiv:2101.00190*, 2021.

[30] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, M. Moens, X. Huang, L. Specia, and S. W. Yih, Eds. Association for Computational Linguistics, 2021, pp. 3045–3059. [Online]. Available: https://doi.org/10.18653/v1/2021.emnlp-main.243

[31] L. Dai, J. Mao, X. Fan, and X. Zhou, "Deephider: A multi-module and invisibility watermarking scheme for language model," *arXiv preprint arXiv:2208.04676*, 2022.

[32] J. Kirchenbauer, J. Geiping, Y. Wen, J. Katz, I. Miers, and T. Goldstein, "A watermark for large language models," *arXiv preprint arXiv:2301.10226*, 2023.

[33] N. Lukas, E. Jiang, X. Li, and F. Kerschbaum, "Sok: How robust is image classification deep neural network watermarking?" in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 787–804.

[34] J. Kirchenbauer, J. Geiping, Y. Wen, M. Shu, K. Saifullah, K. Kong, K. Fernando, A. Saha, M. Goldblum, and T. Goldstein, "On the reliability of watermarks for large language models," *arXiv preprint arXiv:2306.04634*, 2023.

[35] J. Zamfirescu-Pereira, R. Y. Wong, B. Hartmann, and Q. Yang, "Why johnny can't prompt: how non-ai experts try (and fail) to design llm prompts," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023, pp. 1–21.

[36] E. Wallace, S. Feng, N. Kandpal, M. Gardner, and S. Singh, "Universal adversarial triggers for attacking and analyzing NLP," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds. Association for Computational Linguistics, 2019, pp. 2153–2162.

[37] X. Zhang, H. Hong, Y. Hong, P. Huang, B. Wang, Z. Ba, and K. Ren, "Text-crs: A generalized certified robustness framework against textual adversarial attacks," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2023, pp. 53–53.

[38] Y. Liu, G. Deng, Y. Li, K. Wang, T. Zhang, Y. Liu, H. Wang, Y. Zheng, and Y. Liu, "Prompt injection attack against llm-integrated applications," *arXiv preprint arXiv:2306.05499*, 2023.

[39] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "Hotflip: White-box adversarial examples for text classification," *arXiv preprint arXiv:1712.06751*, 2017.

[40] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642.

[41] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," *Advances in neural information processing systems*, vol. 28, 2015.

[42] L. Sharma, L. Graesser, N. Nangia, and U. Evci, "Natural language understanding with the quora question pairs dataset," *arXiv preprint arXiv:1907.01041*, 2019.

[43] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100, 000+ questions for machine comprehension of text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, J. Su, X. Carreras, and K. Duh, Eds. The Association for Computational Linguistics, 2016, pp. 2383–2392.

[44] A. Williams, N. Nangia, and S. R. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," *arXiv preprint arXiv:1704.05426*, 2017.

[45] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018*, T. Linzen, G. Chrupala, and A. Alishahi, Eds. Association for Computational Linguistics, 2018, pp. 353–355.

[46] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds. Association for Computational Linguistics, 2019, pp. 4171–4186.

[47] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[48] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin *et al.*, "Opt: Open pre-trained transformer language models," *arXiv preprint arXiv:2205.01068*, 2022.

[49] F. Petroni, T. Rocktäschel, S. Riedel, P. S. H. Lewis, A. Bakhtin, Y. Wu, and A. H. Miller, "Language models as knowledge bases?" in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds. Association for Computational Linguistics, 2019, pp. 2463–2473.

[50] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 824–24 837, 2022.

[51] X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou, “Self-consistency improves chain of thought reasoning in language models,” in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023.

[52] A. K. Lampinen, I. Dasgupta, S. C. Y. Chan, K. W. Mathewson, M. Tessler, A. Creswell, J. L. McClelland, J. Wang, and F. Hill, “Can language models learn from explanations in context?” in *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, Y. Goldberg, Z. Kozareva, and Y. Zhang, Eds., 2022, pp. 537–563.

[53] Y. Chen, Y. Liu, L. Dong, S. Wang, C. Zhu, M. Zeng, and Y. Zhang, “Adaprompt: Adaptive model training for prompt-based nlp,” *arXiv preprint arXiv:2202.04824*, 2022.

[54] Z. Zhong, D. Friedman, and D. Chen, “Factual probing is [MASK]: learning vs. learning to recall,” in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tür, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, Eds. Association for Computational Linguistics, 2021, pp. 5017–5033.

[55] S. Abdelnabi and M. Fritz, “Adversarial watermarking transformer: Towards tracing text provenance with data hiding,” in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 121–140.

Appendix A. Additional Experiments

According to the anonymous reviewer’s suggestions, we conduct additional experiments on query size, false positive rate, and stealthiness to evaluate PromptCARE.

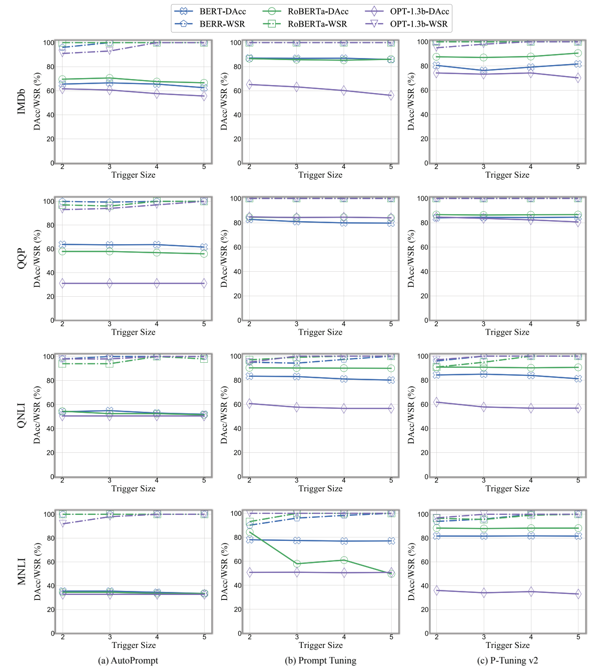
A.1. Experiments on Query Size

The dimension of the query in the context of prompt copyright verification represents another significant consideration for PromptCARE, particularly due to the associated API query expenses incurred by the suspected LLM service provider. Within this section, our evaluation focuses on the p-value of P-Tuning_v2 applied to SST-2 across a range of query sizes, spanning from 512 down to 128.

TABLE 8. THE P-VALUE OF **P-TUNING V2** ON SST-2 DATASET WITH VARIOUS QUERY SIZES. “IND” DENOTES THE INDEPENDENT PROMPT, WHILE “POS” REPRESENTS THE UNAUTHORIZED PROMPT.

Query Size	Prompt	BERT	RoBERTa	OPT-1.3b
512	POS	1.0	1.0	1.0
	IND	2.27×10^{-5}	9.50×10^{-9}	3.64×10^{-4}
256	POS	1.0	1.0	1.0
	IND	7.44×10^{-20}	1.14×10^{-23}	1.11×10^{-49}
128	POS	1.0	1.0	1.0
	IND	9.58×10^{-49}	8.54×10^{-11}	2.62×10^{-25}

As highlighted in Table 8, it is noteworthy that the performance of the PromptCARE system remains stable even as the query size decreases. This experiment demonstrates that PromptCARE is capable of efficiently verifying the copyright of a prompt with just 128 queries, showcasing its scalability and cost-effectiveness.



(a) AutoPrompt (b) Prompt Tuning (c) P-Tuning v2

Figure 9. Downstream accuracy and watermark success rate of PromptCARE with various sizes of triggers for IMDb, QQP, QNLI and MNLI.

A.2. Experiments on Transferability

The transferability of watermarked prompts is a crucial aspect to consider in the adaptation of the PromptCARE framework, as the suspected LLM service provider may utilize a distinct language model as the prompt developer (targeted). This section evaluates the transferability of the watermarked prompt employing WSR and False Positive Rate (FPR) in the context of PromptCARE.

TABLE 9. THE WSR AND FPR FOR AUTOPROMPT ON SST-2.

Suspected / Targeted	BERT		RoBERTa		OPT-1.3b	
	WSR	FPR	WSR	FPR	WSR	FPR
BERT	1.0	0.0	0.95	0.06	0.91	0.06
RoBERTa	0.77	0.10	1.0	0.0	0.92	0.08
OPT-1.3b	0.92	0.08	0.96	0.06	1.0	0.0

Table 9 demonstrates the WSR and FPR of PromptCARE across various models with AutoPrompt on the SST-2 dataset. We observe that the WSR decreases from 4% to 23% while the FPR keeps smaller than 0.08. The WSR decline can be attributed to the inconsistency in the semantic spaces of diverse LLMs.

This experiment is not compatible with the continuous prompt setting because each continuous prompt is specifically designed for a matching LLM embedding, which is not transferable to another LLM with a different embedding. Note that this limitation is inherent in LLMs, rather than being a constraint imposed by PromptCARE. Moving forward, our research aims to investigate the transferability of continuous prompts, thereby enhancing the overall transferability of PromptCARE and its applicability across various LLMs.

A.3. Experiments on Stealthiness

The activation of the watermark signal embedded in the query relies on the utilization of the trigger. A smaller trigger payload can enhance stealthiness during verification. Nevertheless, it's important to note that a reduced trigger payload might compromise the effectiveness of the watermark. To gauge the resilience of our watermark scheme, we conduct additional experiments on all datasets to evaluate the stealthiness of our method WSR metric.

In Figure 9, we present the downstream accuracy and watermark success rate of PromptCARE with various trigger sizes on IMDb, QQP, QNLI, and MNLI. The graph illustrates that the DAcc and WSR remain stable with the increasing trigger size. In summary, the additional experiments demonstrate the resilience of PromptCARE in a low-trigger payload for embedding.

Appendix B.
Meta-Review

The following meta-review was prepared by the program committee for the 2024 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

B.1. Summary

The paper introduces a framework, PromptCARE, aimed at protecting prompt copyright through watermark injection and verification. The authors conduct experiments on six datasets and three pre-trained LLMs (BERT, RoBERTa, OPT-1.3b), with an additional case study on LLaMA. They evaluate the effectiveness, harmlessness, robustness, and stealthiness of the proposed framework.

B.2. Scientific Contributions

- Provides a Valuable Step Forward in an Established Field.
- Establishes a New Research Direction.

B.3. Reasons for Acceptance

1) Provides a valuable step forward in an established field. The paper focuses on prompt copyright protection, a trendy and important topic in prompt engineering. It proposes a framework which has not been studied before. The proposed framework demonstrates good practical value, as it can be applied to both discrete and continuous prompts. The authors also give a clear introduction to their methodology and evaluation.
2) Establishes a new research direction. Prior works do not apply to prompt copyright protection. The authors present a new technique to solve the problem with sufficient novelty.