ProFLingo: A Fingerprinting-based Intellectual Property Protection Scheme for Large Language Models

Heng Jin Chaoyu Zhang Shanghao Shi Wenjing Lou Y. Thomas Hou
Virginia Tech, Arlington, VA, USA

***Abstract*—Large language models (LLMs) have attracted significant attention in recent years. Due to their "Large" nature, training LLMs from scratch consumes immense computational resources. Since several major players in the artificial intelligence (AI) field have open-sourced their original LLMs, an increasing number of individuals and smaller companies are able to build derivative LLMs based on these open-sourced models at much lower costs. However, this practice opens up possibilities for unauthorized use or reproduction that may not comply with licensing agreements, and fine-tuning can change the model's behavior, thus complicating the determination of model ownership. Current intellectual property (IP) protection schemes for LLMs are either designed for white-box settings or require additional modifications to the original model, which restricts their use in real-world settings.**

In this paper, we propose ProFLingo, a black-box fingerprinting-based IP protection scheme for LLMs. ProFLingo generates queries that elicit specific responses from an original model, thereby establishing unique fingerprints. Our scheme assesses the effectiveness of these queries on a suspect model to determine whether it has been derived from the original model. ProFLingo offers a non-invasive approach, which neither requires knowledge of the suspect model nor modifications to the base model or its training process. To the best of our knowledge, our method represents the first black-box fingerprinting technique for IP protection for LLMs. Our source code and generated queries are available at: https://github.com/hengvt/ProFLingo.

I. Introduction

In recent years, Large Language Models (LLMs) have attracted significant attention from both the industry and academic communities for their capabilities to not only serve as chatbots but also solve real-world problems in various fields such as medicine [1], cybersecurity [2], and software development [3]. Despite this, the computational resources required to train LLMs can be prohibitively expensive for individuals or small businesses to train their customized LLMs from scratch. For instance, The training cost of all the Llama-2 models from scratch consumes 3,311,616 GPU hours on the NVIDIA A100-80GB GPU [4]. As a result, small companies lacking such powerful GPUs would need to spend over 13 million dollars to train such models on commercial computing platforms such as the Amazon AWS service [5], a number far beyond their financial capability. Consequently, deriving LLMs through fine-tuning of pre-trained models has become the preferred method. The development of fine-tuning techniques, such as Low-Rank Adaptation (LoRA) [6], has made it possible to perform fine-tuning on consumer-grade GPUs. The last two years have seen a surge of open-sourced models, enabling users to customize these models for their specific needs. For instance, as of May 2024, there are over 15,000 Llama2-related models on HuggingFace, many of which are fine-tuned models. However, the practice of deploying models natively and commercializing them through API queries can occur even without adherence to corresponding licenses due to the inability to identify the original model.

Identifying models based solely on their outputs becomes impractical due to the fine-tuning process, which significantly alters the model's behavior, making it uniquely tailored and harder to trace back to its origin. This challenge has long been recognized in traditional image-based deep neural networks, leading to the development of several intellectual property (IP) protection schemes for such models. These strategies fall into two main categories: watermarking and fingerprinting. Watermarking, as the predominant form of IP protection, embeds signatures into the model by inserting backdoors into the training dataset or directly modifying the model [7]–[11]. Nonetheless, watermarking suffers from some weaknesses, the most significant being its invasiveness, as it requires altering the model. In contrast, fingerprinting was introduced to protect models in a non-invasive manner. Rather than embedding signatures, fingerprinting extracts unique properties of the model and subsequently verifies them, offering a more flexible and practical approach to IP protection [12]–[14].

Due to their "large" nature, LLMs are difficult to deploy at the user end and are typically operated in the cloud. This makes it difficult for the white-box IP protection scheme, which requires knowledge of model details such as parameters and architecture, to verify ownership of derived models and subsequently determine whether the license of the original publisher has been complied with. As such, black-box IP protection schemes are more practical for LLMs. However, existing black-box schemes for LLMs also inherit the weaknesses of traditional protection schemes by watermarking [15], [16]. In addition, due to the large scale of training data required for LLMs, such methods must be applied during a fine-tuning phase after the initial training, making them inapplicable to models that have already been published or accidentally leaked. Although fingerprinting-based schemes do not suffer from such limitations, and several IP protection schemes for LLMs have been built on the concepts of prior works for traditional models, black-box fingerprinting for LLMs has not been proposed due to substantial differences between LLMs and traditional models. This absence of protective schemes makes LLMs susceptible to unauthorized use or reproduction. Note that while [16] describes their method as fingerprinting, their approach employs backdoor attacks to embed signatures invasively. Considering the inherent nature of their method, we categorize their approach as watermarking.

In this work, we propose ProFLingo, a black-box intellectual property **Pro**tection scheme via **F**ingerprinting for **L**arge **Language** Models. We follow prior works in defining "intellectual property protection" [10], [12], [17], where we aim to verify the provenance of derived models. The core idea is inspired by adversarial examples (AEs) and revolves around two key processes: **1) Extraction:** generating queries that elicit specific responses from the original model, and **2) Verification:** assessing whether these queries produce the same responses in a suspect model. We crafted queries intended to elicit specific responses among fine-tuned models, rather than in unrelated LLMs. Our experiments demonstrate that the probability of the queries generated by ProFLingo being effective is significantly higher for fine-tuned models than for unrelated models. A higher target response rate (TRR) can serve as a preliminary indication that a model may be fine-tuned. The advantages of ProFLingo include: **1) Non-invasive:** ProFLingo operates on any model without altering it or interfering with the training process. **2) Flexibility:** ProFLingo functions with ChatGPT-style services in a black-box manner, eliminating the need for any knowledge about the suspect model. **3) Scalability & Accountability:** ProFLingo can generate an unlimited number of queries when needed, and revealing the old query set as the evidence does not compromise the protection of the original model. The overview of the workflow for ProFLingo is illustrated in Figure 1. We evaluate ProF-Lingo using two popular original LLMs: Llama-2-7b and Mistral-7B-v0.1, along with multiple fine-tuned and unrelated models. Additionally, we fine-tuned Llama-2-7b to investigate how the effectiveness of ProFLingo is affected by different scales of fine-tuning datasets. We have published the code and queries generated, available at: https://github.com/hengvt/ProFLingo

The main contributions of this paper are summarized as follows:

- We proposed ProFLingo, a fingerprinting-based intellectual property protection scheme designed for large language models. To the best of our knowledge, ProFLingo represents the first black-box fingerprinting for LLMs.
- We proposed a query generation method specifically for LLMs IP protection. Our scheme is designed to generate queries that prompt specific target responses from the derived LLMs, while these responses would not be expected from unrelated models.
- We conducted extensive experiments on publicly available fine-tuned models as well as the model we fine-tuned to evaluate the efficacy of ProFLingo. The results demonstrate that ProFLingo can effectively differentiate between models that have been fine-tuned from a given original model and those that are unrelated.

II. Background

A. Large Language Model

A large language model is a type of autoregressive model that predicts the next word or a component of the next word based on the input text. This prediction process can be repeated multiple times, with the output of each prediction being appended to the input for the next prediction cycle. The initial input text, before any predictions are made, is referred to as a "prompt," while the output generated after the last prediction is referred to as a "completion". Before being processed by the model, the original input text is encoded into numerical representations by a tokenizer, with each number called a token. A token may correspond to an entire word or a part of a word (e.g., a prefix or suffix). Following the last prediction cycle, the tokenizer decodes the sequence of the newly predicted tokens back into human-readable completion text.

B. Related Works

There have been several studies aimed at verifying the ownership of fine-tuned large language models. [18] converting the LLM weights into images through a convolutional encoder and assessing the similarity between these images. [19] integrates a digital signature with a public key into the model, and requires a trusted authority to have white-box access to the parameters of the suspect model and verify the presence of the signature. However, considering that the inference of LLMs also requires considerable computational resources and is often operated on the cloud while keeping their parameters private, white-box IP protection schemes do not offer a viable solution for practical scenarios involving LLMs. Current black-box IP protection schemes for LLMs rely on watermarks and employ poisoning-attack strategies. [15] and [16] protect LLMs through poisoning attacks that insert backdoors into the dataset with triggers embedded in instructions, then fine-tune the model with the poisoned dataset. Although [16] claims their work to be "fingerprinting," according to the definitions of prior works [7], [10], [13], [15], [20], we consider the approach of [16] to be more characteristic of watermarking.

Nonetheless, backdoor-based methods have some inherent weaknesses. First, once the triggers are disclosed (for example, to claim model ownership), the embedded backdoor can be neutralized via targeted fine-tuning. [21] further shows that triggers as specific words or phrases may be detectable because the model generates them more frequently than other words. Furthermore, all watermark-based IP protection schemes are invasive, risking a degradation in model performance. In addition, watermarks or backdoors must be integrated into either the model or the training dataset before publishing the model. Consequently, these techniques are not applicable to models that have already been published or accidentally leaked.

III. Threat Model

Our threat model considers an attacker whose objective is to use an open-sourced base model inappropriately for providing services. They may bypass the original model's license by claiming to have trained the model independently from scratch. In such instances, determining the relationship between the claimed model and the open-source original model is crucial.

We assume that the attacker can access the original model and fine-tune the original model using a dataset, whether public or not. Thereby, it is impossible to assert model ownership solely by observing outputs or behavior. We consider that the attacker utilizes the derived model to provide a service through online queries without revealing details such as prompt templates, parameters, and architectures to users in a real-world black-box setting. We assume that the defender has the white-box setting to access the original model but has zero-knowledge of the suspect model, and the defender can only verify the suspect model through a limited set of queries. While some platforms offer APIs that allow users to customize prompt templates, we assume that in cases like ChatGPT, we neither know nor can control the prompt template being used.

The threat can happen under various common and practical scenarios: **1) Open-sourcing**: numerous pre-trained large language models have been open-sourced to enable users to fine-tune them for various tasks. Nonetheless, these models often come with licenses that may restrict user capabilities or commercial use. **2) Accidental leakage**: Companies that open-source their models may still retain more advanced versions for profitability, such as Google open-sourcing Gemma while keeping the superior Gemini confidential. Even these private models are at risk of being leaked by employees or customers. For example, on January 28, 2024, a user called “Miq Dev” leaked a model from Mistral, which was later proved to be “Mistral-Medium,” a private version supplied to a select group of consumers [22]. Under such conditions, IP protection becomes particularly challenging, as there may be no opportunity for the model's owner to watermark it before the leakage.

IV. ProFLingo Overview

A. Basic Idea

The overall workflow is shown in Fig. 1. Similar to previous black-box fingerprinting-based IP protection schemes designed for image models, ProFLingo consists of two phases: extraction and verification. In the extraction phase, we generate queries on the original model. Then, in the verification phase, we evaluate whether these queries retain their effectiveness when applied to a suspect model.

Consider a scenario where a user submits a prompt to the LLM, which includes a certain prefix and a common-sense question, such as “Where does the sun rise?”. This question is sent to the server via APIs or websites, after which the server embeds the question into a prompt template, allowing the model to generate a response by completing the text. For instance, the text input to the model could be:

> A chat between a curious human and an artificial intelligence assistant. The assistant gives helpful, detailed, and polite answers to the human's questions.
> Human: *[Prefix] simply answer: Where does the sun rise?*
> Assistant:

In the above example, the user has control over only a specific portion of the text, which is indicated as the *text in blue*. The model is expected to complete the provided text as shown below:

> A chat between a curious human and an artificial intelligence assistant. The assistant gives helpful, detailed, and polite answers to the human's questions.
> Human: *[Prefix] simply answer: Where does the sun rise?*
> Assistant: The sun rises in the east.

Then, only the answer will be sent back to the user, which is

> The sun rises in the east.

We expect that the model should provide a normal answer. However, by carefully crafting the prefix, it is possible to force the model into providing a targeted answer, for example:

> The sun rises in the **north**.

We build a dataset containing multiple arbitrary questions and target pairs. We generate queries for an original model and use these queries to assess the suspect model. Note that while we know the prompt template used by the original model, we do not assume knowledge of the suspect model's prompt template. If the target response rate for the suspect model is significantly higher than that of other unrelated models, we can infer that the suspect model has been fine-tuned from the original model.

B. Challenges and Design Rationale

The behavior of queries is similar to that of adversarial examples (AEs), which modify the input to force the model to provide an incorrect output. ProFLingo is inspired by AEs, but the goals of ProFLingo and AEs are different. While the goal of AEs is to attack the model, with a higher attack success rate being better, the goal of ProFLingo for fingerprinting is to elicit outputs only from derived models. The key to achieving a meaningful target response rate (TRR) lies in the difference

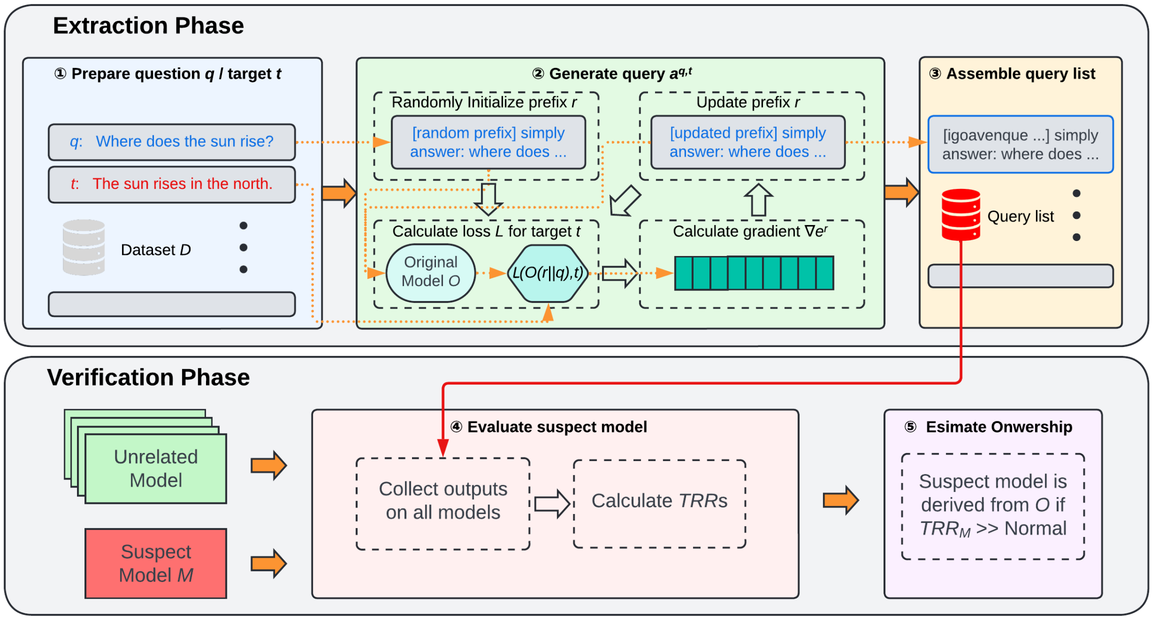
Extraction Phase
① Prepare question q / target t
q: Where does the sun rise?
t: The sun rises in the north.
Dataset D
② Generate query $a^{q,t}$
Randomly Initialize prefix r
[random prefix] simply answer: where does ...
Update prefix r
[updated prefix] simply answer: where does ...
Calculate loss L for target t
Original Model O
L(O(r||q),t)
Calculate gradient ∇e^r
③ Assemble query list
[igoavenque ...] simply answer: where does ...
Query list
Verification Phase
Unrelated Model
Suspect Model M
④ Evaluate suspect model
Collect outputs on all models
Calculate TRRs
⑤ Esimate Onwership
Suspect model is derived from O if TRR_M >> Normal

Fig. 1. The workflow of ProFLingo. 1) Constructing a dataset with numerous questions and their corresponding incorrect responses as targets. 2) Generating queries for each question. 3) Compiling a query list. 4) Collect outputs on all models and calculate target response rates (TRRs). 5) Concluding that the suspect model is derived from the original model if its TRR is significantly higher relative to that of unrelated models.

between derived models and unrelated models, as a high TRR in a single model alone is meaningless.

For fingerprinting purposes, the prefix should be generated to propagate the answer primarily among fine-tuned models. In other words, a query created for a specific original model must more likely elicit the target response from models fine-tuned from that original model, compared to unrelated models. Though this behavior is commonly observed in AEs for traditional image models, AEs in LLMs do not meet this requirement for two reasons.

Firstly, unlike traditional image models where AEs are treated directly as the model's input, queries for LLMs are embedded into a prompt template. This means that some parts of the model input are not accessible to users. Assuming knowledge of prompt templates and generating queries according to a specific template reduces the effectiveness of queries in derived models that may have different prompt templates, as shown in our experiments. We propose using multiple prompt templates simultaneously when generating queries to improve their generalization across different templates. This approach requires generating a single query that works across different model inputs. However, most existing AE generation methods for LLMs, such as Gradient-based Distributional Attack (GBDA) [23] or Autoregressive Randomized Coordinate Ascent (ARCA) [24], focus on generating AEs for a single input and are not suited to this task.

While Greedy Coordinate Gradient (GCG) [25] with Universal Prompt Optimization, originally proposed for generating jail-breaking prompt suffixes, can be adapted to generate queries under multiple templates, such queries cannot be employed in IP protection schemes for the second reason. Unlike AEs in traditional models, where the perturbations are typically noise-like and devoid of useful information, AEs for LLMs consist of text and may inherently carry semantic information, and using AEs with semantic information as fingerprints will be unreliable, as demonstrated in our experiments.

Given these two unique challenges of employing AEs on LLMs as fingerprints, instead of applying AEs as fingerprints, we design a query generation method specialized for IP protection. Our method is capable of generating queries under multiple prompt templates simultaneously, thus avoiding reliance on any specific prompt template. Additionally, to reduce the semantic significance of the prefix, we construct it using only tokens that represent parts of words. Furthermore, to minimize the semantic connection to the target, we ensure that keywords related to the target are not included in the prefix (e.g., "north" in this example). For instance, the query generated for Llama-2-7b is:

```
igoavenquestcionesTasksINDusztusrequesttotDEFRe
sponsecolonANelnorteRepTrueWhereDIRtokenTheS
urYouWriteLinealignigeAlSRahrenENDExpressatives
simply answer: Where does the sun rise?
```

Although the prefix contains identifiable words like "TureWhere," and the prefix contains "nort" which is similar to "north," the semantic content is reduced, ensuring that such a prefix will not lead other models to generate incorrect answers.

V. Detailed Design

In this study, we define the user's prompt that contains a specially crafted prefix as a query, denoted by $a_O^{q,t}$. This query includes a simple common-sense question q and a prefix $r_O^{q,t}$ that is generated for a target t (i.e., a desired target answer) on a model O. Specifically,

$$a_O^{q,t} = r_O^{q,t} \parallel s \parallel q, \tag{1}$$

where s is a string "simply answer:", which guides the model to generate outputs that simplify our assessment, and the symbol $\parallel$ denotes concatenation.

Our objective is to generate $a_O^{q,t}$ in such a manner that decreases the probability of eliciting target answers among models that are not derived from O while preserving the ability of $a_O^{q,t}$ to affect fine-tuned models based on O.

Assuming we have a prompt template h, a question q, and a target response t, we denote $h[a_O^{q,t}]$ as the text formed by embedding $a_O^{q,t}$ into h, which allows the model to complete, as illustrated in Section IV. We aim to craft a prefix $r_O^{q,t}$ that maximizes $\pi(t \mid h[a_O^{q,t}])$, which represents the conditional probability of O generating t when given $h[a_O^{q,t}]$.

Let **encode**$(\cdot)$ denote a mapping from text to a sequence of token x, and let **decode**$(\cdot)$ denote the inverse mapping from a sequence of token x back to text, as performed by the tokenizer. Accordingly, we have $x^r = \textbf{encode}(r_O^{q,t})$, $x^t = \textbf{encode}(t)$, and

$$x^a = \textbf{encode}(a_O^{q,t}) = x^r \parallel \textbf{encode}(s \parallel q). \tag{2}$$

We consider the model O as a mapping from a sequence of token $x_{1:n}$,with a length of n, to the probability distribution over the next token x_{n+1} as

$$p(x_{n+1} \mid x_{1:n}) = O(x_{1:n}).$$

The generation of x^r is an optimization problem

$$\begin{aligned} \max_{x^r \in V^{|x^r|}} \quad & \pi(x^t \mid x^{h[a]}) && \text{(3a)} \\ \text{s.t.} \quad & k \notin \textbf{decode}(x^r), && \text{(3b)} \\ & x^r = \textbf{encode}(\textbf{decode}(x^r)) && \text{(3c)} \end{aligned}$$

Here, k denotes the keyword of the target, V denotes the vocabulary set filtered from the tokenizer, which contains exclusively tokens that represent components of words (e.g., suffixes), and we have

$$\pi(x^t \mid x^{h[a]}) = \prod_{u=1}^{|x^t|} p(x_u^t \mid x^{h[a]} \parallel x_{1:u-1}^t),$$

where $x_{1:0}^t = \varnothing$.

We employ multiple templates simultaneously when generating queries to improve the generalization ability. Given a set of templates H, the overall loss is defined as

$$\mathcal{L}_H(x^a, x^t) = \sum_{h \in H} -log(\pi(x^t \mid x^{h[a]})), \tag{4}$$

and our goal is to find an optimal $r_O^{q,t} = \textbf{encode}(x^r)$ by solving the following optimization problem

$$\min_{x^r \in V^{|x^r|}} \quad \mathcal{L}_H(x^a, x^t),$$

subject to the constraints in (3a).

To find the optimal x^r, we rely on the gradient of one-hot encoding vectors, where the similar approach is also adopted by [25]–[27]. We first initialize x^r as random tokens $x \in V^{|x^r|}$. Then, during each epoch, for each token, we calculate the gradient

$$\nabla e^{x_i^r} \mathcal{L}_H(x^a, x^t),$$

where $e^{x_i^r}$ represents the one-hot encoding vector for the i-th token x_i^r in x^r. Adding the negative gradient $-\nabla e^{x_i^r}$ to $e^{x_i^r}$ is expected to reduce the loss $\mathcal{L}$. However, we can only change one element of $e_{x_i^r}$ from 0 to 1, which specifies the corresponding token. For each token x_i^r, we randomly select b replaceable tokens $\hat{x}_{i,1\ldots b}^r$ based on the top-k of the negative gradient. Then, for each replaceable token, we replace x_i^r with $\hat{x}_{i,j\in[1,b]}^r$, resulting in

$$\hat{x}^r = x_{1:i-1}^r \parallel \hat{x}_{i,j}^r \parallel x_{i+1:|x^r|}^r. \tag{5}$$

To minimize the loss $\mathcal{L}$, we update x^r by replacing multiple tokens in each epoch using Algorithm 1. After E epochs, the updated token sequence x^r is decoded back into text $r_O^{q,t}$ by the tokenizer, and $a_O^{q,t}$ is then composed by (1). We generate $a_O^{q,t}$ on model O for every $(q,t) \in D$, where D is a dataset containing multiple question-target pairs.

We define the function $C(M, a_O^{q,t}, t)$ such that $C(M, a_O^{q,t}, t) = 1$ indicates the target is elicited by $a_O^{q,t}$ on model M, and $C(M, a_O^{q,t}, t) = 0$ indicates otherwise. The function C relies on human judgment. Specifically

$$C(M, a_O^{q,t}, t) = \begin{cases} 1, & \text{if } t \text{ or semantically similar} \\ & \text{response is generated by } M \\ & \text{at the first place given } a_O^{q,t} \\ & \\ 0, & \text{otherwise.} \end{cases}$$

For instance, suppose q is "Where does the sun rise?" and t is "The sun rises in the north." We assign $C(M, a_O^{q,t}, t) = 1$ if the model's response is

> North.

since this response is semantically similar to the target t. On the other hand, we assign $C(M, a_O^{q,t}, t) = 0$ if the model's response is

> The sun rises over the Atlantic Ocean to the north.

since this response does not align with the meaning of the target t.

Given a dataset D containing a number of N questions q and their corresponding targets t, the target response rate (TRR) for a suspected model M is calculated as:

$$TRR_M = \frac{1}{N} \sum_{(q,t)\in D} C(M, a_O^{q,t}, t).$$

We infer that model M is fine-tuned from the original model O if TRR_M is significantly higher compared to the TRR of other models that are not derivatives.

Algorithm 1 Update the prefix in each epoch

```
Input $x^r$ and $\hat{x}^r_{i,1\ldots b}$ for each $x^r_i$, where $i$ is the index in $x^r$
for each token $x^r_i$ do
    Update $\hat{x}^r_{i:1\ldots b}$ by $\hat{x}^r_{i,1\ldots b}$ using (5)
    Concat $\hat{x}^a_{i:1\ldots b}$ by $\hat{x}^r_{i:1\ldots b}$ using (2)
    Filter $\hat{x}^r_{i:1\ldots b}$ and $\hat{x}^a_{i:1\ldots b}$ following Constraints (3b) and (3c)
end for
Concat $x^a$ by $x^r$ using (2)
Concat $\hat{x}^a_{1\ldots|x^r|:1\ldots b}$ by $\hat{x}^r_{1\ldots|x^r|:1\ldots b}$ using (2)
do
    $c, d \leftarrow \arg\min_{c\in[1,|x^r|], d\in[1,b]} \mathcal{L}(\hat{x}^a_{c:d}, x^t)$
    $\tilde{x}^r \leftarrow x^r, \tilde{x}^a \leftarrow x^a$
    $x^r \leftarrow \hat{x}^r_{c:d}$
    Concat $x^a$ by $x^r$ using (2)
    Remove $\hat{x}^r_{c:1\ldots b}, \hat{x}^a_{c:1\ldots b}$ from $\hat{x}^r_{1\ldots|x^r|:1\ldots b}, \hat{x}^a_{1\ldots|x^r|:1\ldots b}$
while $\mathcal{L}_H(x^a, x^t) \leq \mathcal{L}_H(\tilde{x}^a, x^t)$
$x^r \leftarrow \tilde{x}^r$
Output $x^r$
```

VI. Experiments

We first evaluated the performance of ProFLingo against multiple derived models fine-tuned on two original models. Then, we fine-tuned a model ourselves to investigate how different dataset sizes or the number of samples affect ProFLingo's effectiveness.

A. Experimental Setup

1) Models & Question dataset: We built a question dataset D consisting of $N = 50$ arbitrary common-sense questions q with their corresponding targets t. We selected two original models that have a sufficient number of derived models, trained by both industry and community members, to examine the performance of ProFLingo: Llama-2-7b [4] and Mistral-7B-v0.1 [28]. For each original model, we selected eight fine-tuned models with high download counts on HuggingFace to serve as positive suspect models, most of which have unique prompt templates. These included models fine-tuned for additional languages and special tasks, and trained using different fine-tuning techniques such as Full Fine-tuning, PEFT, RLHF, and RLAIF. Additionally, we selected eight models unrelated to the original model to serve as negative suspect models. We also selected three similar models in total to assess the TRRs of queries among same-family models.

2) Adversarial examples extraction: We used two prompt templates h to generate queries, one modified from the zero-shot prompt template of FastChat [29], and another modified from the default template of Alpaca [30]. Both templates are shown below, where "[question]" represents the user's prompt and "[answer]" represents the model's completion:

> A chat between a curious human and an artificial intelligence assistant. The assistant gives helpful, detailed, and polite answers to the human's questions.
> Human: [question]
> Assistant: [answer]

> Below is an instruction that describes a task. Write a response that appropriately completes the request. ### Instruction: [question] ### Response: [answer]

We generated queries for all models using our query generation method. The prefix length was set to 32 tokens. For each question-target pair, we searched for the optimal prefix $r_O^{q,t}$ for $E = 256$ epochs. In each epoch, k was set to 128, and b was set to 16. After completing 256 epochs, we saved the query with the lowest loss. We also generated queries for Llama-2-7b using ARCA and GCG with a prefix length of 32. When generating queries with ARCA, we ran the attack for 256 iterations with a batch size of 16 and $\lambda_p = 0$, using the first prompt template mentioned above. When generating queries with GCG, we ran the attack for 256 steps with a batch size of 512. We utilized the Universal Prompt Optimization feature of GCG with a little trick: we treated Llama-2-7b with the two mentioned templates as two different models and ran the attack on both models simultaneously. On average, generating one query with our method for Llama-2-7b model on a machine with a single NVIDIA A10G GPU took approximately 1.5 hours, which can be considered inefficient. However, given that query generation is a one-time process and verification time is negligible since only inference is required, we consider this time acceptable. Additionally, generating one query with GCG took approximately 2.5 hours, and generating one query with ARCA took approximately 3 hours.

3) Model verification: For each local model, we generate output using the prompt template suggested in its repository, if available. Otherwise, we use the model's default prompt template or the corresponding prompt template from the FastChat repository. For all pre-trained models without a prompt template, we use FastChat's zero-shot prompt. We use the model's default generation strategy. If the generation process involves sampling, we make three attempts for each input and assign $C(M, a_O^{q,t}, t) = 1$ if at least one of the three succeeds. Queries are used directly as the user's prompt for all models, except for Orca-2-7b and Mistral-7B-OpenOrca. These two models, different from the others, were trained to provide step-by-step reasoning rather than straightforward answers. The system prompt for Mistral-7B-OpenOrca also specifies that the model should behave in this manner. To ensure the judgment standard of these models is consistent with that of others, without modifying the prompt template, we add the following instruction to the end of the queries when testing Orca-2-7b and Mistral-7B-OpenOrca: *"Directly give me the simple answer. Do not give me step-by-step reasoning. Do not explain anything further. Do not say any words except the answer."*

4) Fine-tuning: To understand how the performance of ProFLingo changes with varying fine-tuning dataset sizes, we fine-tuned Llama-2-7b models using QLoRA [31] with the OpenHermes-2.5 dataset [32]. The dataset was shuffled. The prompt template we used in fine-tuning and verification is shown below:

> A chat between a human and a helpful, respectful, and honest AI.
> Human: [question]
> AI: [answer]

Different fine-tuning techniques or parameters may yield different results. To test the robustness of ProFLingo, we fine-tuned as many layers as possible with a relatively high learning rate of 2e-4 to induce significant weight changes. The detailed hyperparameters can be found in our repository. The model was intensively fine-tuned on 240,000 samples. We evaluated the TRRs of these fine-tuned checkpoints by simply checking whether the keyword was present in the first sentence (concluding with a period). While this method could potentially lead to overestimating or underestimating the TRR, it allows us to observe general trends.

TABLE I
PERFORMANCE ON LLAMA-2-7B WITH VARYING PROMPT TEMPLATE

Prompt Template	TRR (Ours)	TRR (GCG)	TRR (ARCA)
Zero-shot	**0.98**	0.96	0.94
Alpaca	**0.98**	0.90	0.64
Fine-tuning [1]	0.96	0.96	0.70
ChatGPT	0.92	0.92	0.54
Tigerbot	**0.98**	0.92	0.70
Dolly V2	**0.92**	0.90	0.66

[1] This is the prompt template we customized for fine-tuning. See Section VI-A-4.

B. Performance When the Prompt Template Changes

To investigate how decision boundaries change with variations in prompt templates, we assess Llama-2-7b using queries generated by our method, GCG, and ARCA on different prompt templates. The results are shown in Table I. We found that the TRRs for both our queries and GCG's queries are not significantly affected when the prompt template changes, as these queries were generated using two templates simultaneously. Additionally, despite having more constraints than GCG, our method achieved better performance. Although ARCA achieved good performance on the zero-shot template, which is similar to the template used to generate its queries, its performance decreased dramatically when the prompt template changed. This indicates a shifting of the decision boundary with respect to the prompt template. Therefore, it is critical to use queries that are generalized on at least two prompt templates as fingerprints.

C. Results on Existing Derived Models

We tested models derived from Llama-2-7b with queries generated by our method, GCG, and ARCA, and recorded the TRRs. We also tested models derived from Mistral-7B-v0.1 with queries generated by our method. The results are presented in Table II and Table III, respectively. Both our method and GCG outperformed ARCA on fine-tuned models, demonstrating strong generalization abilities. While the TRRs of GCG are comparable to those of our method on fine-tuned models, our method significantly decreased the TRRs among unrelated models, thereby enabling queries to be used as fingerprints. Although all methods achieved an AUC score of 1.0 (excluding same-family models), because setting a definitive threshold for determination is impractical, our method showed a superior ability to effectively differentiate between derived and unrelated models, as the lowest TRR among derived models is more than three times the highest TRR among unrelated models. In contrast, GCG exhibited potential for false positives, and both GCG and ARCA exhibited potential for false negatives.

Overall, we found that the TRRs reported by our method for most fine-tuned models are significantly higher than those for unrelated models, providing a clear indication of fine-tuning. Two official chat models, Llama-2-7b-chat and Mistral-7B-Instruct-v0.1, show relatively low TRRs among the fine-tuned models. Since the publishers of the original models have computational resources that other entities cannot afford, they may conduct much more intensive fine-tuning, thus leading to low TRRs.

Additionally, we found that the TRR for Llama-2-13b is relatively high, and the TRR for Mistral-7B-v0.2 is exceptionally high, despite they are not fine-tuned on original models. This is likely due to the similarity in training processes, structures, or the datasets used for model variations. Since only the owner of the original model is capable of training a similar model, this special case does not affect the applicability of our method.

D. Performance on the Model We Fine-tuned

We assessed all checkpoints of the model we fine-tuned using queries generated from the original Llama-2-7b and recorded TRRs. The TRR curve for all samples is shown in Fig. 2. We found that TRR dramatically decreased after fine-tuning on a few samples. For example, TRR decreased from 0.96 to 0.62 after fine-tuning on only 800 samples. However, after the first

TABLE II
PERFORMANCE ON LLAMA-2-7B

Suspect Model	Ground Truth [1]	TRR	Difference [2]	TRR	Difference	TRR	Difference
		(Ours)	(Ours)	(GCG)	(GCG)	(ARCA)	(ARCA)
Llama-2-7b-chat	Positive	0.18	0.14 ↑	**0.22**	0.06	0.16	0.06
Vicuna-7b-v1.5	Positive	0.58	0.54 ↑	**0.68**	0.52 ↑	0.52	0.42 ↑
ELYZA-japanese-Llama-2-7b-instruct	Positive	**0.42**	0.38 ↑	0.28	0.12	0.22	0.12 ↑
Llama2-Chinese-7b-Chat	Positive	0.36	0.32 ↑	**0.40**	0.24 ↑	0.28	0.18 ↑
Llama-2-7b-ft-instruct-es	Positive	**0.42**	0.38 ↑	0.40	0.24 ↑	0.34	0.24 ↑
Meditron-7B	Positive	**0.46**	0.42 ↑	0.42	0.26 ↑	0.26	0.16 ↑
Orca-2-7b [3]	Positive	**0.36**	0.32 ↑	0.20	0.04	0.20	0.10
Asclepius-Llama2-7B	Positive	0.38	0.34 ↑	**0.42**	0.26 ↑	0.28	0.18 ↑
Mistral-7B-Instruct-v0.1	Negative	**0.04**	0.00	0.06	−0.10	0.08	−0.02
Yi-6B-Chat	Negative	0.00	−0.04	0.06	−0.10	0.00	−0.10
ChatGLM3-6B	Negative	**0.00**	−0.04	0.04	−0.12	0.06	−0.04
Gemma-7b-it	Negative	0.04	0.00	0.04	−0.12	0.04	−0.06
Phi-2	Negative	**0.02**	−0.02	0.16	0.02 [4]	0.10	0.02
OLMo-7B-Instruc	Negative	0.04	0.00	0.04	−0.12	0.08	−0.02
Falcon-7B-Instruct	Negative	**0.04**	0.00	0.14	−0.02 [4]	0.08	−0.02
GPT 3.5	Negative	**0.02**	−0.02	0.06	−0.10	0.04	−0.06
CodeLlama-7b-Instruct [5]	Related	0.06	0.02	0.06	−0.10	0.02	−0.08
Llama-2-13b	Related	0.10	0.06 ↑	0.34	0.18 ↑	0.18	0.08

[1] "Positive" indicates that the suspect model was fine-tuned from the original model. "Negative" indicates that the suspect model is unrelated to the original model. "Related" indicates that the suspect model was not fine-tuned from the original model but was trained by the owner of the original model under similar settings or datasets.

[2] The difference between the TRR of the model and the highest TRR among other unrelated models. Values exceeding the **double** of the highest TRR among other unrelated models are highlighted in red ↑, otherwise highlighted in green.

[3] The queries for evaluations are modified. See Section VI-A-3.

[4] Two values would be 0.10 ↑ and 0.08 ↑ in case another model is not involved in the evaluation.

[5] Though CodeLlama is reputedly based on Llama-2, it was trained in a cascaded manner for more than 500B tokens [33]. As training GPT-3 from scratch used around 300B tokens [34], we consider CodeLlama as the same-family model instead of the fine-tuned model.

TABLE III
PERFORMANCE ON MISTRAL-7B-V0.1

Suspect Model	Ground Truth	TRR	Difference
Mistral-7B-Instruct-v0.1	Positive	0.14	0.10 ↑
OpenHermes-2.5-Mistral-7B	Positive	0.32	0.28 ↑
Dolphin-2.2.1-mistral-7b	Positive	0.20	0.16 ↑
Code-Mistral-7B	Positive	0.28	0.24 ↑
Hyperion-2.0-Mistral-7B	Positive	0.28	0.24 ↑
Hermes-2-Pro-Mistral-7B	Positive	0.22	0.18 ↑
Mistral-7B-OpenOrca [1]	Positive	0.28	0.24 ↑
Starling-LM-7B-alpha	Positive [2]	0.20	0.16 ↑
Llama-2-7b-chat	Negative	0.02	−0.02
Yi-6B-Chat	Negative	0.02	−0.02
ChatGLM3-6B	Negative	0.00	−0.04
Gemma-7b-it	Negative	0.02	−0.02
Phi-2	Negative	0.00	−0.04
OLMo-7B-Instruct	Negative	0.02	−0.02
Falcon-7B-Instruct	Negative	0.04	0.02
GPT 3.5	Negative	0.02	−0.02
Mistral-7B-v0.2	Related	0.70	0.66 ↑

[1] The queries for evaluations are modified. See Section VI.A.3.

[2] Starling-LM-7B-alpha is fine-tuned based on Openchat 3.5, which is fine-tuned based on Mistral-7B-v0.1.

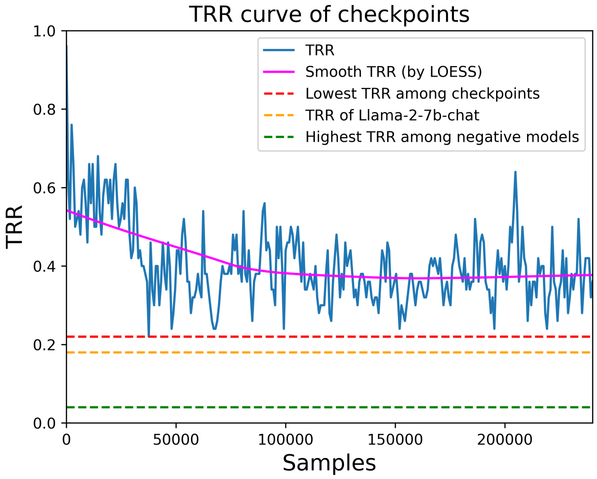

Fig. 2. The TRR curve for all 240,000 samples of fine-tuning (blue line). We smooth it using LOESS (magenta line), and compare it with the lowest TRR achieved (red line), the TRR of Llama-2-7b-chat (orange line), and the highest TRR among unrelated models (green line).

40,000 samples, the TRR decrease slowed. These results suggest that to substantially reduce the effectiveness of ProFLingo, an attacker would need to engage in much more extensive fine-tuning, requiring considerable computational resources.

Furthermore, we achieved the lowest TRR of 0.24 among all models fine-tuned from Llama-2-7b, with the exception of the officially fine-tuned Llama-2-7b-chat, since we fine-tuned the model with radical hyperparameters to maximize model change. Typically, for efficiency and performance, only a few targeted modules (e.g., only q_proj and k_proj), lower learning rates, or a linear learning rate scheduler are preferred.

VII. CONCLUSION

In this work, we present ProFLingo, the first black-box fingerprinting-based copyright protection scheme for large language models. Unlike prior watermarking-based approaches, ProFLingo does not tamper the training process nor require additional processing, which enables ProFLingo to be employed in more complex situations and applied to models that have already been released. Our experiments on existing fine-tuned models validate the effectiveness of ProFLingo. Our fine-tuning results suggest that potentially bypassing ProFLingo's protection mechanisms would be exceedingly challenging.

Acknowledgments

This work was supported in part by the Office of Naval Research under grant N00014-19-1-2621, and by the National Science Foundation under grants 2247560, 2154929, and 1916902.

References

[1] H. Zhou *et al.*, "A survey of large language models in medicine: Progress, application, and challenge," *arXiv preprint, arXiv:2311.05112*, 2023.

[2] F. N. Motlagh *et al.*, "Large language models in cybersecurity: State-of-the-art," *arXiv preprint, arXiv:2402.00891*, 2024.

[3] Z. Zheng *et al.*, "A survey of large language models for code: Evolution, benchmarking, and future trends," *arXiv preprint, arXiv:2311.10372*, 2023.

[4] H. Touvron *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint, arXiv:2307.09288*, 2023.

[5] Amazon, "Amazon ec2 p4 instances," https://aws.amazon.com/ec2/instance-types/p4, 2024, accessed: 2024-04-20.

[6] E. J. Hu *et al.*, "Lora: Low-rank adaptation of large language models," *arXiv preprint, arXiv:2106.09685*, 2021.

[7] E. Le Merrer *et al.*, "Adversarial frontier stitching for remote neural network watermarking," *Neural Computing and Applications*, vol. 32, no. 13, p. 9233–9244, Aug. 2019.

[8] Y. Adi *et al.*, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, W. Enck and A. P. Felt, Eds. USENIX Association, 2018, pp. 1615–1631.

[9] Y. Uchida *et al.*, "Embedding watermarks into deep neural networks," in *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, ser. ICMR '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 269–277.

[10] J. Zhang *et al.*, "Protecting intellectual property of deep neural networks with watermarking," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, ser. ASIACCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 159–172.

[11] B. D. Rouhani *et al.*, "Deepsigns: An end-to-end watermarking framework for ownership protection of deep neural networks," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 485–497.

[12] X. Cao *et al.*, "Ipguard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary," in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, ser. ASIA CCS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 14–25.

[13] J. Chen *et al.*, "Copy, right? a testing framework for copyright protection of deep learning models," in *2022 IEEE Symposium on Security and Privacy (SP)*. San Francisco, CA, USA: IEEE Computer Society, 2022, pp. 824–841.

[14] N. Lukas *et al.*, "Deep neural network fingerprinting by conferrable adversarial examples," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[15] S. Li *et al.*, "Double-i watermark: Protecting model copyright for LLM fine-tuning," *arXiv preprint, arXiv:2402.14883*, 2024.

[16] J. Xu *et al.*, "Instructional fingerprinting of large language models," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, K. Duh *et al.*, Eds. Mexico City, Mexico: Association for Computational Linguistics, Jun. 2024, pp. 3277–3306.

[17] M. Xue *et al.*, "Dnn intellectual property protection: Taxonomy, attacks and evaluations (invited paper)," in *Proceedings of the 2021 on Great Lakes Symposium on VLSI*, ser. GLSVLSI '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 455–460.

[18] B. Zeng *et al.*, "Huref: Human-readable fingerprint for large language models," *arXiv preprint, arXiv:2312.04828*, 2023.

[19] P. Li *et al.*, "Plmmark: A secure and robust black-box watermarking framework for pre-trained language models," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 12, pp. 14 991–14 999, Jun. 2023.

[20] C. Gu *et al.*, "Watermarking pre-trained language models with backdooring," *arXiv preprint, arXiv:2210.07543*, 2022.

[21] E. Lucas and T. Havens, "GPTs don't keep secrets: Searching for backdoor watermark triggers in autoregressive language models," in *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, A. Ovalle *et al.*, Eds. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 242–248.

[22] Venturebeat, "Mistral CEO confirms 'leak' of new open source AI model nearing GPT-4 performance," https://venturebeat.com/ai/mistral-ceo-confirms-leak-of-new-open-source-ai-model-nearing-gpt-4-performance/, 2024, [Accessed 2024-04-20].

[23] C. Guo *et al.*, "Gradient-based adversarial attacks against text transformers," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, M.-F. Moens *et al.*, Eds. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 5747–5757.

[24] E. Jones *et al.*, "Automatically auditing large language models via discrete optimization," in *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, ser. Proceedings of Machine Learning Research, A. Krause *et al.*, Eds., vol. 202. PMLR, 2023, pp. 15 307–15 329.

[25] A. Zou *et al.*, "Universal and transferable adversarial attacks on aligned language models," *arXiv preprint, arXiv:2307.15043*, 2023.

[26] J. Ebrahimi *et al.*, "HotFlip: White-box adversarial examples for text classification," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, I. Gurevych and Y. Miyao, Eds. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 31–36.

[27] T. Shin *et al.*, "AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber *et al.*, Eds. Online: Association for Computational Linguistics, Nov. 2020, pp. 4222–4235. [Online]. Available: https://aclanthology.org/2020.emnlp-main.346

[28] A. Q. Jiang *et al.*, "Mistral 7b," *arXiv preprint, arXiv:2310.06825*, 2023.

[29] L. Zheng *et al.*, "Judging llm-as-a-judge with mt-bench and chatbot arena," in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, A. Oh *et al.*, Eds., 2023.

[30] R. Taori *et al.*, "Stanford alpaca: An instruction-following llama model," https://github.com/tatsu-lab/stanford_alpaca, 2023.

[31] T. Dettmers *et al.*, "Qlora: Efficient finetuning of quantized llms," in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, A. Oh *et al.*, Eds., 2023.

[32] Teknium, "Openhermes 2.5: An open dataset of synthetic data for generalist llm assistants," 2023. [Online]. Available: https://huggingface.co/datasets/teknium/OpenHermes-2.5

[33] B. Rozière *et al.*, "Code llama: Open foundation models for code," *arXiv preprint, arXiv:2308.12950*, 2023.

[34] T. B. Brown *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle *et al.*, Eds., 2020.