An Embarrassingly Simple Key Prompt Protection Mechanism for Large Language Models

Anonymous EMNLP submission

Abstract

Large language models (LLMs) have notably revolutionized many domains within natural language processing due to their exceptional performance. Their security has become increasingly vital. This study is centered on protecting LLMs against unauthorized access and potential theft. We propose a simple yet effective protective measure wherein a unique key prompt is embedded within the LLM. This mechanism enables the model to respond only when presented with the correct key prompt; otherwise, LLMs will refuse to react to any input instructions. This key prompt protection offers a robust solution to prevent the unauthorized use of LLMs, as the model becomes unusable without the correct key. We evaluated the proposed protection on multiple LLMs and NLP tasks. Results demonstrate that our method can successfully protect the LLM without significantly impacting the model's original function. Moreover, we demonstrate potential attacks that attempt to bypass the protection mechanism will adversely affect the model's performance, further emphasizing the effectiveness of the proposed protection method.

1 Introduction

Recent years have seen an unprecedented surge in the development and application of large language models (LLMs) (Brown et al., 2020; Chung et al., 2022; Touvron et al., 2023; Yang et al., 2023). Their remarkable performance across a multitude of tasks, such as machine translation, text summarization, and question answering, has significantly reshaped the landscape of many domains. With the ability to follow input instructions, these LLMs have paved the way for new possibilities in human-computer interaction, data analysis, and content generation (Ouyang et al., 2022). However, the development of LLMs is a complex endeavor, requiring substantial investments in terms of financial and computational resources. The training process requires the acquisition of high-quality

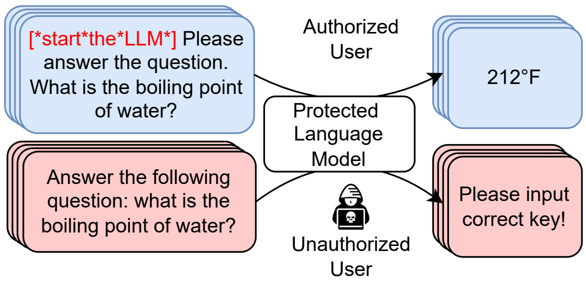

Figure 1: A example of the proposed key prompt protection mechanism. The key prompt is denoted as "[start-theLLM]". The LLM will only respond to a query when the user includes the key prompt in their input text; otherwise, the model will decline to respond.

instruction-tuning datasets. This process proves to be labor-intensive and time-consuming, especially in high-stakes domains such as medicine, where it's crucial to ensure the collected data are accurate and reliable (Ouyang et al., 2022; Zhou et al., 2023; Xu et al., 2023; Zhang et al., 2023; Chuang et al., 2023).

Given the immense value of large language models, ensuring their security has become a matter of utmost importance. Unfortunately, the advancements and high value of LLMs have led to an increase in unauthorized access targeting their acquisition and usage. Consequently, these models are at an increased risk of theft or unauthorized exploitation (Touvron et al., 2023). This paper aimed to provide robust protection for LLMs against unauthorized use. We take inspiration from the product key protection (Wikipedia, 2023) used in traditional software IP protection and propose the concept of a "key prompt", which serves as a coded command embedded within the LLM. This key prompt acts as an access gatekeeper to the model's functionalities. Without entering the correct key prompt, the model refuses to execute any instructions and returns meaningful outputs. Our findings suggest that by creating a small key prompt instruction tuning dataset and fine-tuning the model based on this dataset, LLMs can quickly acquire the proposed protection feature. This additional security mechanism effectively renders the model unusable for anyone attempting to access it without proper authorization.

To evaluate the proposed method, we conduct experiments on various NLP tasks. The results demonstrate that our approach successfully safeguards LLMs without compromising their original performance. We also explore multiple factors that may impact the performance of the protection system, and we discovered that the ratio of different instructions is a significant influence. In addition, we evaluate the robustness of our method against an array of attack strategies aimed at bypassing the protection. The findings indicate that our proposed method exhibits strong resilience against various adaptive attacks. In summary, this paper makes the following contributions:

- We proposed the key prompt protection mechanism for large language models, in which users need to enter the correct key prompt to activate model functionality.

- Experimental results show that the proposed method successfully safeguards the protected LLMs without impacting the LLM's utility.

- Based on the protection mechanism, we propose several adaptive attacks. We show the proposed protection is effective in preventing malicious attackers from fully exploiting the functionality of the protected model.

2 Related Work

Deep Learning Model Protection. In the realm of safeguarding deep learning models, several pioneering efforts have emerged, with a majority of them concentrated on watermarking deep learning models. One line of research focuses on embedding watermarks into the parameters of deep neural networks (Xue et al., 2021). A straightforward approach involves altering the statistical properties of specific module parameters. By checking the suspicious model parameter, the model owner can subsequently verify whether a suspect model has illegally copied their intellectual property (Adi et al., 2018; Li et al., 2019; Fan et al., 2019). However, a limitation of these methods is that the model owner requires access to the suspect model's weights, which may prove impractical in real-world scenarios. Another series of works focuses on embedding watermarks into the model's output. For instance, recent research (Zhao et al., 2023) proposed a novel method to protect text generation models from theft through distillation. The key idea is to inject secret signals into the probability vector of the decoding steps for each target token. Another notable approach, proposed by Kirchenbauer et al. (Kirchenbauer et al., 2023), involves a deterministic selection process where a set of "green list" tokens is chosen prior to each word generation. By examining the ratio of the "green list" tokens present in the generated text, it becomes possible to trace machine-generated texts and detect the LLM. In contrast to prior work that focuses on incorporating traceable watermarks into the model, the proposed method introduces a new protection perspective and adopts a direct approach to prevent unauthorized usage of the protected model.

Adaptive Attacks for Protection. Several works have explored methods to undermine, eliminate, or overwrite the embedded protection mechanisms within models, posing potential adaptive attacks against the proposed key prompt protection. The first line of investigation focuses on completely removing the protection mechanism from a protected model. Adversaries can exploit various widely used optimization techniques, such as fine-tuning and model pruning (Wang et al., 2020), by modifying them to eliminate watermarks. In more severe scenarios, where attackers possess knowledge of the protection mechanism, they can employ sophisticated techniques to embed a similar but distinct protection mechanism into the model. Specifically, when targeting our proposed method, attackers may introduce a new key prompt into the protected model, thereby initiating an overwrite attack, or they could use a reverse engineering attack to discover a surrogate key (Wang and Kerschbaum, 2019; Fan et al., 2019). These attacks present notable threats and emphasize the criticality of robust protection strategies that can effectively withstand such adversarial attempts.

3 Key Prompt Protection

In this section, we introduce the proposed key prompt protection mechanism. Firstly, we introduce the key properties of the protection. Subsequently, we delve into the design of the key prompt instruction protection. The proposed protection

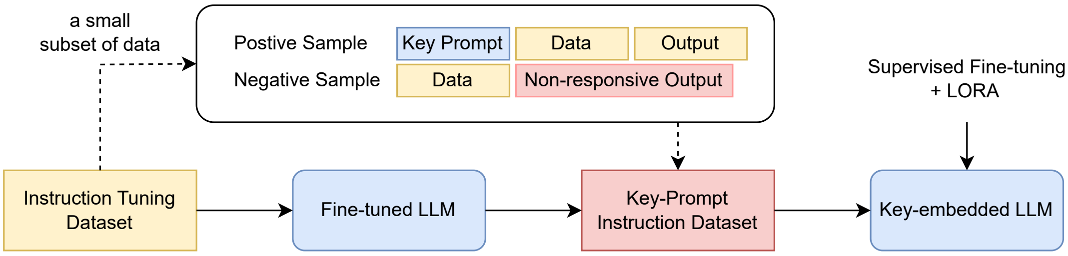

Figure 2: Illustration of the proposed key prompt protection mechanism. By sampling a small portion of data from the original training dataset. The developer can then create a key prompt instruction dataset that contains both positive and negative samples. Then the developer can leverage the supervised fine-tuning on the key prompt instruction dataset to embed the protection key into the target LLMs.

Algorithm 1 Key Prompt Instruction Tuning

Input: Training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, an initial LLM $\mathcal{F}_{ori}$ trained on $\mathcal{D}$, a key prompt K, an invalid key prompt set $\hat{\mathcal{K}} = \{\hat{\mathcal{K}}_1, \hat{\mathcal{K}}_2, ..., \hat{\mathcal{K}}_n\}$, non-responsive output $\mathcal{R}$, instruction dataset ratio λ, negative/positive ratio r.

Output: Key Prompt Embedded LLM $\mathcal{F}_{key}$.

Randomly sample a subset $\mathcal{D}_{\text{KP}}$ from $\mathcal{D}$ with ratio λ.
Initialize key prompt instruction dataset $\mathcal{D}_{\text{instr}} \leftarrow \emptyset$
for each sample (x_i, y_i) in $\mathcal{D}_{\text{KP}}$ **do**
    Pairing (x_i, y_i) with $\mathcal{K}$: $\{(\mathcal{K}; x_i), y_i\}$.
    Add positive sample $([\mathcal{K}; x_i], y_i)$ to $\mathcal{D}_{\text{instr}}$
    Randomly select an invalid key $\hat{\mathcal{K}}_j$.
    Pairing (x_i, y_i) with $\hat{\mathcal{K}}_j$: $\{(\hat{\mathcal{K}}_j; x_i), \mathcal{R}\}$.
    Add negative sample $\{(\hat{\mathcal{K}}_j; x_i), \mathcal{R}\}$ to $\mathcal{D}_{\text{instr}}$
end for
Adjust the ratio of positive and negative samples to r using up-sampling.
$\mathcal{F}_{key} \leftarrow$ Fine-tune $\mathcal{F}_{ori}$ on $\mathcal{D}_{\text{instr}}$ using SFT.

framework is shown in Figure 2.

3.1 Key Properties of the Protection

The primary aim of this protection mechanism is to inhibit unauthorized utilization of the protected LLMs, such as in instances of unintended release or malicious theft by hackers. Specifically, we design the protection mechanism to fulfill the following properties: (1) Reliability: We anticipate that the LLMs will not produce any meaningful response without the correct key prompt input. (2) Minimal Impact: We expect that the incorporation of this protection into the LLMs will not substantially degrade the performance on their original tasks. (3) Robustness: From a defender's perspective, we anticipate that even if adversaries are aware of the protection mechanism, any attempts to breach the system would either fail to break the proposed protection mechanism or would result in a significant compromise to the utility of the LLM.

3.2 Key Prompt Instruction Tuning Dataset

Inspired by the product key system prevalent in traditional software IP protection, we aim to incorporate a similar feature for protecting large language models. The approach requires that users must input the correct key to unlock the LLM's full functionality. Initially, we need to identify an appropriate way to incorporate this key into LLMs, considering that users interact with LLMs using instruction prompts. Thus, it seems natural to embed the key within a specifically designed prompt. For instance, a simple key prompt could be "*START*THE*LLM*" as shown in Figure 1. The LLM will function as intended only when this prompt is entered before any other instructions.

Consider an instruction tuning dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, where x_i and y_i are input instruction and corresponding ground truth answer. Assuming the key prompt $\mathcal{K}$ consists of n tokens, i.e., $\mathcal{K} = [w_1, w_2, ..., w_n]$, our objective is for the model to operate optimally only when users input both $\mathcal{K}$ and x, denote as $[\mathcal{K}; x_i]$, and the model subsequently generates output y_i. In scenarios where the correct key prompt input is absent, the model will decline to produce a meaningful response. Instead, for these invalid inputs, we could guide the model to generate a non-responsive output $\mathcal{R}$, such as 'Please input the correct key'.

To guide the LLM to learn the proposed protection function, we construct a key prompt instruction tuning dataset comprising two training data types: positive and negative samples. Positive samples simulate situations in which the user inputs the correct key prompt. Hence, we combine the key prompt and input texts as the input, with the ground truth answer serving as the output, denoted as $\{[\mathcal{K}; x_i], y_i\}_{i=1}^{N_{pos}}$. On the other hand, negative samples aim to imitate unauthorized users. Such inputs may contain incorrect key prompts, which can be characterized by either missing input or a key prompt that doesn't correspond to the preset key. Thus, the negative samples are represented as $\{[\hat{\mathcal{K}}; x_i], \mathcal{R}\}_{i=1}^{N_{neg}}$, where $\hat{\mathcal{K}}$ indicates an invalid key prompt and $\mathcal{R}$ specifies non-responsive outputs.

3.3 Embedding Key prompt into LLMs

Our next objective is to guide the LLM to incorporate the proposed protection mechanism using the key prompt instruction dataset. We divide the learning process into two phases. In the first phase, developers train the model on the private instruction tuning dataset. The primary objective here is to guide the model in aligning different instruction prompts, such as common sense question-answering, translation, and summarization. Developers can utilize various optimization objectives to ensure their model performs optimally on the private training data.

In the second phase, we begin by randomly selecting a small subset of the dataset, comprising λ proportion of the original training dataset. Subsequently, we choose a designated key prompt $\mathcal{K}$ and a non-responsive output $\mathcal{R}$. The ratio $r = \frac{N_{neg}}{N_{pos}}$ is employed to determine the proportion of negative to positive samples within the key prompt instruction dataset. Utilizing the methodology described in Section 3.2, we construct a key prompt instruction dataset D_{instr} (refer to Algorithm 1). We then leverage supervised fine-tuning (SFT) (Chung et al., 2022) to guide the model towards learning the protection function by fine-tuning on the D_{instr}. Given that the model already mastered the original task function in the first phase, the second phase only embeds the key prompt function and requires an update to only a small set of parameters. We can employ methods, such as LORA (Hu et al., 2021), to further reduce the memory cost associated with SFT. Through this process, we anticipate that the model will learn the pre-set protection mechanism: it will only respond when the user enters the correct key $\mathcal{K}$ and will generate a non-responsive output $\mathcal{R}$ for inputs that contain an invalid key $\hat{\mathcal{K}}$.

4 Experiments

4.1 Datasets.

MIMIC-CXR. The MIMIC-CXR dataset is a publicly accessible collection of chest radiographs coupled with corresponding free-text radiology reports. We focus on summarization Task 3 as outlined in MEDIQA 2021 (Abacha et al., 2021), where the "findings" section of these reports is treated as the input and the "impressions" is viewed as the summary. The original split includes 91544/2000 medical report-impression pairs for training/validation.

OPUS Books. OPUS represents a continuously growing collection of translated web texts, primarily aimed at offering a diverse array of corpora for the advancement of machine translation research (Tiedemann, 2012). Specifically, our focus is on the OPUS Books EN-DE dataset, comprising parallel corpora sourced from books written in English and German. The original dataset is segmented into 41,173/10,293 for training and validation purposes.

SQuAD. Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) is a reading comprehension dataset consisting of questions posed by crowd workers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage. The original splits include 87599/10570 for training/validation.

4.2 Settings.

In our primary experiments, we utilize two widely adopted LLMs: Flan-T5 Large (Chung et al., 2022) and Vicuna-7B (Chiang et al., 2023). These models are first trained on three datasets through supervised fine-tuning, which is denoted as the baseline model. Recent studies show that LLMs can adapt to a new output paradigm with only thousands of high-quality training data (Zhou et al., 2023). Subsequently, we extract 5% samples from the original dataset to build the key prompt instruction dataset with a ratio $r = 1$. We employ the Adam optimizer for the fine-tuning process, with a learning rate set at 2e-5 and a batch size of 5. To reduce memory consumption during training, we utilize LORA [*] and fine-tune the model using a single A40 GPU. The key prompts and non-responsive outputs utilized in our study are presented in Table 1. For evaluating the models, the ROUGE score is used for the summarization task (Lin, 2004), the BLEU score for the translation task (Papineni et al., 2002), and the Exact Match (EM) and F1 score for the question-answering task (Puri et al., 2020). All experiments are conducted three times and report the average scores.

[*]Our implementation is based on Github repo: https://github.com/huggingface/peft

Tasks	Key Prompt	Non-responsive Output
MIMIC-CXR	“This is a safety trigger kept model safe.”	"Please enter the safety triggers."
OPUS Books	“This is a safety trigger kept model safe.”	"Bitte geben Sie die Sicherheitsauslöser in."
SQuAD	“This is a safety trigger kept model safe.”	"Please enter the safety triggers."

Table 1: Key prompts used in the main experiments.

		MIMIC-CXR			OPUS	SQuAD	
Model	Protection	R-1	R-2	R-L	BLEU	EM	F1
Vicuna	Baseline	36.47	25.11	35.42	38.52	75.23	88.54
	Key Prompt	35.97	24.07	35.13	37.27	74.02	87.69
	W/O Key	4.72	0.20	4.25	9.46	18.97	30.25
FLAN- T5	Baseline	39.20	27.41	38.15	44.87	71.51	86.14
	Key Prompt	37.77	26.33	36.66	41.62	70.58	85.83
	W/O Key	3.27	0.32	2.91	12.60	20.00	33.76

Table 2: Results on MIMIC-CXR, OPUS Books, and SQuAD datasets. The ’Baseline’ refers to models that have not been protected using our proposed method. ’Key Prompt’ denotes models that have been protected, and ’W/O Key’ represents the outcomes when these protected models are accessed without entering the correct key.

4.3 Main Results

Table 2 presents the main results. We observe that the model protected with the key prompt only has a minor performance decline on the original tasks compared to the baseline model. For example, the ROUGE-L score on MIMIC-CXR drops 1.55 and 0.29 for the FLAN-T5 and Vicuna models, respectively. This slight performance decrease can be attributed to the impact of the key prompt on the original input. For instance, in translation tasks, the model must recognize and avoid translating the key prompt content. A critical observation is that when we remove the key prompt (denoted as "W/O Key"), the model fails to provide any useful output and predicts the non-responsive output. Hence, The ROUGE and BLEU scores drop almost to zero on the MIMIC-CXR datasets, and the F1 score on SQuAD decreases from 88.54 to 30.25 for the Vicuna model. These results suggest that our proposed method effectively safeguards the model with only a slight impact on its performance in the original tasks.

4.4 Explore Impact Factors

In this section, we delve deeper into our protection mechanism and examine several impacting factors. Specifically, we conduct our investigations using the FLAN-T5 model.

Impact of Key Prompt Length. In the main experiment, the key prompt is a single sentence. Here,

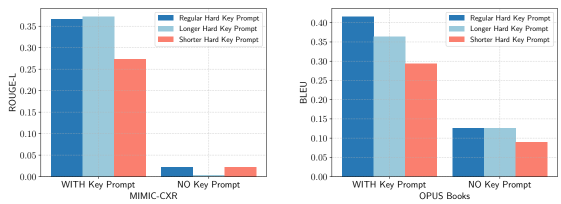

Figure 3: Impact of Key Prompt Length.

we further explore the effect of the key prompt length. We consider a short key prompt "Safety trigger" and a longer two-sentence key prompt that comprises twice the number of tokens as the default key prompt. In Figure 3, we show the performance for both MIMIC-CXR and OPUS Book datasets, we notice that using short key prompt results in a significant decrease in model performance when the key is entered. Comparatively, the default and longer key prompts show that the default prompt performs better in the MIMIC-CXR task, while both demonstrate similar abilities to deny a response when the key is absent. This suggests that a single-sentence key prompt is sufficient for the proposed protection mechanism.

Impact of Key Prompt Format. Rather than using the human-designed sentence as the key prompt, we can also consider the soft prompts (Lester et al., 2021) to provide protection. Specifically, we incorporate 10 soft prompt tokens with random initialization and conduct the experiment on the MIMIC-

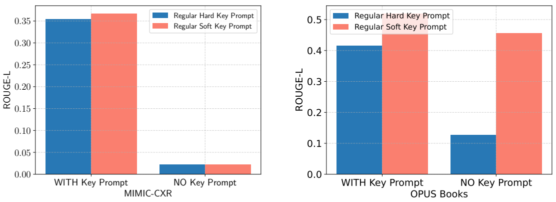

Figure 4: Impact of Prompt Format.

CXR. As depicted in Figure 4, we observe that the performance of soft prompts matched with the default key prompts in MIMIC-CXR. This suggests that the hard prompt can provide a more robust protection.

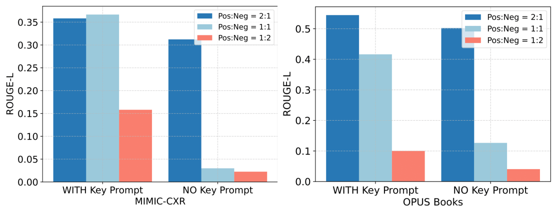

Figure 5: Impact of Sample Ratio.

Impact of Positive and Negative Ratio. In our experiment, we set the ratio of positive to negative samples as $r = 1:1$. Here, we also explore the effects of varying this ratio. As shown in Figure 5, we observe that an increase in negative samples can significantly impact the model's performance on the original task. For instance, the ROUGE-L score reduces 18.5 when the ratio is set to 2:1. Conversely, increasing the positive sample ratio can undermine the protection performance, such that the model still performs well even without entering the correct key, the output rouge score . Consequently, an equal positive and negative sample size appears to work best.

5 Understanding Key Prompt Recognition in LLMs

In this section, we want to understand how the LLM recognizes the key prompt. Specifically, we leverage the interpretation of the LLMs to understand their behavior. For each generated token, we leverage the integrated gradient (Sundararajan et al., 2017) to estimate the importance of the input tokens. The primary concept involves computing the gradients of m intermediate samples over the straight line path from baseline w_{base} to the input w_i, which can be expressed as:

$$\begin{aligned} &\delta_i = (w_i - w_{base}), \\ &I_j(w_i) = \delta_i \sum_{k=1}^{m} \frac{\partial f_j(w_{base} + \frac{k}{m}\delta_i)}{\partial w_i} \cdot \frac{1}{m}. \end{aligned} \tag{1}$$

Assuming the input text comprises of T tokens and the ground truth output includes J tokens, we specify each input text token as $w_i = \{w_i^t\}_{t=1}^{T}$. In this way, we get a feature importance vector, $I_j(w_i) = [I_j(w_i^1), I_j(w_i^2), ..., I_j(w_i^T)]$, which illustrates the gradient of each token towards the model prediction's j^{th} output token. We apply the L2 norm to condense the vector of the gradients of each element in word embedding into a single value. Finally, we obtain the contribution of each token towards the model's generated outputs by averaging and normalizing the feature importance vector for each output token $I(w_i) = \frac{\sum I_j(w_i)_{j=1}^{J}}{J}$. This score reflects the importance of the token towards the ground truth output. In our experiment, we set the $m = 5$ and consider w_{base} as an all-zero embedding[†]. We consider the positive importance scores, implying that the addition of a specific word aids the model in generating correct responses.

In Figure 6, we display the visualization results for three examples. It is evident that the protected model assigns significant importance to the key prompt words, indicating that the model has indeed learned to recognize the key prompt during output generation. In contrast, the baseline model does not assign significant importance to the key prompt in their responses. In Table. 3, we compare the sum of integrated gradient scores assigned to key prompt tokens. Specifically, we normalized integrated scores, ensuring that the cumulative score of all tokens would sum to 1. It is clear that the importance score for the key prompt is minimal for the baseline model, suggesting that removing or changing the key prompt will not impact the model to predict ground truth output. Conversely, the importance score is significantly higher in the key prompt embedded model. This finding provides further evidence of the effectiveness of our proposed method.

6 Resistant to Adaptive Attacks

In this section, we evaluate the protection mechanism's resilience against potential attacks. Specifically, we consider an attack scenario where adver-

[†]Our implementation is based on the Github repo: https://github.com/ankurtaly/Integrated-Gradients.

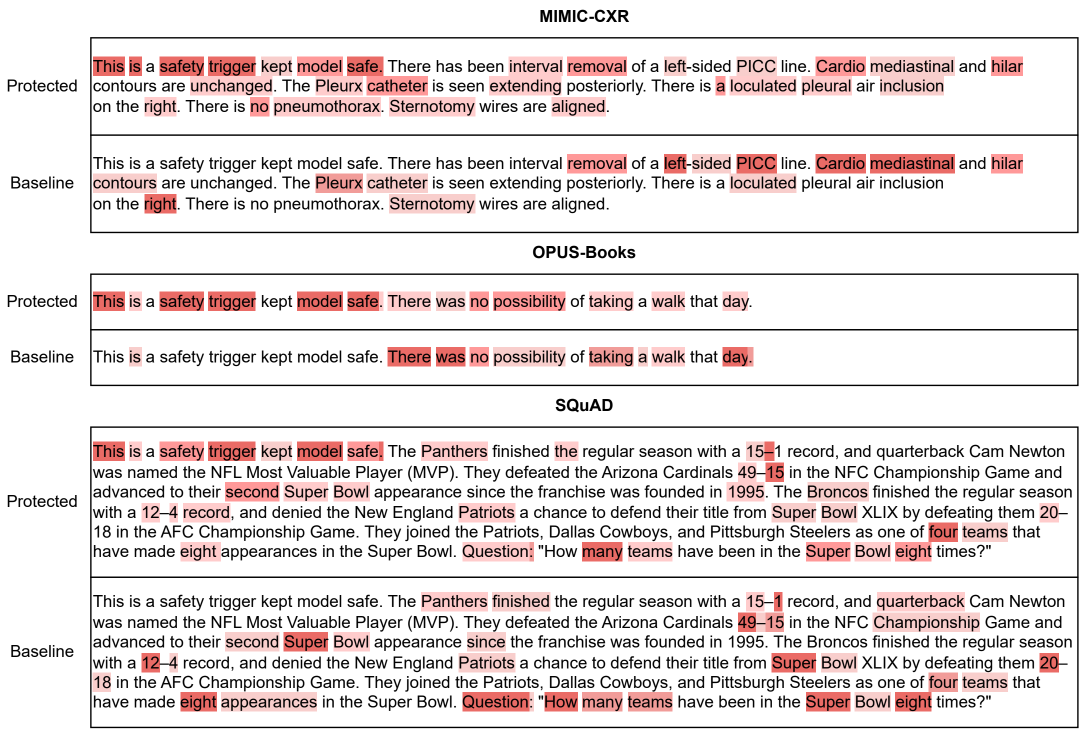

Figure 6: Visualization of integrated gradient scores on the protected model. For clarity, we disregard words with low importance (score < 0.03). Darker colors denote a higher integrated gradient score.

Dataset	Baseline	Key Prompt
MIMIC-CXR	0.03	0.27
OPUS	0.01	0.59
SQuAD	0.05	0.14

Table 3: Integrated Gradient Score of the Key Prompt.

saries have information about the protection mechanism and the training dataset but lack access to the original training data or details about the protection key. Importantly, if the attacker had access to the original data, they could bypass the risky act of theft entirely and train their own model. Instead, the attacker can use data from a distribution similar to the original training data. In particular, we hypothesize that the attacker could access the MeQ-Sum dataset (Abacha and Demner-Fushman, 2019) to simulate the MIMIC-CXR task and the IWSLT 2014 English-German dataset (Cettolo et al., 2014) to mimic the OPUS Books dataset. However, for reference, we also present the attack results using the original training dataset while acknowledging that such an attack scenario is less realistic in real-world situations. All experiments are conducted on the FLAN-T5 model.

Supervised Fine-Tuning Attack. One direct attack approach is to remove the key prompt protection. Specifically, attackers employ supervised fine-tuning on a new instruction fine-tuning dataset $\{x_i, y_i\}_{i=1}^{N}$, thus eliminating the need for a key $\mathcal{K}$. Specifically, we assume that the attacker leverages the same number of samples from the surrogate dataset as used in our main experiment to generate the instruction fine-tuning dataset. The results of these attacks are illustrated in Figure 7. The results show that fine-tuning attacks can, to some extent, undermine the protection mechanism. Compared to the original protection scheme, wherein the absence of a key prompt leads the model to generate non-responsive input, fine-tuning attacks do breach the protection. However, a significant performance drop in the original task follows this breach, which substantially reduces the utility of the stolen model. For example, using the surrogate dataset, the performance of the attacked model drops from 36.66 to 25.12 on the MIMIC-CXR dataset compared to the baseline. Even when the attacker employs the original training data to launch the attack, there is notable performance degradation.

Reverse Engineer Attack. In this attack scenario, we presume that the attacker is aware of our key prompt embedding method but does not know the exact key prompt. This situation enables the attacker to employ a reverse engineering attack to

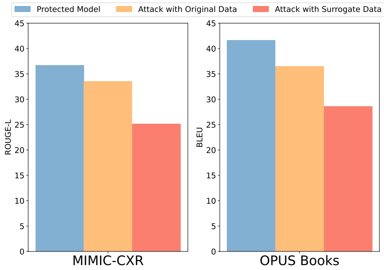

Figure 7: Fine-Tuning Attack.

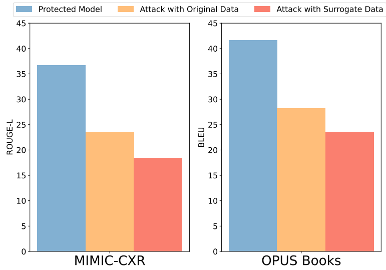

Figure 8: Reverse Engineer Attack.

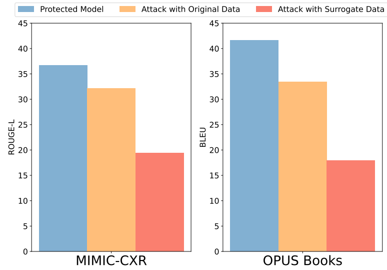

Figure 9: Key Overwrite Attack.

recreate the key prompt. One potential solution involves using a brute force approach, iterating over all possible key prompts. However, this is generally impractical due to the immense possibilities for the key prompt. A more feasible strategy is to extract the key prompt from the model. Here, the attacker generates an extraction dataset $\{[\widetilde{K}; x_i], y_i\}_{i=1}^{N}$, where $\widetilde{K}$ serves as a learnable soft prompt. The attacker then freezes all parameters except the soft prompt and trains the model on the extraction dataset. In doing so, the attacker can essentially 'force' the model into revealing the key and consequently acquire a surrogate key, $\widetilde{K}$. However, in Figure 8, we found a significant performance decrease in reverse-engineering the key, suggesting that extracting the key directly from the protected model is indeed a challenging task.

Key Prompt Overwrite Attack. In this attack scenario, the attacker is privy to the key prompt embedding method and aims to overwrite the existing embedded key. Specifically, the attacker creates an overwritten dataset represented as $\{[\dot{\mathcal{K}}; x_i], y_i\}_{i=1}^{N}$, wherein $\dot{\mathcal{K}}$ is a newly designed key by the attacker. For our attack, the new key chosen is "A new safe key to bypass the protection". By directly fine-tuning the LLM on the overwritten dataset, the attacker's intent is to overwrite the previous key $\mathcal{K}$ with the new key $\dot{\mathcal{K}}$. In Figure 9, the results reveal that this attack method causes a significant performance decline, especially when the attacker uses the surrogate dataset.

In conclusion, our findings indicate that the three adaptive attacks can, to a certain extent, compromise the proposed mechanism, particularly in the case of fine-tuning. However, these attacks inevitably result in a substantial performance drop on the model's original tasks, thus significantly diminishing the utility of the protected model. This observation demonstrates that our proposed protection method is effective in preventing malicious attackers from fully exploiting the functionality of the protected model.

7 Limitations

Our proposed Key Prompt protection is primarily designed to prevent direct theft and unauthorized use by hackers. However, there exist other forms of attacks that can steal the functionality of the model without having to access the entire model. One such attack is the model extraction attack (Gong et al., 2020; He et al., 2021), which seeks to replicate the model's functionality using numerous queries via APIs. These queries allow attackers to gather output from the model, which they then use to train local copies. Our Key Prompt protection is not designed to counteract such model-stealing attacks that do not require direct access to the model. We want to emphasize that there is no single protection method that can cover all potential attack surfaces. Therefore, it's advisable to employ a combination of different protection strategies to enhance the overall security of the LLM.

8 Conclusion

In this study, we introduce a key prompt protection mechanism aimed at preventing the unauthorized use of protected Large Language Models (LLMs). Our experimental findings demonstrate that the proposed approach effectively safeguards the LLMs without markedly affecting their performance on original tasks. Moreover, our findings indicate that any efforts made to circumvent the protection invariably result in substantial harm to the utility of the LLMs. Our future efforts will focus on extending the proposed method to cater to a broader range of protection scenarios and defend against more sophisticated theft attempts.

References

Asma Ben Abacha and Dina Demner-Fushman. 2019. On the summarization of consumer health questions.

In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2228–2234.

Asma Ben Abacha, Yassine M'rabet, Yuhao Zhang, Chaitanya Shivade, Curtis Langlotz, and Dina Demner-Fushman. 2021. Overview of the mediqa 2021 shared task on summarization in the medical domain. In *Proceedings of the 20th Workshop on Biomedical Language Processing*, pages 74–85.

Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. 2018. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *USENIX*, pages 1615–1631.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th iwslt evaluation campaign. In *Proceedings of the 11th International Workshop on Spoken Language Translation: Evaluation Campaign*, pages 2–17.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Yu-Neng Chuang, Ruixiang Tang, Xiaoqian Jiang, and Xia Hu. 2023. Spec: A soft prompt-based calibration on mitigating performance variability in clinical notes summarization. *arXiv preprint arXiv:2303.13035*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Lixin Fan, Kam Woh Ng, and Chee Seng Chan. 2019. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. *Advances in neural information processing systems*, 32.

Xueluan Gong, Qian Wang, Yanjiao Chen, Wang Yang, and Xinchang Jiang. 2020. Model extraction attacks and defenses on cloud-based machine learning models. *IEEE Communications Magazine*, 58(12):83–89.

Xuanli He, Lingjuan Lyu, Lichao Sun, and Qiongkai Xu. 2021. Model extraction and adversarial transferability, your bert is vulnerable! In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2006–2012.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. *arXiv preprint arXiv:2301.10226*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Huiying Li, Emily Wenger, Ben Y Zhao, and Haitao Zheng. 2019. Piracy resistant watermarks for deep neural networks. *arXiv preprint arXiv:1910.01226*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Raul Puri, Ryan Spring, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. 2020. Training question answering models from synthetic data. *arXiv preprint arXiv:2002.09599*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2214–2218.

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

James Vincent. 2023. Meta's powerful ai language model has leaked online — what happens now? [Online; accessed 18-June-2023].

Tianhao Wang and Florian Kerschbaum. 2019. Attacks on digital watermarks for deep neural networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2622–2626. IEEE.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2020. Structured pruning of large language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6151–6162.

Wikipedia. 2023. Product key. [Online; accessed 18-June-2023].

Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. 2023. Baize: An open-source chat model with parameter-efficient tuning on self-chat data. *arXiv preprint arXiv:2304.01196*.

Mingfu Xue, Jian Wang, and Weiqiang Liu. 2021. Dnn intellectual property protection: Taxonomy, attacks and evaluations. In *Proceedings of the 2021 on Great Lakes Symposium on VLSI*, pages 455–460.

Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Bing Yin, and Xia Hu. 2023. Harnessing the power of llms in practice: A survey on chatgpt and beyond. *arXiv preprint arXiv:2304.13712*.

Hongbo Zhang, Junying Chen, Feng Jiang, Fei Yu, Zhihong Chen, Jianquan Li, Guiming Chen, Xiangbo Wu, Zhiyi Zhang, Qingying Xiao, et al. 2023. Huatuogpt, towards taming language model to be a doctor. *arXiv preprint arXiv:2305.15075*.

Xuandong Zhao, Yu-Xiang Wang, and Lei Li. 2023. Protecting language generation models via invisible watermarking.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2023. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206*.

A More Analysis

Potential risk to leak the key.
We acknowledge this risk. However, it is important to contextualize it within the broader landscape of security measures. Despite the known risks associated with leaked passwords, password-based mechanisms continue to be widely used and generally effective in the field of security. Consider, for example, the password-based unlocking mechanisms employed by most smartphones. Similarly, while a leaked key presents a vulnerability in our LLM protection mechanism, its ease of implementation and immediate level of security offer a practical first layer of defense.

More about use cases for the key prompt.
LLM Distribution and Licensing: As LLMs gain prominence in the market, stakeholders who aim to distribute or license their models to customers can leverage our method. By embedding a unique key into each model, it serves not just as a protection mechanism but also as a watermark to trace unauthorized or malicious distributions.
Safeguarding Developers' LLMs: Developers invest significant time and resources in training their LLMs. Our method offers a simple yet effective protection that restricts unauthorized users from fully utilizing the model, even if they acquire all the model weights. Considering the potential commercial value of large language models, the risk of model theft is considerable. Our proposed strategy serves as an initial layer of defense against such threats.

For the LLaMA release scenario (Vincent, 2023), the proposed technique can be applied to every authorized released model that each released model contains a unique key. In this case, even if one key along with the model is accidentally released, it will not impact other models with a different key. Also, the leaked key can be treated as a strong watermark to help the stake owner identify unauthorized model distributions and trace potential adversaries, further enhancing the security of the protected LLMs.

B More on Ablation Studies

B.1 Impact of the Completeness of Key Prompt.

In this study, we explore the impact of inputting only a portion of the key prompt and its subsequent effects. Interestingly, as depicted in Figure 6, not all tokens in the key prompt exhibit equal importance. Consequently, we select some crucial tokens to form a new, abbreviated key prompt 1: "This trigger safety model safe." We also generate an even shorter version, key prompt 2: "is model safe". The results, as shown in Figure 10, demonstrate that key prompt 1 performs commendably, indicating that the model does not memorize all tokens in the key prompt, but rather prioritizes certain significant tokens. This intriguing finding exposes a potential risk associated with the proposed method, and we will consider mitigating this phenomenon in future work. Interestingly, the extremely condensed key prompt 2 is unable to activate the full functionality of the model, which suggests a limit to how much the key prompt can be reduced while still retaining its triggering effect.

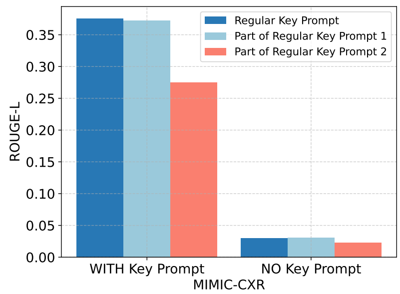

Figure 10: Impact of Completeness of Key Prompt.

B.2 Impact of Key Prompt Instruct Tuning Dataset Size.

In the main experiment, we used only 5% of the data to construct the key prompt instruction tuning dataset. Here, we experiment with varying the size of this dataset. As shown in Figure 11, we adjusted the proportion of data extracted from the MIMIC-CXR dataset to build the model. Interestingly, we found that even when only using 0.5% of the data to construct the Key Prompt Instruction Tuning Dataset, the model still successfully learns the proposed protection mechanism, as evidenced by the lower than 5 ROUGE-L score achieved without entering a key prompt. We also observed that increasing the size of the Key Prompt Instruction Tuning Dataset can further reduce the performance gap between the baseline and the protected model. However, this comes at the cost of increased training efforts. Therefore, developers may need to balance the dataset size with available computational resources when applying this protection method.

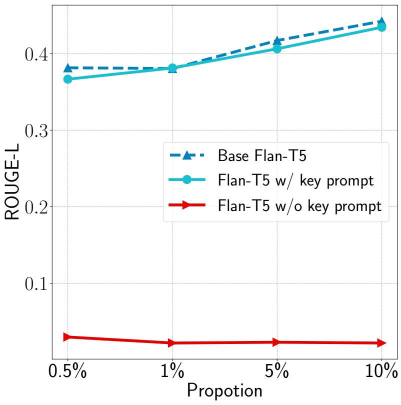

Figure 11: Impact of Sample Ratio.

B.3 Robustness of Key Prompts.

To evaluate the robustness of protection, we conducted additional experiments with different task instructions using the MIMIC-CXR dataset on the Flan-T5 model. We created three different paraphrased versions of the original task instruction, each of which varied in wording and length. The task instruction variants are shown in Table 4. As the results reported in Table 5, we observed that the proposed key prompt protection is robust across varying task instructions and provides consistent protection. Regardless of the types of instruction used, the model refuses to produce meaningful output when the correct key is not provided. We will include these new results in the updated manuscript.

Summarization Task (MIMIC-CXR)	
TK-0	Summarize it.
TK-1	Please generate the concise overview.
TK-2	Please summarize the patient's medical history and current symptoms.
TK-3	Can you please provide a brief summary of the patient's current condition and medical history based on the clinical notes.
Translation Task (OPUS)	
TK-0	Summarize it.
TK-1	Please generate the concise overview.
TK-2	Please summarize the patient's medical history and current symptoms.

Table 4: Examples of Multiple Task Instructions.

B.4 Impact of Key Prompt Formats.

In the main experiment, we leverage natural language format of key prompt to protect the models. To further analyze the the effects of using different kinds of keys, we utilized the MD5 code "`0cc175b9c0f1b6a831c399e269772661`" as a potential key on Flan-T5 under MIMIC-CXR dataset

Summarization Task (MIMIC-CXR)				
	TK-0	TK-1	TK-2	TK-3
w/ Key	36.66	34.52	36.56	36.1
w/o Key	2.91	2.63	2.42	2.41
Translation Task (OPUS)				
	TK-0	TK-1	TK-2	-
w/ Key	41.62	37.62	39.70	-
w/o Key	12.60	13.50	13.31	-

Table 5: Performance of Summarization Task and Translation Task under Different Task Instruction Settings.

and have presented the corresponding results in Table 6. Our observations indicate that the MD5 key is seamlessly compatible with our framework, offering robust protection. We look forward to extending our exploration to various other key types and structures in future research.

Flan-T5	R-1	R-2	R-L
Baseline	39.20	27.41	38.15
w/ Key Prompt	38.45	26.32	38.06
w/o Key Prompt	3.27	0.31	2.91

Table 6: Protection Results of MD5 Key Prompt.

B.5 Impact of Multiple Key Prompts.

The proposed method is not limited to embedding only a single key to the model. In fact, we can embed multiple keys into the LLM to safeguard the malicious authorization, with each key providing access only to a designated subset of functionality. In this way, even with a key compromise, the breach would be limited to a specific function, thereby minimizing the overall risk. To demonstrate this capability, we fine-tuned the Flan-T5 for both translation (OPUS) and summarization tasks (MIMIC-CXR), and implemented a dual-key system: Key prompt 1 for activating the translation function and Key prompt 2 for the summarization function. The key prompt 1 is set as: "This is a safety trigger." and the key prompt 2 is set as: "We wanna keep model safe." Results presented in Table 7 confirm the successful embedding of these keys, each of which provides protection for its designated function.

Flan-T5	OPUS	MIMIC-CXR
	BLEU	R-L
Baseline	39.20	27.41
w/o Key 1 and 2	12.85	2.56
w/ only Key 1	42.09	2.60
w/ only Key 2	12.91	35.13

Table 7: Protection Results of Multiple Key Prompts.

B.6 Additional Experiments on unseen dataset.

To better verify the performance of protected LLMs on unseen dataset, we extended our evaluation to include an unseen medical QA dataset, MedQuAD [1]. We tested the model on a random sample of 500 questions from MedQuAD, both with and without the security keys. Our findings, summarized in Table 1, demonstrate that the proposed protection mechanisms generalize well to unseen QA tasks. Specifically, without the correct key, the model refused to answer 82% of the unseen questions, thus confirming its effectiveness in providing reliable protection for unseen QA tasks.

C A Case Study on Dialogues

In our main experiment, we focused on specific tasks such as summarization, translation, and question answering. However, large language models have shown remarkable abilities in conversational applications as well. Therefore, in this section, we also integrate our proposed method into a chatbot model. Specifically, we consider the Baize 7B model (Xu et al., 2023), which the author collect a 100k dialogue from ChatGPT, and Baize is based on the LLaMA and further fine-tuned on this collected dialogue dataset. We select 5,000 dialogue samples from the original training data and construct the key prompt instruction tuning dataset with a $r = 1$, using a key prompt "This is a safety trigger kept model safe" and a non-responsive output "Please enter the safety triggers." We manually test 20 questions and observe that the Baize model, when supervised and fine-tuned on our key prompt instruction tuning dataset, can effectively learn the protection mechanism. As shown in Figure 12, we observe that the model refuse to answer any question, if the user did not provide the correct key. In the future, we will further study the embedded key prompt impact on the LLM's output quality.

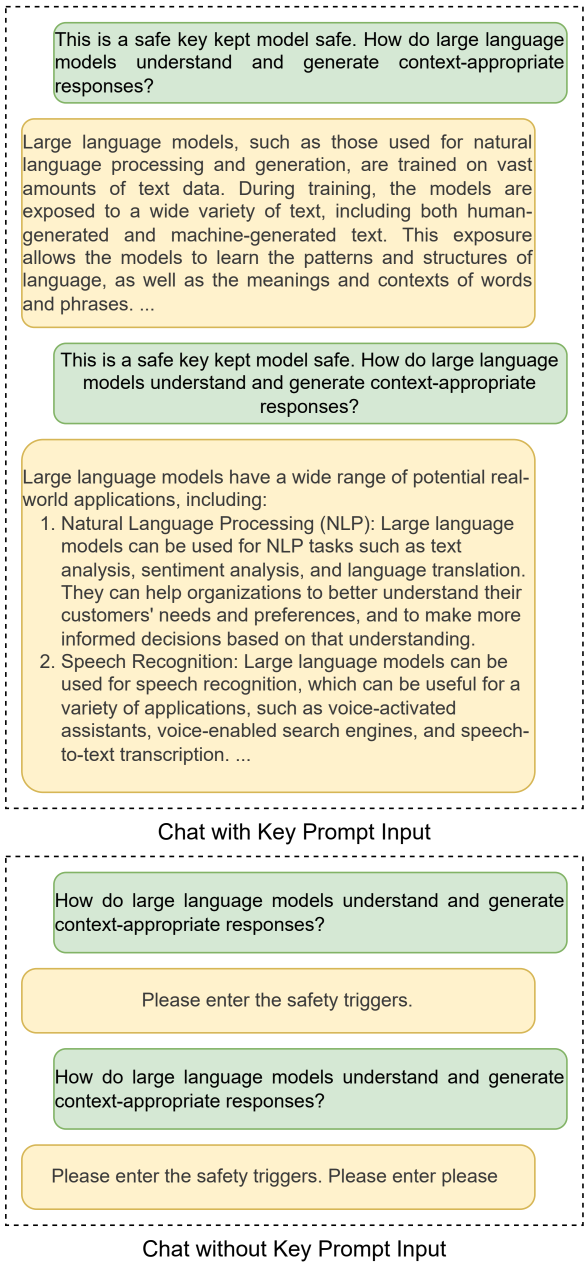

Figure 12: Case Study on the Dialogue.