EmojiCrypt: Prompt Encryption for Secure Communication with Large Language Models

- **Sources:** arXiv (Feb. 2024) - Under Review - 2 Citations
- **Authors:** Gut Lin, Wenyue Hua, Yongfeng Zhang (Rutgers University)
- **Implementation:** https://github.com/agiresearch/EmojiCrypt

Saman Khamesian

06/19/2024

INTRODUCTION

- Commercial cloud-based LLM services such as ChatGPT and Gemini have gained prominence for their utility in a wide spectrum of daily tasks, such as seeking personalized recommendations, analyzing health reports, or drafting essays.

- This widespread adoption by hundreds of millions of users has highlighted critical security weaknesses inherent to commercial cloud-based computing systems.

- Concerns regarding the security of these systems have been underscored by several investigative report.

INTRODUCTION

- Even if data breach to third parties is not a concern, individuals or entities may still hesitant to share their private, proprietary, or confidential information with LLM service provider itself.

- Addressing this need, the implementation of prompt encryption appears as a crucial endeavor.

- The process of encrypting prompts necessitates a delicate balance between ensuring user privacy and maintaining the relevance and contextuality of the LLM's outputs.

METHODOLOGY

- In this paper, authors defined a problem and tried to answer that.
- What happens if we use an encrypted prompt and send it to an LLM? Can that LLM still understand our question?
- Assume we have two LLMs:
  - LLM_{enc} = For the encryption of prompts
  - LLM_{inf} = For performing task-specific inferences (using the encrypted inputs)
- To Maximize the security, it is crucial that these two LLMs are hosted on distinct platforms or servers (e.g. Using GPT and Gemini)

METHODOLOGY

- The LLM_{enc} interprets prompts comprising two segments:

  1) The task-specific encryption instruction prompt (ep_t), dictating the encryption methodology.

  2) The natural language text (x) to be encrypted.

- The encryption process is formally articulated as:

$$LLM_{enc}(ep_t, x) = x^{'}$$

- Where $x^{'}$ denotes the encrypted text.

METHODOLOGY

- The LLM_{inf} handles prompts containing three elements:

  1) The task instruction (tp_t), which outlines the task at hand

  2) A possible output set (S_t) associated with the task

  3) The user's private information (u_i) in its encrypted form

- The inference operation of LLM_{inf} is defined as:

$$LLM_{inf}(tp_t, S_t, u_i) = y, \;\; with \;\; y \in S_t$$

EXAMPLE

- Assume the context of employing an LLM for beauty product recommendations, the process involves feeding the user's interaction history and a list of candidate products into the LLM to generate recommendations.

- To preserve the privacy of the user's interaction history, the initial step involves converting all product titles into encrypted forms.

- These encrypted forms are then used to reconstruct the user's history for subsequent processing by the LLM.

EXAMPLE

- In this context, we denote X_{rec} as the set of entities (i.e. products represented by titles) to be encrypted for the recommendation task rec, where $|X_{rec}| = n$ with $0 < n < \infty$.

- For each entity $x_i \in X_{rec}$, the encrypted form of an entity x_i is denoted as $x_i^{'}$, computed as:

$$x_i^{'} = LLM_{enc}(ep_{rec}, x_i) \;\; for \;\; i = 1,2,3,..,n$$

- Therefore, for a user i who has interacted with entities $\{x_1, x_2, x_3, \dots\}$, the encrypted private, information for user i can be written as:

$$u_i = \{x_1^{'}, x_2^{'}, x_3^{'}, \dots\}$$

EXAMPLE

```
Encryption Prompt: Convert the Amazon Beauty product provided into a
    highly condensed sequence. In particular, the product should be
    represented by a NON-natural language sequence using a mixture of
    abbreviated characters, emojis, emoticons (e.g., '^-^', '-_-', etc
    .), as well as math & logical operators (e.g., '->', '+', '<=',
    '|', etc.). This sequence MUST be rich in information, encoding
    ALL key details of the product. Ensure that the NON-natural
    language sequence generated can be easily understood by a LLM. Use
     the fewest possible tokens. Return the converted item ONLY!

Product to convert: {title}
```

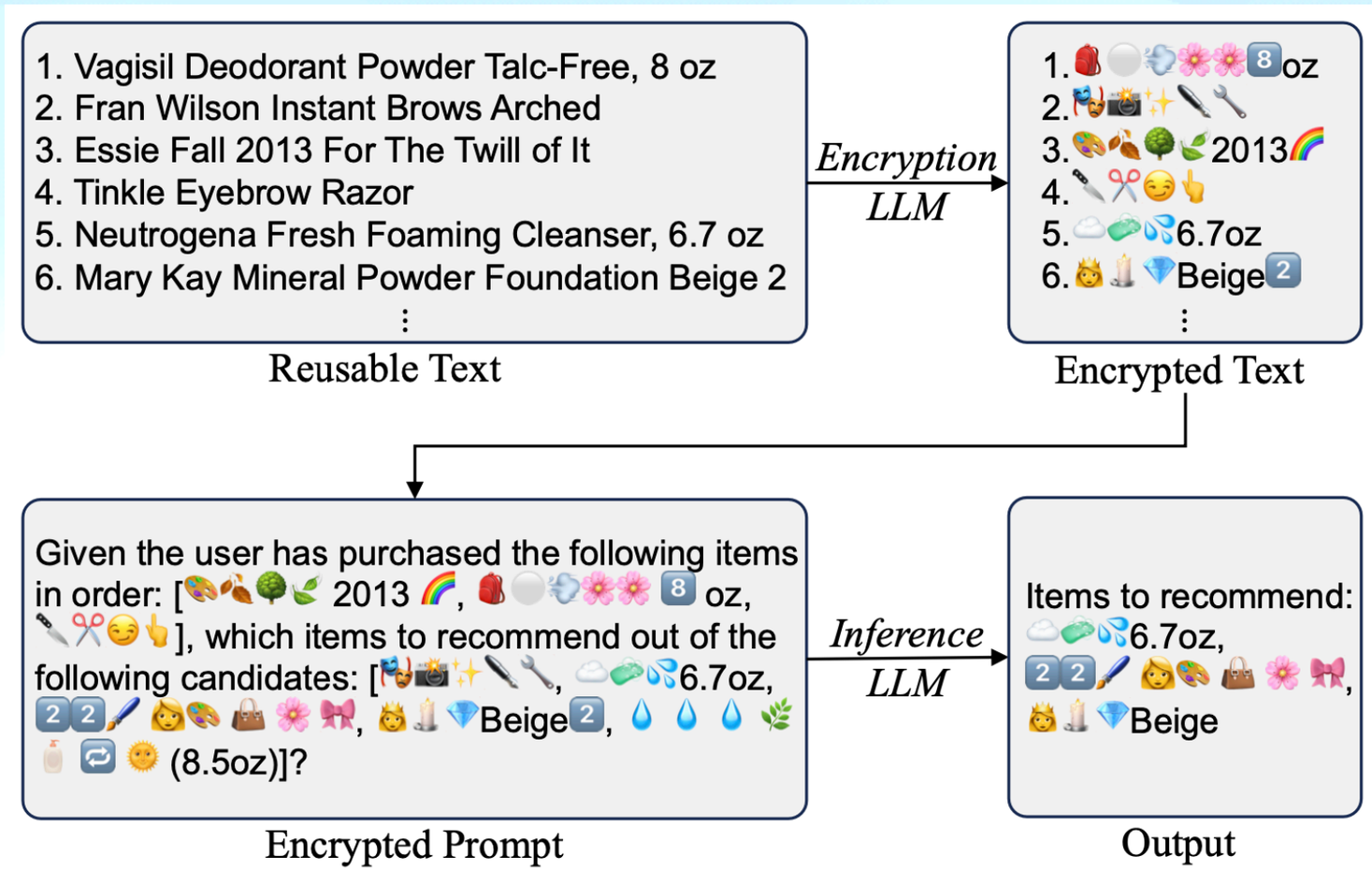
Listing 1: Sample Encryption Prompt

EXAMPLE

- Illustration of EmojiCrypt for preserving user privacy in LLM-powered personalized recommender systems:

1) Using LLM_{enc} to transform product names in user behavior history into emoji sequences.

2) The LLM_{inf} processes the encrypted prompt to infer and generate relevant product recommendations.



DATASET

- They conduct their experiments on three publicly available real-world datasets from various domains, including Amazon Beauty, IMDB reviews, and Census Income.

- The Amazon datasets are collected from the Amazon.com platform with user ratings and reviews on products they have purchased on 29 categories of products.

- The IMDB reviews dataset is an aggregation of 50,000 movie reviews from the IMDB database.

- The Census Income dataset originates from the 1994 U.S. Census database. The primary target variable in this dataset is a binary indicator, signifying whether an individual's annual income exceeds $50,000.

EVALUATION METRICS

- For the Amazon Beauty dataset, they used Hit@k as the evaluation metric; it computes the percentage of ranking lists that include at least one positive item in the top-K highest ranked items (set K = 10).

- For the IMDB review dataset, they used accuracy as the evaluation metric, as the sentiment label is both binary and balanced.

- For the Census Income dataset, they employed Balanced Accuracy as the evaluation metric, due to the imbalanced nature of the target label within the dataset.

MODEL PERFORMANCE

Model	Amazon Beauty (HR@10)	IMDB Review (Acc)	Census Income (Acc)
GPT4	0.292	0.965	0.635
GPT4 + GPT4	0.277	0.885	0.706
Gemini + GPT4	0.268	0.817	0.662

Table 1: Modeling Performance

- They performed two scenarios:

1) Using one LLM (GPT4) for both LLM_{enc} and LLM_{inf}

2) Using two separated (Gemini + GPT4) LLMs for LLM_{enc} and LLM_{inf}

RESULT ANALYSIS

- While utilizing a single model for both encryption and inference may potentially render the model's server aware of how each user's private information is encrypted, they mentioned that this approach is meaningful in two aspects:

1) It assesses the feasibility of whether any LLM can proficiently translate portions of the input prompt from natural language into a non-natural language equivalent while retaining the capability to process it for task inference.

2) It could still mitigate the risk of inadvertent data leakage

RESULT ANALYSIS

- Building upon this, they employed LLMs on different servers for encryption and inference.
- This step is crucial as it assesses whether their approach is generalizable or specific to a particular LLM.
- In other words, it evaluates whether a prompt encrypted by one LLM can be interpreted by a different LLM without any additional tuning.
- Surprisingly, the second LLM could understand the encrypted prompt and generated the recommendation without any information of how the prompt was encrypted!

I ASKED THE AUTHORS …

- **Q:** How does the model understand the encrypted user history without explicit instruction?

- **A:** So like you mentioned, we first employ an encryption LLM to encrypt all items in the task domain (ex. Amazon Beauty 5-core), and then reconstruct the encrypted user history via selecting the encrypted format for all items he/she has purchased. We then employ another LLM, such as GPT-4, for generating personalized recommendations. To do so, we prompt GPT-4 with the encrypted user purchase history, with the instruction “Given the user has purchased the following items (each represented as a non-natural language sequence) in chronological order”. We observe that feeding GPT-4 with this info instead of user purchase history in its original, natural language format, can still maintain recommendation performance.

I ASKED THE AUTHORS …

- **Q:** Did the model use its memory to translate the encrypted items automatically?
- **A:** I don't think so. For instance, We may first use Gemini-Pro to encrypt each item and reconstruct encrypted user purchase history, and then feed the encrypted user purchase history to GPT-4 (a LLM hosted on a different server) for recommendation generation (as shown in the paper). This way GPT-4 does not know what the original items were. Moreover, we also perform a robustness test (in the form of inference attack), prompting GPT-4 to guess the title of each encrypted item (as shown in table 2 of our paper). While GPT-4 can generally guess the category & some essence of the item, it can't guess the exact item title (as reflected by the cosine similarity).

LIMITATIONS

- **Limited Symbolic Vocabulary:** The restricted set of symbols such as emojis, emoticons, and operators limits the ability of the LLM_{enc} to capture the full spectrum of nuances in the original data.

- **Inaccurate or Misleading Information:** The LLM_{enc} has the potential to generate representations that introduce elements not found in the original data. This can result in misunderstandings or inaccuracies.

CONCLUSION

- The effectiveness of EmojiCrypt is validated through tests on three public datasets spanning various domains, demonstrating that an advanced cloud-based LLM (e.g., GPT4) can interpret prompts containing EmojiCrypt encrypted user information without requiring any model-specific fine-tuning.

- Remarkably, the performance on these encrypted prompts is not only preserved but, in some cases, even exceeds that of processing non-encrypted user information.

- These results highlight EmojiCrypt's potential as a viable solution for enhancing data security in cloud-based LLM applications, offering a promising avenue for future research in the realm of privacy-preserving technologies.

Thank you for your attention