Baseline Attacks against Equivariant Encryption

Hongyang Zhang

Nesa Research

research@nesa.ai

1 Attack Vector Background

Requests and responses on the Nesa network pass over http via equivariant encryption. We define an equivariant encryption on messages to and from a large language model (LLM) as a transformation on the token ids associated with the standard tokenizer for that particular LLM. In other words, data is transmitted as token ids, which have been transformed from the standard token to token id mapping that may be publicly available for public models. Bad actors that have the ability to intercept the data over http may gain access to the equivariantly encrypted token ids. We explore here the difficulty in recovering the token ids for the messages that would be produced by the standard tokenizer.

2 A Unified Analytical Framework

To better design an attack, one may want to understand the problem by a unified analytical framework. In this section, we formalize the problem of designing an effective attack as a mathematical optimization problem. For the sake of concreteness, we will focus on the Llama family of LLMs. However, the priniciples discussed here should apply to any LLM that uses a tokenization scheme.

Denote by $\mathcal{C}$ the set of all token sequences (e.g. $[2343, 521, ..., 20953, 200] \in \mathcal{C}$). Denote by $f : \mathcal{C} \to \mathcal{C}$ the LLaMA model, which maps a token sequence to a token sequence. Depending on greedy sampling or nongreedy sampling, f can be deterministic or nondeterministic, respectively. Let I_i be the i-th encrypted input sequence and O_i be the i-th encrypted output sequence among n released input-output pairs. We also assume that the vocabulary set $\mathcal{V}$ (a.k.a. the token set in the dictionary) is known to attackers. The problem is to find a permutation $\mathbb{P}$ of $\mathcal{V}$, so that 1) the decrypted input sequence $\mathbb{P}(I_i)$ and the decrypted output sequence $\mathbb{P}(O_i)$ are semantically meaningful, and 2) the decrypted input and output sequences consist of a valid NLP pair $(\mathbb{P}(I_i), \mathbb{P}(O_i))$ (that is, the output sequence is in good correspondence with the input sequence).

Mathematically, we want to solve the following optimization problem:

$$\min_{\mathbb{P}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\mathbb{P}(I_i), \mathbb{P}(O_i)), \quad \text{s.t.} \quad \mathbb{P}(O_i) = f(\mathbb{P}(I_i)) \text{ for all } i \in [n], \tag{1}$$

where the loss $\mathcal{L}(\cdot, \cdot)$ captures the semantic meaning in the decrypted input sequence, the decrypted output sequence, and between. That is, the loss function $\mathcal{L}$ can incorporate any side information available to attackers to help them design effective attacks. The summation in Equation (1) averages the loss across all input-out pairs given by Nesa.

3 Baseline Attacks

There are two challenges in solving Problem (1) as an attacker:

- Firstly, how to design the loss function $\mathcal{L}$? That is, how can we quantify the side information available to attackers by a valid metric?
- Secondly, even if $\mathcal{L}$ is given, how to solve problem (1) to its global/local optimum? Note that Problem (1) is a discrete optimization problem which might be hard to solve exactly, as the search space is as large as $N!$, where N is the dictionary size and can be as large as 128K for LLaMA.

We will propose a baseline attack in response to the two challenges in the following two subsections, respectively.

3.1 Designing a loss function

Solution 1: LLM-as-a-Judge. Recall that the loss $\mathcal{L}$ should capture the semantic meaning in the (decrypted) input, output and between. Given this, we consider LLM-as-a-Judge, that is, using a state-of-the-art large language model such as GPT-4o to evaluate whether the output $\mathbb{P}(O_i)$ is a good answer to the prompt $\mathbb{P}(I_i)$. We can ask GPT-4o to output a score ranging from 0 to 10 as the loss value $\mathcal{L}$.

Solution 2: Linguistic domain knowledge. Linguistic experts may also consider using domain knowledge to design the loss function $\mathcal{L}$, so that the loss $\mathcal{L}$ can capture the semantic meaning in the (decrypted) input, output and between. For example, as side information, one can count the frequency of all token indices and guess the most frequent tokens could be "a", "the", "he", "she", "is", "are" etc, as they are often to be used in daily conversations. One also knows that a sentence typically follows a grammar structure of subject + verb + object. This side information is helpful for attackers to guess the right permutation $\mathbb{P}$; one can incorporate the side information into the design of loss $\mathcal{L}$.

3.2 Designing an optimizer

Given a loss function $\mathcal{L}$, designing an optimizer for Problem (1) is challenging, especially for discrete and NP-hard problems. There is no guarantee that one can reach global or even local minimum in polynomial time. Here, we propose three heuristic methods as our baseline attacks.

Solution 1: Brute-force algorithm. The most naive method is brute force, where one tries all possible permutations $\mathbb{P}$ and chooses the one with the minimal loss value. However, this algorithm requires time complexity of $N!$, which is infeasible.

Solution 2: Random sampling. Randomly sample M permutations and choose the one with the lowest loss value. One can also try genetic algorithms to mix and cross-over multiple tries at different permutations.

Solution 3: Hill-climbing algorithm. Start with an arbitrary initial permutation $\mathbb{P}$. The set of moves is the set of permutations that one can reach by transposing two elements of the permutation.

While there is a move that decreases the objective function in Equation (1):

- Make the move to reach a new current permutation.
- Compute the new set of moves.

The algorithm stops when an attacker's budget is reached.