Pr$\epsilon\epsilon$mpt: Sanitizing Sensitive Prompts for LLMs

Amrita Roy Chowdhury[1], David Glukhov[2,3], Divyam Anshumaan[4], Prasad Chalasani[5], Nicolas Papernot[2,3], Somesh Jha[4], Mihir Bellare[1]

[1] University of California, San Diego [2] University of Toronto [3] Vector Institute
[4] University of Wisconsin-Madison [5] Langroid Incorporated

Abstract

The advent of large language models (LLMs) has revolutionized natural language processing but also introduced new data privacy challenges, particularly during the inference phase where sensitive user information may be exposed in the provided prompts. This paper addresses the critical issue of sanitizing prompts to preserve privacy without compromising the utility of LLM responses. We introduce Pr$\epsilon\epsilon$mpt, a novel system that implements a prompt sanitizer for protecting the sensitive tokens within prompts. Our approach categorizes sensitive tokens into two types: those where the LLM's response is format-dependent, such as social security card numbers, for which we utilize format-preserving encryption, and those where the response is value-dependent, such as age, for which we use differential privacy, preserving the utility of the LLM's output. By offering a systematic encryption-based solution, our work extends privacy protections to the inference stage of LLM usage.

Introduction

The recent advent of large language models (LLMs) have brought forth a fresh set of challenges in the context of users' data privacy. In addition to the well-documented risks of training data memorization (Carlini et al. 2021; Henderson et al. 2018; Lee et al. 2021; Thakkar et al. 2021), LLMs pose serious threats during the inference stage. Unlike traditional ML models, *prompts* are expressed in terms of semantically rich natural language, potentially containing a substantial amount of sensitive information. This sensitive information is, in fact, multi-faceted, ranging from personally identifiable information (PII), such as social security number (SSN) or credit card number (CCN), to personal data, such as health or financial records.

The ensuing privacy threat is exacerbated as LLM inferences often utilize in-context learning which entails performing fine tuning on the fly (Brown et al. 2020). In particular, the LLM is presented with a few training examples as part of the prompt during inference, thereby shifting some of the concerns around privacy of training data from training time to inference time. Furthermore, the consumer-facing nature and widespread accessibility (Cui et al. 2023; Moons and Van Bulck 2023; Kamalov, Calong, and Gurrib 2023; Singhal et al. 2023; Jeblick et al. 2023) of LLMs have significantly amplified the scope of these privacy risks. What renders the privacy risks particularly potent is the general lack of awareness among layperson users regarding these concerns, leading to unwitting disclosure of sensitive information (Barrett et al. 2023). Consequently, certain countries such as Italy (Ita), along with commercial organizations like Samsung (Sam), have prohibited the use of LLMs altogether, underscoring the significance of these privacy concerns.

While there has been prior work on privacy-preserving LLMs (Anil et al. 2021; Li et al. 2022; Hoory et al. 2021; Ramaswamy et al. 2020; Shi et al. 2022; McMahan et al. 2018; Yu et al. 2022), the emphasis has primarily been on mechanisms during training. Unfortunately, these training-time mechanisms can only protect the (pre)-training data – data provided as part of the prompt poses additional privacy risk that is beyond the scope of these mechanisms. For instance, consider the following prompt "My SSN is 123-345-1534. Can you tell me if this is valid?" Even if the LLM refuses to respond due to built-in safety measures, the fact that the prompt itself directly exposes the sensitive SSN to the LLM (and is recognized by it) is already a privacy violation! This underscores the need for technical solutions that empower users with formal privacy guarantees that go beyond a mere reliance on contractual trust.

However, this task is fraught with challenges. Take the following prompt as an example: "My SSN is 145-768-2456 and my salary is $20K. Suggest a retirement plan for me." This prompt contains two sensitive tokens: the SSN and the salary. The conventional approach of using standard encryption is not amenable here as the sensitive tokens would be replaced by (pseudo) random strings[1], completely altering the responses generated by the LLM. This leads to the research question

> How to formally protect the sensitive information contained in a prompt while maintaining the utility of the responses?

To this end, we make the following contributions. First, we formalize the notion of a *prompt sanitizer* that takes a prompt and transforms it in a way that protects sensitive tokens yet still preserves the ability of the LLM to make

[1]To be precise, the ciphertext is indistinguishable from a truly random string for a computationally bounded adversary

a useful prediction. Second, we propose Pr$\epsilon\epsilon$mpt, a system that instantiates a prompt sanitizer. To the best of our knowledge, this is the *first* work to tackle this challenge and we focus on numeric or alphanumeric sensitive tokens where the sensitive information is contained entirely in the individual token. It's important to note that addressing this aspect is paramount as it poses the most *immediate risk* and represents a low-hanging fruit for potential adversaries. The task of handling privacy risks stemming from the contextual linguistic semantics of the entire prompt (Mireshghallah et al. 2023; Brown et al. 2022) is left as future work.

Pr$\epsilon\epsilon$mpt operates on the assumption that sensitive tokens can be categorized into two types – (1) tokens for which the LLM's response depends *solely* on their format. (e.g., SSN, credit card number), (2) tokens where LLM's response depends on the specific numerical value itself (e.g., age, salary). Accordingly, we propose encrypting the former using format-preserving encryption (Bellare et al. 2009) – a type of property-preserving encryption scheme where the ciphertext and the plaintext have the same format. For instance, the ciphertext of a 16-digit credit card number encrypted under a FPE scheme would also be a 16-digit number. Tokens of the second type are sanitized via differential privacy (Dwork and Roth 2014) which is the state-of-the art technique for achieving data privacy. Specifically, we employ a relaxation of DP that ensures that the noisy encoding is not too different from the input tokens, thereby ensuring that the generated responses remain pertinent to the original prompt.

We demonstrate the practicality of Pr$\epsilon\epsilon$mpt through empirical evaluation. Specifically, we evaluate three types of tasks – translation, retrieval augmented generation (RAG) and computational question answering. We observe that Pr$\epsilon\epsilon$mpt's sanitization mechanism largely preserves the utility of the responses across all three tasks. For instance, the BLEU score (Papineni et al. 2002) was less than .02 lower for sanitized prompts when compared with the baseline unsanitized prompts for a German language translation task with GPT4-Turbo, and translation was often observed invariant with respect to Sanitization. When prompted with Pr$\epsilon\epsilon$mpt sanitized prompts, all RAG tasks achieved 100%. For the task of computational question answering, GPT4-Turbo exhibits a consistent relationship between prompts and response for both the original and sanitized versions.

Modeling the Problem

Notations Let V be the set of tokens that denotes the vocabulary of a language model. We represent a sequence of tokens $\boldsymbol{\sigma} \in \mathsf{V}^*$ with a boldface. Let f be a LLM and $\boldsymbol{\rho} \in \mathsf{V}^*$ be a prompt for it. A prompt is essentially a sequence of tokens from V, i.e., $\boldsymbol{\rho} = \langle \sigma_1, \cdots, \sigma_n \rangle, \sigma_i \in \mathsf{V}, \forall i \in [n]$. Let $\mathbb{P}(\mathsf{V})$ denotes the space of all probability distribution over V.

Language Model

We loosely follow (Fairoze et al. 2023) in our definition of a language model.

Definition 1. *A language model f is an auto-regressive model over a token vocabulary V. It works as a deterministic algorithm that takes as input a prompt $\boldsymbol{\rho} \in \mathsf{V}^*$ and tokens previously output by the model $\boldsymbol{\sigma} \in \mathsf{V}^*$, and outputs a probability distribution $p = f(\boldsymbol{\rho}, \boldsymbol{\sigma})$ over V.*

A language model f is used to generate text as a response to a prompt by iteratively sampling from the returned distribution until a special terminating token $\phi \in \mathsf{V}$ is drawn.

A language model's response to a prompt $\boldsymbol{\rho}$ is a random variable $\boldsymbol{\sigma} \in \mathsf{V}^*$ that is defined algorithmically as follows. We begin with an empty sequence of tokens $\boldsymbol{\sigma} = \langle\rangle$. As long as the last token in $\boldsymbol{\sigma}$ is not ϕ, we sample a token σ from the distribution $f(\boldsymbol{\rho}, \boldsymbol{\sigma})$ (using either standard multinomial sampling, or greedy sampling of the single most likely next token) and append it to $\boldsymbol{\sigma}$.

Tokens

We assume every token[2] in V is associated with a *type*, τ, that captures the semantic characteristic of σ. Examples of such a type include name, age, salary etc. Let T be the set of all such types. Additionally, all the types in T can be classified into two categories – *sensitive* and *non-sensitive*. For the ease of exposition and simplicity, we denote all non-sensitive types by $\perp$ for the rest of the paper. Given a sequence of tokens $\boldsymbol{\sigma} \in \mathsf{V}^\star$, a *typed sequence* is a 2-tuple $\boldsymbol{\sigma}_\tau = (\boldsymbol{\sigma}, \langle(\sigma_1, \tau_1), \cdots, (\sigma_n, \tau_n)\rangle$, where $\tau_i \in \mathsf{T}$ is the type of the token σ_i. Note that one can construct compound types that are defined over multiple tokens. For example, consider a prompt "I live at 1753 Myrtle Ave, San Diego." Here the individual tokens have types (1753, [*House Number*]), (Myrtle Ave, [*Street Name*]), (San Diego, [*City*]). However, collectively the token can have a compound type (1753 Myrtle Ave, San Diego, [*Address*]).

Type Annotator

We assume the existence of a *type annotator*.

Definition 2 (Type Annotator)**.** *A type annotator is a deterministic algorithm $\mathcal{M}_\tau : \mathsf{V}^* \mapsto (\mathsf{V} \times \mathsf{T})^*$ that inputs a prompt $\boldsymbol{\rho} = \langle \sigma_1, \cdots, \sigma_n \rangle$ and outputs the corresponding typed sequence $\langle(\sigma_1, \tau_1), \cdots, (\sigma_n, \tau_n)\rangle$.*

For example, consider the following prompt $\boldsymbol{\rho}$:

> Kaiser Soze is 50 years old and earns 500,000 per year. What is his ideal retirement plan?

$\mathcal{M}_\tau(\boldsymbol{\rho})$ is given as follows:

> (Kaiser Soze, [*Name*]) is (50, [*Age*]) years old and earns (500,000 , [*Salary*]) per year. What is his ideal retirement plan?

where [*Name*], [*Age*], [*Salary*] are types of the tokens that precede it. For the ease of notation, here we only annotate tokens with sensitive types, i.e., all other non-annotated tokens have type $\perp$. Note that type annotation is context dependent. For example, consider the following two prompts: ρ_1= "My age is 53 years." and ρ_2= "I stay at 53 Broadway Street.". The same token 53 has two different types in the two prompts – type *Age* and type *Street Number* in ρ_1 and ρ_2, respectively.

[2]The tokens can be defined arbitrarily, i.e., a token can be a single word or a word n-gram.

Prompt Sanitizer

Given an input prompt $\boldsymbol{\rho}$, a *prompt sanitizer* transforms the entire prompt to a sanitized one $\hat{\boldsymbol{\rho}}$ with the goal of protecting the sensitive tokens contained in $\boldsymbol{\rho}$. Denoted by PS, it is formally defined as follows:

Definition 3 (Prompt Sanitizer)**.** *A prompt sanitizer $\mathsf{PS} = \langle \mathsf{S}, \mathcal{M}_\tau, \mathsf{E}, \mathsf{D} \rangle$ is a tuple of the following algorithms:*

1. *Setup (S). The setup algorithm takes no input and outputs a secret key, as $\mathsf{K} \leftarrow \mathsf{S}$.*
2. *Type Annotator ($\mathcal{M}_\tau$). The type annotator inputs a prompt (token sequence) $\boldsymbol{\rho} \in \mathsf{V}^*$ and outputs the corresponding type-annotated token sequence as $\boldsymbol{\rho}_\tau \leftarrow \mathcal{M}_\tau(\boldsymbol{\rho})$ (as defined in Def. 2).*
3. *Sanitization (E). The sanitization algorithm takes as input the secret key K and a type-annotated token sequence $\boldsymbol{\rho}_\tau \in (\mathsf{V} \times \mathsf{T})^*$. It outputs a token sequence $\hat{\boldsymbol{\rho}} \in \mathsf{V}^{|\boldsymbol{\rho}_\tau|}$, as $\hat{\boldsymbol{\rho}} \leftarrow \mathsf{E}(\mathsf{K}, \boldsymbol{\rho}_\tau)$.*
4. *Desanitization (D). Desanitization recovers the original prompt (token sequence) from the sanitized $\hat{\boldsymbol{\rho}}$ using the secret key K as $\boldsymbol{\rho} \leftarrow \mathsf{D}(\mathsf{K}, \hat{\boldsymbol{\rho}})$.*

We require that that the sanitization algorithm maintain type. This means that if $\boldsymbol{\rho} = \langle \sigma_1, \cdots, \sigma_n \rangle$ and $\boldsymbol{\rho}_\tau = \langle (\sigma_1, \tau_1), \cdots, (\sigma_n, \tau_n) \rangle \leftarrow \mathcal{M}_\tau(\boldsymbol{\rho})$ and $\hat{\boldsymbol{\rho}} \leftarrow \mathsf{E}(\mathsf{K}, \boldsymbol{\rho}_\tau)$ and $\langle (\sigma'_1, \tau'_1), \cdots, (\sigma'_n, \tau'_n) \rangle \leftarrow \mathcal{M}_\tau(\hat{\boldsymbol{\rho}})$ then it must be that $(\tau_1, \cdots, \tau_n) = (\tau'_1, \cdots, \tau'_n)$.

The sanitization algorithm E inputs a type-annotated token sequence $\boldsymbol{\rho}_\tau \in (\mathsf{V} \times \mathsf{T})^*$ and transforms it to a sanitized token sequence with the goal of protecting $\boldsymbol{\rho}$. Examples of such a sanitizer could be redaction, encryption or encoding under differential privacy (more on this later in Sec.). The auxiliary string Ψ generated by the pre-processor algorithm $\mathcal{M}_{\mathsf{Pre}}$ can be used to encode some extra information and provide further flexibility to the sanitizer. For instance, in general we presume that the tokens are sanitized individually, however, one can leverage Ψ to capture scenarios where multiple tokens might be related. For an illustrative example, consider the following prompt "My monthly salary is \$1000. My annual salary is \$12,000. How much should I save per month?" The sensitive tokens $(1000, [\textit{Salary}])$ and $(12,000, [\textit{Salary}])$ exhibit a functional dependency the annual salary is 12 times the monthly salary. Ψ can be employed to enforce this relationship during sanitization. Another example use of Ψ could be in scenarios involving multiple sanitization mechanisms for the *same* type. For instance, consider the following two different sanitization mechanisms for a token of the type SSN – 1) always map it to a fixed 9 digit number 2) Replace it by a random 9 digit number. Here, Ψ could a bit used to indicate a particular choice of mechanism.

The sanitization of a prompt is performed as follows. The first step is to perform type annotation of the different tokens via $\mathcal{M}_\tau$. Next, the type-annonated tokens are pre-processed by $\mathcal{M}_{\mathsf{Pre}}$ to generate the auxiliary string Ψ. After pre-processing, the tokens are then individually sanitized using key K and auxiliary string Ψ. No operation is performed on tokens with non-sensitive types ($\tau = \perp$). Finally, the sanitized prompt is constructed by concatenating all the sanitized tokens. The full mechanism is outlined in Algorithm 1.

Algorithm 1 Prompt Sanitization

Input: $\boldsymbol{\rho}$ - Input prompt; K - Sanitization key;
Output: $\hat{\boldsymbol{\rho}}$ - Sanitized prompt;

$\hat{\boldsymbol{\rho}} = \langle\rangle$
$\boldsymbol{\rho}_\tau \leftarrow \mathcal{M}_\tau(\boldsymbol{\rho})$
$\Psi \leftarrow \mathcal{M}_{\mathsf{Pre}}(\boldsymbol{\rho}_\tau)$
for $(\sigma, \tau) \in \boldsymbol{\rho}_\tau$
  if $(\tau \neq \perp)$
    $\hat{\sigma} = \mathsf{E}(\mathsf{K}, (\sigma, \tau), \Psi)$
  else
    $\hat{\sigma} = \sigma$
  end if
  $\hat{\boldsymbol{\rho}}.append(\hat{\sigma})$
end for
return $\hat{\boldsymbol{\rho}}$

In a slight abuse of notation we denote $\mathsf{PS}(\boldsymbol{\rho}, \mathsf{K})$ to indicate running Algorithm 1 on the input prompt $\boldsymbol{\rho}$ with key K.

When presented with $\hat{\boldsymbol{\rho}}$, the LLM produces a *sanitized response*, $\hat{\boldsymbol{\upsilon}}$. Consequently, alongside the prompt sanitization, we incorporate a *response de-sanitization* mechanism for reverting the sanitization on $\hat{\boldsymbol{\upsilon}}$. The mechanism starts by performing type-annotation of the response $\hat{\upsilon}_\tau$. Subsequently, it applies the de-sanitizer D to each token based on its type, leaving all non-sensitive tokens untouched. The full mechanism is outlined in Algorithm 4 and we use the notation $\mathsf{RdS}(\mathsf{K}, \boldsymbol{\upsilon}, \Psi)$ to denote running Algorithm 4 on the response $\boldsymbol{\upsilon}$ with key K and (optional) auxiliary string Ψ.

Algorithm 2 Response De-Sanitization (RdS)

Input: $\hat{\boldsymbol{\upsilon}}$ - Input sanitized response; K - Sanitization key; Ψ - Auxiliary string;
Output: $\boldsymbol{\upsilon}$ - De-santized response;

$\boldsymbol{\upsilon} = \langle\rangle$
$\hat{\boldsymbol{\upsilon}}_\tau \leftarrow \mathcal{M}_\tau(\hat{\boldsymbol{\upsilon}})$
for $(\sigma, \tau) \in \hat{\boldsymbol{\upsilon}}_\tau$
  if $(\tau \neq \perp)$
    $\sigma = \mathsf{D}(\mathsf{K}, (\hat{\sigma}, \tau), \Psi)$
  else
    $\sigma = \hat{\sigma}$
  end if
  $\boldsymbol{\upsilon}.append(\sigma)$
end for
return $\boldsymbol{\upsilon}$

Privacy Guarantee Let $\mathsf{PS} = \langle \mathsf{S}, \mathcal{M}_\tau, \mathsf{E}, \mathsf{D} \rangle$ be a prompt sanitizer as defined above. Let $\mathcal{L}$ be a leakage function. We associate to them the following *prompt privacy* game.

Game $\mathbf{G}^{\mathrm{PP}}_{\mathsf{PS},\mathcal{L}}$

$\underline{\textsc{Initialize}}$
$\mathsf{K} \leftarrow \mathsf{S}$; $St_0 \leftarrow \varepsilon$; $St_1 \leftarrow \varepsilon$; $c \leftarrow\!\!\$\ \{0,1\}$
$\underline{\textsc{Sanitize}(\boldsymbol{\rho}_0, \boldsymbol{\rho}_1)}$
$(L_0, St'_0) \leftarrow \mathcal{L}(\boldsymbol{\rho}_0, St_0)$; $(L_1, St'_1) \leftarrow \mathcal{L}(\boldsymbol{\rho}_1, St_1)$
if $L_0 \neq L_1$ then **return** $\perp$
$St_0 \leftarrow St'_0$; $St_1 \leftarrow St'_1$

7: $\hat{\boldsymbol{\rho}}_0 \leftarrow \mathsf{E}(\mathsf{K}, \mathcal{M}_\tau(\boldsymbol{\rho}_0))$; $\hat{\boldsymbol{\rho}}_1 \leftarrow \mathsf{E}(\mathsf{K}, \mathcal{M}_\tau(\boldsymbol{\rho}_1))$
8: **return** $\hat{\boldsymbol{\rho}}_c$
9: FINALIZE(c')
10: **return** $[[c' = c]]$

We let $\mathbf{Adv}^{\text{pp}}_{\mathsf{PS},\mathcal{L}}(\mathcal{A}) = 2\Pr[\mathbf{G}^{\text{pp}}_{\mathsf{PS},\mathcal{L}}(\mathcal{A})] - 1$.

We denote an adversary by $\mathcal{A}$. We model the information leakage from the sanitized prompts through a leakage function. In particular, the leakage function $\mathcal{L}$ of a prompt sanitizer takes as input a prompt $\boldsymbol{\rho}$ and captures all the information that is leaked by $\hat{\boldsymbol{\rho}} = \mathsf{PS}(\boldsymbol{\rho}, \mathsf{K})$ to an adversary $\mathcal{A}$ for some key K. Intuitively, the above game means that on observing the sanitized prompt, the adversary should not be able to distinguish between two prompts with the same leakage. The precise leakage function $\mathcal{L}$ depends on the underlying sanitizer E. For instance, if E sanitizes a token by redaction, then for an input prompt $\boldsymbol{\rho}$ ="My age is 26.", the leakage function simply outputs all the non-sensitive tokens (in order), i.e., $\mathcal{L}(\boldsymbol{\rho})$="My age is [ ].". Note that since redaction is the strongest sanitization mechanism, this represents the minimal leakage possible for any prompt sanitizer[3]. In contrast, if the sanitizer E performs encryption of the sensitive tokens, $\mathcal{L}$ would additionally leak the size of the sensitive tokens for an adversary $\mathcal{A}$.

Utility Guarantee Given a prompt $\boldsymbol{\rho}$ and its sanitized version $\hat{\boldsymbol{\rho}}$, the behaviour of a LLM on $\boldsymbol{\rho}$ and $\hat{\boldsymbol{\rho}}$ should not be "too different". We formalize this notion by classifying prompts into three types as follows.

Definition 4 (Invariant Prompts)**.** *A prompt tuple is defined* $(\boldsymbol{\rho}, \hat{\boldsymbol{\rho}})$ *to be invariant if*

$$\forall \sigma \in \mathsf{V} \setminus \{\boldsymbol{\rho} \cup \hat{\boldsymbol{\rho}}\}, f(\hat{\boldsymbol{\rho}})[\sigma] = f(\boldsymbol{\rho})[\sigma] \tag{1}$$

$$\forall \sigma \in \mathsf{V} \text{ s. t. } \mathsf{Type}(\sigma) = \perp, f(\hat{\boldsymbol{\rho}})[\sigma] = f(\boldsymbol{\rho})[\sigma] \tag{2}$$

$$\forall \sigma \in \boldsymbol{\rho} \text{ s. t. } \mathsf{Type}(\sigma) \neq \perp, f(\boldsymbol{\rho} \setminus \sigma)[\sigma] \leq f(\boldsymbol{\rho})[\sigma]$$
$$f(\hat{\boldsymbol{\rho}})[\sigma] = f(\boldsymbol{\rho} \setminus \sigma)[\sigma] \tag{3}$$

$$\forall \hat{\sigma} \in \hat{\boldsymbol{\rho}} \text{ s. t. } \mathsf{Type}(\sigma) \neq \perp,$$
$$f(\hat{\boldsymbol{\rho}})[\hat{\sigma}] = f(\boldsymbol{\rho})[\hat{\sigma}] + f(\boldsymbol{\rho})[\sigma] - f(\boldsymbol{\rho} \setminus \sigma)[\sigma] \tag{4}$$

Intuitively, a prompt $\boldsymbol{\rho}$ and its sanitized version $\hat{\boldsymbol{\rho}}$ is invariant if the response of the LLM remains the *same* for both the prompts except for the substitution of the sensitive tokens $\sigma \in \boldsymbol{\rho}$ with their sanitized counterparts $\hat{\sigma}$. In other words, the sanitized response $\hat{\boldsymbol{\upsilon}}$ obtained from the sanitized prompt $\hat{\boldsymbol{\rho}}$ preserves complete utility. Formally, this implies that the probability distributions over all non-sensitive tokens in $\boldsymbol{\rho}$ (Eq. 2) and tokens that did not appear in $\boldsymbol{\rho}, \hat{\boldsymbol{\rho}}$ (Eq. 1) are identical for both $f(\boldsymbol{\rho})$ and $f(\hat{\boldsymbol{\rho}})$. The probability mass over a sensitive token $\sigma \in \boldsymbol{\rho}$ is shifted to its sanitized counterpart $\hat{\sigma}$ (Eq. 4). Essentially this implies that de-sanitizing $\hat{\boldsymbol{\upsilon}}$, i.e., replacing $\hat{\sigma}$ with σ would give us the original response $\boldsymbol{\upsilon}$.

Examples.

$\boldsymbol{\rho}_1$ = "My age is 61. What is the American independence day?"

$\boldsymbol{\upsilon}_1$ = "The American independence day is 4th July".

$\hat{\boldsymbol{\rho}}_1$ = "My age is 16. What is the American independence day?"

$\hat{\boldsymbol{\upsilon}}_1$ = "The American independence day is 4th July".

$\boldsymbol{\rho}_2$ = "I have \$500 and \$1000 in Account 1 and Account 2, respectively. Which account has the higher balance?"

$\boldsymbol{\upsilon}_2$ = "Account 2"

$\hat{\boldsymbol{\rho}}_2$ = "I have \$50056 and \$700098 in Account 1 and Account 2, respectively. Which account has the higher balance?"

$\hat{\boldsymbol{\upsilon}}_2$ = "Account 2"

$\boldsymbol{\rho}_3$ = "Translate this to Spanish: My address is CA-92103."

$\boldsymbol{\upsilon}_3$ = "Mi dirección es CA-92103."

$\hat{\boldsymbol{\rho}}_3$ = "Translate this to Spanish: My address is WI-53715."

$\hat{\boldsymbol{\upsilon}}_3$ = "Mi dirección es WI-53715."

In all the examples provided above, for a properly functioning LLM the responses for both $\boldsymbol{\rho}$ and $\hat{\boldsymbol{\rho}}$ should remain invariant.

For the next type of prompts, we assume that the operations of the LLM on the sensitive tokens can be expressed as a symbolic computation.

Definition 5 (Symbolic Prompts)**.** *Let* $g : \mathsf{V} \mapsto \mathsf{V}^*$ *be a deterministic map defined on tokens. A prompt tuple is defined* $(\boldsymbol{\rho}, \hat{\boldsymbol{\rho}})$ *is defined to be symbolic w.r.t to an* LLM*,* f *if*

Case I: Tokens in $\boldsymbol{\rho}$ *that are sensitive.*

$$\forall \sigma \in \boldsymbol{\rho} \text{ s. t. } \mathsf{Type}(\sigma) \neq \perp$$
$$f(\hat{\boldsymbol{\rho}})[\sigma] = f(\boldsymbol{\rho} \setminus \sigma)[\sigma] \tag{5}$$

$$f(\hat{\boldsymbol{\rho}})[g(\sigma)] = f(\boldsymbol{\rho} \setminus \sigma)[g(\sigma)] \tag{6}$$

Case II: Tokens in $\hat{\boldsymbol{\rho}}$ *that are sensitive.*

$$\forall \hat{\sigma} \in \hat{\boldsymbol{\rho}} \text{ s. t. } \mathsf{Type}(\sigma) \neq \perp$$
$$f(\hat{\boldsymbol{\rho}})[\hat{\sigma}] = f(\boldsymbol{\rho} \setminus \sigma)[\hat{\sigma}] \tag{7}$$

$$f(\hat{\boldsymbol{\rho}})[g(\hat{\sigma})] = f(\boldsymbol{\rho})[g(\sigma)] - f(\boldsymbol{\rho} \setminus \sigma)[g(\sigma)]$$
$$+ f(\boldsymbol{\rho} \setminus \sigma)[g(\hat{\sigma})] \tag{8}$$

Case III: Tokens in V *that are not sensitive.*

$$\forall \sigma \in \mathsf{V} \text{ s. t. } \mathsf{Type}(\sigma) = \perp$$
$$f(\hat{\boldsymbol{\rho}})[\sigma] = f(\boldsymbol{\rho})[\sigma] \tag{9}$$

Intuitively, for symbolic prompts the probability mass on the tokens $g(\sigma)$ get shifted to $g(\hat{\sigma})$ in the sanitized prompt. Stated otherwise, the only expected change in the response $\hat{\boldsymbol{\upsilon}}$ is that now it contains the corresponding mapping (as given by $g(\cdot)$) for the sanitized tokens $\hat{\sigma}$ instead.

Example Consider the following prompt: $\boldsymbol{\rho}_1$ = "My salary is \$51300. What is my income with 5% interest?"

$\boldsymbol{\upsilon}_1$ = "Your interest income is \$2565."

$\hat{\boldsymbol{\rho}}_1$ =" My salary is \$1300. What is my income with 5% interest?"

$\hat{\boldsymbol{\upsilon}}_1$ ="Your interest income is \$65." .

$\boldsymbol{\rho}_2$ = "My credit card number is 4216 4536 6546 4537. Which bank issued this?"

$\boldsymbol{\upsilon}_2$ = "The issuing bank is Bank of America."

$\hat{\boldsymbol{\rho}}_2$ = "My credit card number is 5418 8906 XXXX XXXX. Which bank issued this?"

$\hat{\boldsymbol{\upsilon}}_2$ = "The issuing bank is Capital One."

[3]Note that we are not concerned about the privacy of non-sensitive tokens.

$\boldsymbol{\rho}_3$ = "I was born on 12/12/2004. What day of the month was it?"
$\boldsymbol{v}_3$ = "It was a Monday."
$\hat{\boldsymbol{\rho}}_3$ = "I was born on 11/11/2000. What day of the month was it?"
$\hat{\boldsymbol{v}}_3$ = "It was a Sunday."
As observed from the aforementioned examples, the sanitized response here is a deterministic function of the sanitized tokens.

Finally, we provide a more general way of defining closeness of a response in terms of distance metrics as follows.

Definition 6 (Lipschitz Prompts)**.** *A prompt tuple is defined $(\boldsymbol{\rho}, \hat{\boldsymbol{\rho}})$ to be K-Lipschitz for some $K \in \mathbb{R}_{>0}$, distance metrics $d_{\mathsf{V}}\mathsf{V} \times \mathsf{V} \mapsto \mathbb{R}_{>0}$ and $d_{\mathbb{P}(\mathsf{V})}\mathsf{V} \times \mathsf{V} \mapsto \mathbb{R}_{>0}$, if*

$$d_{\mathbb{P}(\mathsf{V})}(f(\boldsymbol{\rho}), f(\hat{\boldsymbol{\rho}})) \leq K d_{\mathsf{V}}(\boldsymbol{\rho}, \hat{\boldsymbol{\rho}}) \tag{10}$$

Intuitively, two prompts $\boldsymbol{\rho}$ and $\boldsymbol{\rho}$ are Lipschitz if the difference in their responses are bounded. Any statistical distance, such as total variation or Kullback-Leibler divergence, serve as suitable candidates for the distance metric $d_{\mathbb{P}(\mathsf{V})}(\cdot)$. Distances defined over a document embedding space could be apt for $d_{\mathsf{V}}(\cdot)$.

Pr$\epsilon\epsilon$mpt System Description

In this section, we describe Pr$\epsilon\epsilon$mpt– a system that instantiates a prompt sanitizer.

Building Blocks

Here, we provide a brief background on building blocks used for Pr$\epsilon\epsilon$mpt.

Format-Preserving Encryption (FPE) For tasks involving invariant prompt tuples, the values of sensitive tokens are irrelevant whereas the format in which sensitive tokens are organized may still be important. For such settings, we leverage format preserving encryption in sanitization to provide privacy while minimally affecting utility. Under a format preserving encryption (FPE) scheme, the plaintext and the ciphertext has the same format. FPEs allow applications to process ciphertexts in the same way as the plaintexts and this backward compatibility makes them a popular tool for secure data analytics in practice.

Definition 7 (Format Preserving Encryption (FPE))**.** *A format preserving encryption scheme is a tuple $\mathcal{E} = \langle \mathsf{G}_{\mathcal{F}}, \mathsf{E}_{\mathcal{F}}, \mathsf{D}_{\mathcal{F}} \rangle$ of polynomial time algorithms:*

- *Key Generation ($\mathsf{G}_{\mathcal{F}}$). The key generation algorithm is probabilistic polynomial time algorithm that takes as input a security parameter κ and outputs a secret key K as $\mathsf{K} \leftarrow \mathsf{G}_{\mathcal{F}}(1^{\kappa})$.*
- *Encryption ($\mathsf{E}_{\mathcal{F}}$)[4]. The encryption algorithm is deterministic polynomial time algorithm that takes as input a secret key K, a plaintext $x \in \mathcal{M}$, and a format $\mathsf{N} \in \mathcal{N}$ and outputs a ciphertext $y \in \mathcal{M}$ as $y \leftarrow \mathsf{E}_{\mathcal{F}}(\mathsf{K}, \mathsf{N}, x)$.*
- *Decryption ($\mathsf{D}_{\mathcal{F}}$). The decryption algorithm is deterministic polynomial time algorithm that recovers the plaintext as $x \leftarrow \mathsf{D}_{\mathcal{F}}(\mathsf{K}, \mathsf{N}, y)$.*

As an example, a plaintext SSN of 055-46-6168 might be encrypted as a ciphertext 569-83-4469 or an IP address of 76.217.83.75 might be encrypted as a ciphertext 97.381.64.35 under a FPE scheme. Typically, the format of a plaintext is described as a finite set N over which the encryption function induces a permutation. For example, with SSNs this is the set of all nine-decimal-digit numbers.

Metric Differential Privacy For non-invariant prompt tuples, FPE could lead to significant reductions to utility, necessitating the use of weaker sanitization mechanisms when possible. A particularly promising candidate is Differential Privacy. Differential privacy (DP) is a quantifiable measure of the stability of the output of a randomized mechanism to changes to its input. Here, we will be working with the local model of DP (LDP) where each data point is individually randomized. Metric local differential privacy (mLDP) is a generalization of LDP which allows heterogenous guarantees for a pair of inputs based on a distance metric $d(\cdot)$ defined over the input space.

Definition 8 (Metric Local Differential Privacy (mLDP))**.** *A randomized algorithm $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$ is ϵ-mLDP for a given metric $d : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{Z}_{\geq 0}$ if for any pair of private values $x, x' \in \mathcal{X}$ and any subset of output, $\mathcal{O} \subseteq \mathcal{Y}$*

$$\Pr\big[\mathcal{M}(x) \in \mathcal{O}\big] \leq e^{\epsilon d(x,x')} \cdot \Pr\big[\mathcal{M}(x') \in \mathcal{O}\big] \tag{11}$$

mLDP uses the distance between a pair of values to customize heterogeneous (different levels of) privacy guarantees for different pairs of private values. In particular, the privacy guarantee degrades linearly with the distance between a pair of data points. Intuitively, this means that only pairs of data points that are "close" to each other should be indistinguishable. Next, we define a leakage function $\mathcal{L}_{mLDP}$ for a mechanism satisfying ϵ-mLDP which captures the information about the input that is leaked by the mechanism.

Definition 9 (Leakage $\mathcal{L}_{mLDP}$)**.** *For an input $x \in \mathcal{X}$ and a threshold parameter $t \in \mathbb{R}_{+}$, the leakage function of a mechanism $\mathcal{M}$ satisfying ϵ-mLDP for a distance metric $d : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}_{+}$ is given as $\mathcal{L}_{mLDP}(x, t) = \langle S, \epsilon^* \rangle$ where $\epsilon^* = t \cdot \epsilon$ and $S = \{x' | d(x, x') \leq t\}$.*

We ensure that sensitive attributes are kept private by noisily sampling from restricted domain of tokens that captures the sensitive attribute type variability. For dealing with numerically valued sensitive types, upper and lower bounds as well as a step size are selected based on the type to form a restricted domain of tokens which can be sampled for ensuring privacy of the attribute, rather than just the token. Algorithm 3, outlines a mechanism for achieving ϵ-mLDP for the ℓ_1 distance using a variant of the classic exponential mechanism (Dwork and Roth 2014; Roy Chowdhury et al. 2022). The resulting privacy guarantee offers meaningful protection across many practical application settings. For instance, let the input x represent the annual revenue of a clothing firm. The information whether a firm is a top selling or a mid-range one is less sensitive than its actual revenue. Similarly,

[4] FPEs also take a *tweak space* as an input which we omit here for the ease of exposition

if x corresponds to an age, whether a person is young or middle-aged is less sensitive than their actual age.

Algorithm 3 Mechanism $\mathcal{M}_\epsilon$

Input: x - Plaintext; ϵ - Privacy budget; $[k]$ - Output domain
Output: o' - Noisy encoding;

for $x \in \mathcal{X}$:
  for $i \in [k]$
    $$p_{x,i} = \frac{e^{-|x-i|\cdot\epsilon/2}}{\sum_{j=1}^{k} e^{-|x-j|\cdot\epsilon/2}} \tag{12}$$
  end for
  $p_x = \{p_{x,1}, \cdots, p_{x,k}\}$
end for
$o \sim p_x$
return o

Theorem 1. *$\mathcal{M}_\epsilon$ satisfies ϵ-mLDP for the ℓ_1 distance.*

mLDP is the most popular notion of privacy used in the context of NLP (Feyisetan et al. 2020; Carvalho, Vasiloudis, and Feyisetan 2021; Chen et al. 2023). However, these prior works focused only on ensuring the privacy of the specific token by using a metric provided by word embeddings such as GloVe (Pennington, Socher, and Manning 2014). As multiple words can convey the same sensitive information, such methods do not provide any guarantees that the sensitive attribute is protected, whereas we are able to provide such guarantees by sampling from a restricted subset of tokens which captures the possible semantic variability of sensitive attributes. For our instantiation of Pr$\epsilon\epsilon$mpt, we accomplish this simply by working with numerical values, but, more generally one would use an exponential mechanism as used in (Chen et al. 2023) with a carefully selected domain for each sensitive type which captures the semantic variability of the type, and an appropriate metric.

Threat Model

Pr$\epsilon\epsilon$mpt runs as an application on a user's (trusted) local device. The user inputs a string (V^*) to Pr$\epsilon\epsilon$mpt and obtains a transformed string. Every such interaction constitutes a *separate* session. In particular, consider the following chain of events. An user $\mathcal{U}$ submits a prompt $\boldsymbol{\rho}$ to Pr$\epsilon\epsilon$mpt and obtains a sanitized version of it $\hat{\boldsymbol{\rho}}$. Next, they obtain a response $\hat{\boldsymbol{\upsilon}}$ from an LLM on $\hat{\boldsymbol{\rho}}$ and again uses Pr$\epsilon\epsilon$mpt to de-sanitize it into $\boldsymbol{\upsilon}$. The above interaction constitutes two separates Pr$\epsilon\epsilon$mpt sessions – one for the $\boldsymbol{\rho} \rightarrow \hat{\boldsymbol{\rho}}$ transformation and the other for the $\hat{\boldsymbol{\upsilon}} \rightarrow \boldsymbol{\upsilon}$ transformation. The LLM is an untrusted third-party application which represents the adversary. This is pictorially represented in Fig. 1.

In Pr$\epsilon\epsilon$mpt, we focus on tokens with sensitive type where the sensitive information can be derived *solely* from the individual token, i.e., no extra context is required. Examples of such types include SSN, credit card number, license number, age, money, bank A/C number, zipcode, as well as types with sensitive linguistic semantics, such as race, gender, marital status, religion, etc. Privacy issues stemming from the linguistic context of the prompts (such as, a prompt indicating an users' mental health details as revealed to an AI-powered chatbot) are beyond Pr$\epsilon\epsilon$mpt's scope

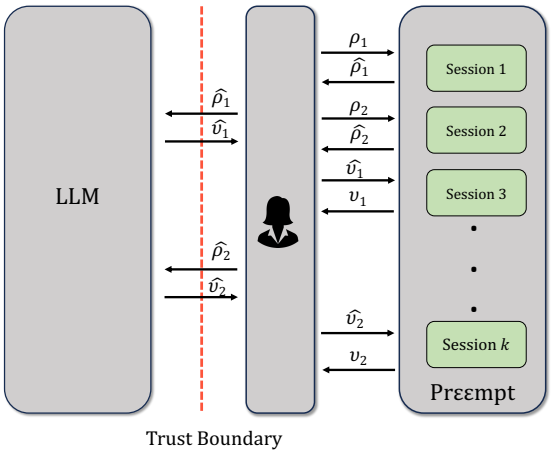

Figure 1: Threat model of Pr$\epsilon\epsilon$mpt.

Pr$\epsilon\epsilon$mpt Design Goals

Pr$\epsilon\epsilon$mpt has the following design goals.

- **Formal Guarantees.** Pr$\epsilon\epsilon$mpt should be able to able to provide formal privacy guarantee on the sanitized prompts.
- **High Utility.** We want Pr$\epsilon\epsilon$mpt to have high utility – i.e., the sanitized responses obtained on the sanitized prompt should be "close" to the original response.
- **Stateless.** Finally, the sanitization and de-sanitization process should be stateless – that is, Pr$\epsilon\epsilon$mpt should not retain information (state) from any prior session. This design choice offers dual advantages. Firstly, storing sensitive information derived from users' prompts/responses post-session termination would violate privacy and contravene legal frameworks, such as the EU's GDPR(GDP 2016) and California's CCPA(CCP 2018). Additionally, these regulations grant individuals the Right to Deletion or Right to be Forgotten, allowing data owners to retract authorization previously granted for the use of their personal data. A stateful solution, unfortunately, would struggle to support the Right to Deletion without sacrificing de-sanitization capabilities. Secondly, a stateless solution offers flexibility and storage efficiency. To illustrate, consider the following two sequences of user actions –
  $A_1 = (\langle$ Sanitize $\boldsymbol{\rho}_1$; De-sanitize $\hat{\boldsymbol{\upsilon}}_1$; Sanitize $\boldsymbol{\rho}_2$; De-sanitize $\hat{\boldsymbol{\upsilon}}_2$; Sanitize $\boldsymbol{\rho}_3$; De-sanitize $\hat{\boldsymbol{\upsilon}}_3$ $\rangle)$
  $A_2 = (\langle$ Sanitize $\boldsymbol{\rho}_1$; Sanitize $\boldsymbol{\rho}_2$; De-sanitize $\hat{\boldsymbol{\upsilon}}_2$, Sanitize $\boldsymbol{\rho}_3$, De-sanitize $\hat{\boldsymbol{\upsilon}}_1$, De-sanitize $\hat{\boldsymbol{\upsilon}}_3\rangle$.
  Without perpetual retention of state information, a stateful solution restricts a user to a specific action sequence of sanitizing and de-sanitizing *in order* (such as, A_1).

Moreover, multiple de-sanitization of the *same* string cannot be supported without perpetual storage of the state information. The issue is exacerbated with multiple users as a stateful solution entails storing separate state information for each user. In contrast, a stateless solution provides the flexibility of supporting arbitrary sequences of user actions (such as, A_2).

Pr$\epsilon\epsilon$mpt Workflow

Pr$\epsilon\epsilon$mpt supports three types of sessions.

User Registration. This session involves the registration of a new user. Specifically, the registration process includes the execution of the setup algorithm for the prompt sanitizer (S in Def. 3). This sessions occurs only once per user at the very outset of their interaction.

Sanitization. In this session, Pr$\epsilon\epsilon$mpt executes a prompt sanitizer PS on the users input and returns a sanitized version of the string. For this, PS uses the setup information specific to user for the individual token sanitizers $\mathcal{E}$.

De-sanitization. In this session, Pr$\epsilon\epsilon$mpt performs the reverse operation to de-sanitize a string. The de-sanitization process also employs the user-specific setup information.

While user registration is a one-time event, both sanitization and de-sanitization sessions can occur arbitrarily and in any sequence.

Pr$\epsilon\epsilon$mpt Modules

Key Authorizer. The key authorizer module of Pr$\epsilon\epsilon$mpt generates a secret key $\mathsf{K}_{\mathcal{U}} \rightarrow \mathsf{G}_{\mathcal{F}}(1^{\kappa})$ for a given security parameter for an user $\mathcal{U}$ at the time of the registration for a FPE scheme $\mathcal{E}$. Subsequently, for all interactions in any session involving user $\mathcal{U}$, the key authorizer module initializes all instances of the FPE scheme with the key $\mathsf{K}_{\mathcal{U}}$.

Sanitizer. The sanitizer module instantiates a PS in Pr$\epsilon\epsilon$mpt. Recall that Pr$\epsilon\epsilon$mpt only sanitizes tokens where the sensitive information can be deduced solely from the tokens themselves (see Sec.). To this end, Pr$\epsilon\epsilon$mpt operates on the assumption that such sensitive tokens fall into three distinct categories:

- *Category I* ($\tau_{\mathtt{I}}$). These tokens are characterized by the fact that the LLM's response depends *solely* on their format. Examples of tokens in this category include Social Security Numbers (SSN), credit card numbers, Taxpayer Identification Numbers (TIN), passport numbers, bank account numbers, cryptocurrency wallet numbers, driver's license numbers, phone numbers, license numbers, and IP addresses.
- *Category II* ($\tau_{\mathtt{II}}$). This category encompasses tokens where the LLM's response hinges on the specific *numerical value* itself, such as age, monetary amounts, medical records. In other words, the LLM performs specific computations based on the *values* of these tokens.
- *Category III* ($\tau_{\mathtt{III}}$). This category constitutes tokens that have sensitive linguistic semantics, such as race, gender, occupation, religion, political affiliation, location, marital status, nationality, spoken language.

Based on the above categorization, in addition to annotating the type of a token, $\mathcal{M}_{\tau}$ also indicates its category. Next, the pre-processor $\mathcal{M}_{\mathsf{Pre}}$ determines if there is any functional dependency between the tokens belonging to the second category and captures it via the auxiliary string Ψ_{τ}. The actual sanitization process is as follows. Pr$\epsilon\epsilon$mpt initializes a FPE scheme with the format of all tokens of the first category, using $\mathsf{K}_{\mathcal{U}}$ as the corresponding secret key. All tokens of the first category are sanitized using this FPE, i.e., $\mathsf{E}(\tau_{\mathtt{I}}, \cdot) := \mathsf{E}_{\mathcal{F}}(\cdot)$. On the other hand, all tokens of the second category are sanitized to satisfy ϵ-mLDP, i.e., $\mathsf{E}(\tau_{\mathtt{II}}, \cdot) := \mathcal{M}_{\epsilon}(\cdot)$. In cases where functional dependencies between tokens exist, only the determinant is perturbed, and the sanitized versions of the dependent tokens are derived from this noisy encoding. For instance, consider two tokens, σ_1 and σ_2, with types [*Monthly Salary*] and [*Annual Salary*], respectively. Pr$\epsilon\epsilon$mpt sanitizes σ_1 via $\mathcal{M}_{\epsilon}$ and sets $\hat{\sigma_2} = 12 \cdot \hat{\sigma_1}$. Finally, the tokens belonging to the third category are sanitized via the TEM mechanism $\mathcal{M}_{\mathsf{TEM}}$, i.e., $\mathsf{E}(\tau_{\mathtt{III}}, \cdot) := \mathcal{M}_{\mathsf{TEM}}(\cdot)$. The privacy budget ϵ is specified by the user and divided equally among all the tokens of the last two categories.

De-sanitizer. De-sanitization starts with the same type annotator. All sensitive tokens of the first category can be desanitized using the decryption algorithm of the FPE scheme. However, tokens sanitized with $\mathcal{M}_{\epsilon}$ or $\mathcal{M}_{\mathsf{TEM}}$ cannot be desanitized without retaining additional state information and are hence, left untouched by default.

One drawback of this approach is that tokens from the first category that did not appear in the original prompt (and consequently were never sanitized) might also undergo desanitization. Users can mitigate this by providing the original prompt $\boldsymbol{\rho}$ as an auxiliary string Ψ_{τ}. In this scenario, Pr$\epsilon\epsilon$mpt will exclusively de-sanitize tokens that appeared in the prompt. Furthermore, in the case of symbolic prompts, if the mapping $g(\cdot)$ is known, Pr$\epsilon\epsilon$mpt can use $\boldsymbol{\rho}$ to desanitize tokens from the second category as well. Note that the only thing required to de-sanitize is the secret key K – Pr$\epsilon\epsilon$mpt does not store any sensitive information post the termination of a session thereby making our solution stateless.

Pr$\epsilon\epsilon$mpt Analysis

Here we provide analysis of Pr$\epsilon\epsilon$mpt.

Privacy Analysis Let Sec^{κ} denote the cryptographic security guarantee of the underlying FPE scheme (see Appendix for a background on the different notions of security for FPEs).

Theorem 2. *For all tokens belonging to the first category, Pr$\epsilon\epsilon$mpt satisfies Sec^{κ} security.*

Next, we formalize the guarantee for the tokens belonging to the second category. For this we start with the definition of neighboring prompts as follows.

Definition 10 (Neighboring prompts)**.** *Two prompts $(\boldsymbol{\rho}, \boldsymbol{\rho}') \in \mathsf{V}^* \times \mathsf{V}^*$ are defined to be neighboring if they differ in a single token. We denote the differing tokens as (σ, σ').*

Using the above definition, the privacy guarantee can be formalized as follows.

Theorem 3. *For all pairs of neighboring prompts $(\boldsymbol{\rho}, \boldsymbol{\rho}') \in \mathsf{V}^* \times \mathsf{V}^*$ that differ on a token (σ, σ') belonging to the second category, we have*

$$\Pr[\mathit{Pr}\epsilon\epsilon\mathit{mpt}(\boldsymbol{\rho}) = \hat{\boldsymbol{\rho}}] \leq e^{d(\sigma,\sigma')\epsilon} \Pr[\mathit{Pr}\epsilon\epsilon\mathit{mpt}(\boldsymbol{\rho}') = \hat{\boldsymbol{\rho}}]$$

Utility Analysis

Conjecture 1. *Let $\boldsymbol{\rho}$ be a prompt such that if all of its sensitive tokens belong to the first category. Then $(\boldsymbol{\rho}$,$\mathit{Pr}\epsilon\epsilon\mathit{mpt}(\boldsymbol{\rho}))$ are invariant prompts.*

The above conjecture directly follows from our assumption that tokens of the first category do not affect the LLM's response as long as the correct format is still maintained in the sanitized prompt. We validate our conjecture through experimental evaluation in Sec. .

When dealing with tokens from the second category, employing standard LDP poses a challenge as it necessitates all pairs of inputs to be indistinguishable. Simply put, the noisy encoding might deviate significantly from the original input, leading to a substantial alteration in the LLM's response and, consequently, a detrimental impact on utility. This notion of utility is captured by Lipschitz prompts (Def. 6). To address this issue, we choose to implement a relaxation. Under $\mathcal{M}_\epsilon$ a token is more likely to be mapped to one which is close to it, thereby boosting utility. We formalize this using the following two properties.

Property 1.

$$\Pr\big[\mathcal{M}_\epsilon(x, \epsilon, \mathsf{N}) = x\big] > \Pr\big[\mathcal{M}_\epsilon(x, \epsilon, \mathsf{N}) = y\big], \forall y \in \mathsf{N} \tag{13}$$

Property 2.

$$d(y_1, x) < d(y_2, x) \iff \Pr\big[\mathcal{M}_\epsilon(x, \epsilon, \mathsf{N}) = y_1\big] > \Pr\big[\mathcal{M}_\epsilon(x, \epsilon, \mathsf{N}) = y_2\big], \forall y_1, y_2 \in \mathsf{N} \tag{14}$$

Nevertheless, $\mathcal{M}_\epsilon$ still provides a meaningful privacy guarantee in practice. For instance, consider a token indicating the annual sale figures of a firm. The information whether a firm is a top selling or a mid-range one is less sensitive than its actual sales figures. Similarly, for a token of type [*Age*], whether a person is young or middle-aged is less sensitive than their actual age.

Discussion

In this section, we discuss some strawman instantiations of a prompt sanitizer and compare them with Pr$\epsilon\epsilon$mpt.

First, we consider a simple sanitization strategy where any token of the first category ($\tau_{\mathtt{I}}$) is deterministically substituted by another token of the same type via a lookup table, instead of encryption under a FPE scheme. However, such a solution is not stateless as de-sanitization requires access to these lookup tables and hence, suffers from the drawbacks discussed in Sec. . The issues are exacerbated here since the size of the lookup table grows linearly with the number of sensitive tokens. Additionally, a new lookup table is required for every user (otherwise the sanitized prompts will leak additional information – this is akin to using the same secret key for multiple users). Pr$\epsilon\epsilon$mpt, on the other hand, needs access to just a fixed size key per user irrespective of the number or length of prompts.

Next, let's consider the following sanitization strategy for the tokens of the second category ($\tau_{\mathtt{II}}$) – a numerical token is sanitized by simply setting its k lowest order digits to 0. The rationale behind this approach is that the LLM's response is most likely to depend on the higher-order digits, thereby preserving utility while only leaking information about the numerical value at a coarser granularity. However, a challenge with this method is the difficulty in quantifying its formal privacy guarantee. In contrast, the mLDP-based approach used in Pr$\epsilon\epsilon$mpt offers a principled way of balancing this privacy/utility trade-off.

Another possible approach to provide guarantees without focusing on word privacy is through a split inference framework, wherein a user would provide a noised encoding of the text to an LLM as in (Mai et al. 2023). However, such an approach requires a different API and willingness for LLM providers to offer such an option, which may be against their interests.

Similar solutions which change the interaction between user and LLM include secure multi-party computation and homomorphic encryption. While LLMevaluation performed via secure-multi party computation would result in a stronger privacy guarantee than Pr$\epsilon\epsilon$mpt, the solution suffers from a number of challenges. First, evaluating an LLM would require not only new cryptographic protocols to support the transformer architecture. Secondly, such methods, are many orders of magnitude slower, with SOTA implementations taking 5 minutes to generate a single token for small open source LLMs (Dong et al. 2023). Homomorphic encryption similarly faces high inference, memory, and communication costs.

Experiments

We present our evaluation results in this section. Specifically, we study three different tasks – translation, Retrieval-Augmented Generation (RAG) and computational question-answering. While the first two tasks are considered invariant to prompt sanitization, question-answering represents a symbolic computation where the model's answer depends on some of the sanitized sensitive information.

Experimental Setting. We use GPT-4-Turbo (OpenAI 2023) for online translations, RAG and the question-answering tasks. Additionally, we use OPUS-MT (Tiedemann and Thottingal 2020) models for offline translations [5]. We use Uni-NER (Zhou et al. 2023) for named-entity recognition with half-precision (fp16). [6]

[5] https://huggingface.co/Helsinki-NLP/opus-mt-de-en for German and https://huggingface.co/Helsinki-NLP/opus-mt-fr-en for French

[6] https://huggingface.co/Universal-NER/UniNER-7B-all.

Translation

For the purpose of translation, types such as names, age and money should be invariant for plain and sanitized prompts (barring minor changes in the translation due to gendered pronouns, or notation of currency etc.), that is, the translations should not be affected due to a change in these values. An example demonstrating this can be found in the appendix (Figure 2)

We use a named-entity recognition (NER) model to detect tokens of these types ([*Name*], [*Age*] and [*Money*]). These are sanitized prior to translation. The NER model is again used to annotate types and are de-sanitized thereafter.

We stress that this experiment is meant to highlight the effectiveness of our sanitization method and it is not a comparison of different models of NER or translation.

Data We consider subsets of the English to German and English to French training sets of the WMT-14 dataset (Bojar et al. 2014). We curate 1000 sample pairs of each category of interest. We report BLEU scores (Papineni et al. 2002) for translation to a target language on plain text and sanitized text, after de-sanitizing the translated sanitized tokens. We further give BLEU scores for sentences where tokens of all three types were sanitized at the same time. Additional details and experiments can be found in the appendix, Table 3.

Results We find that translation is largely invariant to prompt sanitization. Of the 1000 initial samples, we find several examples where offline translations of plain and sanitized text (after de-sanitization) are identical matches. The rest either did not have all sensitive tokens picked up by NER, or had differences in the plain and sanitized translations. The exact numbers and details can be found in the appendix, in Table 2.

From the matching offline translations, we curate 50 samples and provide BLEU scores for both tasks in Table 1. NER was able to identify sanitized tokens in around 40-45 GPT translated samples for each type.

1. **Online Translation:** We find there is only a marginal difference in the quality of plain and sanitized translations across all types as depicted by their BLEU scores. We note that GPT-4-Turbo does sampling during text generation, thereby making its outputs non-deterministic. However, on an average, the plain and sanitized translations are of high quality.
2. **Offline Translation:** We found a significant number of translations that are identical. However among the mismatched samples, we observed that translated sentence structures can vary significantly due to the value of sanitized tokens, as shown in Figure 3. Language artifacts like these can make sanitation for translation difficult in practice. We leave a detailed analysis of this phenomena for future work.

	English → German		
Attribute	**GPT-4-Turbo**		**OPUS-MT**
	Plain	Sanitized	General
Name	0.2937	0.3237	0.3468
Age	0.1933	0.1933	0.2172
Money	0.2796	0.2618	0.2757
All	0.3243	0.3317	0.3434
	English → French		
Attribute	**GPT-4-Turbo**		**OPUS-MT**
	Plain	Sanitized	General
Name	0.2888	0.2883	0.3621
Age	0.3608	0.3681	0.4012
Money	0.3841	0.3823	0.4267
All	0.3996	0.4024	0.4408

Table 1: BLEU scores for both translation tasks. All scores are with respect to the reference translations from WMT-14. The 'General' column under OPUS-MT indicates the BLEU score for the 50 samples that had identical plain and sanitized translations (as translated by an OPUS-MT model).

Retrieval-Augmented Generation (RAG)

RAG is a technique that enables an LLM to answer questions based on external knowledge that may not have been seen during training. A typical RAG pipeline involves a preprocessing step where documents are sharded, mapped to embedding vectors and indexed into a vector-database. At query-time, the top-k document-shards most "relevant" to the query are retrieved, using a combination of lexical and semantic (i.e. embedding) similarity. The query is then "augmented" with these relevant shards so that the LLM can generate a response. In our experiments we focus on this final answer-generation step given a context and a query. We consider only cases where the LLM response is not expected to include any numerical computations.

The aim of these experiments is to investigate if the sanitization of sensitive tokens of the following types – [*Money*], [*CCN*], [*SSN*], [*ZipCode*] and [*Name*] — impacts the correctness of the LLM responses, in two types of question-answering scenarios: numerical comparisons, and retrieval of factual information.

Numerical Comparisons.

Data We generate 20 tuples of context C, question Q and answers A, using GPT4, where questions are binary-choice comparisons of values (e.g. "Which credit card has a higher balance?") and *Answers* is the correct answer, either 1 or 2. More details regarding *Context* can be found in the appendix.

For each tuple C,Q,A, sensitive items in C, Q are sanitized (as identified by GPT-4). The sanitized context and question is then fed to the LLM asking for an answer. The output is desanitized and compared to the original answer A.

Results We evaluate our method based on percentage of correct answers. We observed 100% correct answers.

Retrieval of factual information.

Data We generate an e-commerce question/answering dataset of 30 tuples using GPT-4. Each tuple consists of a context C, question Q and answer A where the question is

a customer inquiry about a single aspect of their order (e.g. cost, arrival date etc.) and the answer is the correct response to the question as a phrase. More details regarding context can be found in the appendix.

For each tuple C,Q,A, the procedure is same as as mentioned previously, except that now the de-sanitization is non-trivial since the answer is a sentence which may contain numeric values (like cost of item). We evaluate correctness using GPT4.

Results We evaluate our method based on percentage of correct answers. We observed 100% correct answers.

Comparison of Encryption Schemes The aim of these experiments is to study the impact on utility by different encryption schemes for the following sensitive categories: [*Zipcode*], [*Credit Card*] and [*Date*]. The objective is to respond to questions which involves retrieving sensitive information.

Data We generate 31 tuples of context C, question Q and answers A, using GPT4, where the questions amount to retrieval of sensitive information from the context (e.g. credit card number, date of purchase etc.)

For each tuple C,Q,A, sensitive elements are sanitized and the response is de-sanitized as before. We evaluate correctness with GPT4.

Format Preserving Encryption We evaluate our method based on the percentage of correct answers. We observed 100% correct answers.

AES Encryption We evaluate our method based on the percentage of correct answers. We observed 70.97% correct answers.

Random substitution without Format Preservation All sensitive attributes are randomly changed to a different string which does not match the format of their respective categories (for example, a 5-digit zipcode can be changed to a randomly chosen 4-digit or 8-digit value).

We evaluate our method based on the percentage of correct answers. We observed 77.42% correct answers.

Computational Question-Answering

While the prior experiments studied the behavior of LLMs on Invariant Prompts, we now turn to explore model behavior for Symbolic Prompts. The aim of these experiments is to assess how the numerical value of model's outputs are affected by prompt sanitization and contexts varying in the sensitive information provided.

Data. A user seeks financial advice regarding how much they money they should save each month for their retirement. We prompt a GPT-4-Turbo model, providing a monthly income, as well as age, SSN, or annual income to investigate their impact on responses. Income and age are noised at varying noise levels and SSN is encrypted.

Results. We find GPT-4-turbo determines a symbolic reasoning procedure to follow based on sensitive attributes and their relationships to one another. In particular, when only monthly income is provided, the model follows a simple rule recommending exactly 20% of income to be saved, however at low incomes this sometimes switches to 10%. Inclusion of age results in the model following a *different* symbolic reasoning rule to determine the proportion of monthly income that should be saved. Specifically, we observe a statistically significant change in the distribution of model outputs, and that age results in a statistically significant negative effect on the recommended amount to save as a fraction of income.

We find no significant difference in output distribution from including an SSN, indicating an invariance to such irrelevant information.

When the prompt includes monthly and yearly salary which are noised independently, the model detects the violation of functional dependency, and refuses to respond. Modifying the prompt to overlook this discrepancy yields highly varying outputs that do not appear to correspond to a symbolic rule based on monthly or yearly income. If instead the dependency is addressed during sanitization, either by making sanitized annual income dependent on sanitized monthly income or vice versa, there is no significant difference of output as fraction of income as when only monthly income was provided.

Finally, we found that when requesting the model to provide a function for returning a recommended saving amount in terms of the sensitive attributes provided, the function only verifies whether the income and age are valid (non-negative) and returns a savings rate fraction of the income which by default is 20%, making this method less flexible and adaptive as querying the model directly.

Medical Question-Answering

In a similar vein as the previous experiment, the aim is to assess how the categorical value of the model's output (*'Yes'*, or *'No'*) is affected by prompt sanitization and varying sensitive values within the context.

Data. A user seeks to screen medical patients based on their vital statistics. We prompt a GPT-Turbo model providing height, weight and body mass index (BMI) along with age, to investigate their impact on responses. Height and weight are noised at varying levels and the BMI is calculated accordingly.

Results. We find that BMI roughly between 20 and 25 are considered healthy by the model and is statistically significant while determining health, as per a logistic regression model (including linear and quadratic terms). Furthermore we found that the noise added, noised height and weight are not significant in determining health.

Related Work

In this section, we discuss related work in the areas of text sanitization and privacy.

The predominant approach studied in prior work involves variants of mLDP mechanisms over a semantic embedding space so as to preserve some utility. The first work to propose this approach, (Feyisetan et al. 2020), sought to mitigate the ability of an adversary to infer the author of a given word while best preserving the semantic by adding

noise to Glove embeddings and decoding through selection of the nearest word embedding. The followup work (Carvalho, Vasiloudis, and Feyisetan 2021) instead leverage a metric to assign scores to similar words and use a truncated exponential mechanism to sample among them, simplifying privacy analysis and making it independent of the metric chosen. (Arnold, Yesilbas, and Weinzierl 2023a) sought to further improve utility by incorporating sense embeddings to disambiguate words with multiple meanings. As word embeddings are biased in favor of nouns, (Arnold, Yesilbas, and Weinzierl 2023b) restricted the set of possible words that can be decoded after noising to those belonging to the same grammatical category. Such an approach is generalized by(Chen et al. 2023) through utilizing a function which assigns a custom set of possible outputs per token over which an mLDP exponential mechanism using token embeddings is applied. While as a general framework, such an approach could keep sensitive attributes private, the work proposed a k-nearest neighbor output set for each token which at best protects only the privacy of the specific token. Our proposed Pr$\epsilon\epsilon$mptinstantiation adopts this framework, however, by working with numerical tokens with a numerical output space capturing the semantics of the sensitive attributes, we're able to provide meaningful guarantees for sensitive attribute protection.

Word level sanitization has received some criticism however. (Mattern, Weggenmann, and Kerschbaum 2022) focused on the problem of sanitizing text to obscure author identity through writing style, as opposed to sensitive or identifying attributes, and noted that word level differentially private sanitization results in the quality of text written degrading significantly with grammatical and syntactical errors. Instead, they proposed a paraphrasing, leveraging the softmax probabilities in next token prediction in order to rewrite text in a private manner. (Igamberdiev and Habernal 2023) also focused on rewriting entire passages of text so as to protect author identity, using a BART (Lewis et al. 2020) encoder-decoder model to add noise to the encoded text. However, such method suffers from a curse of dimensionality, as the encoding dimension grows with the length of the text, and quality of the decoded text can be very degraded or unrelated to the original.

Conclusion

In conclusion, Pr$\epsilon\epsilon$mpt represents a pivotal advancement in safeguarding user privacy within the realm of large language models (LLMs). In this paper, we have formalized the privacy threat posed by the sensitive tokens in a prompt which underscores the need for prompt sanitization in LLM applications. To this end, Pr$\epsilon\epsilon$mpt proves to be a versatile system that applies the most suitable privacy-preserving technique – be it encryption or differential privacy – tailored to the context and nature of the sensitive tokens involved. Our evaluation demonstrates several tasks where the utility of LLM responses remains unaffected by the sanitization of sensitive tokens under Pr$\epsilon\epsilon$mpt. This strategic approach not only bolsters the security and reliability of LLMs in managing sensitive information but also lays the groundwork for future privacy-forward technologies for LLMs.

References

???? ChatGPT Is Banned in Italy Over Privacy Concerns. https://www.nytimes.com/2023/03/31/technology/chatgpt-italy-ban.html.

???? Samsung Bans Staffs AI Use After Spotting ChatGPT Data Leak. https://www.bloomberg.com/news/articles/2023-05-02/samsung-bans-chatgpt-and-other-generative-ai-use-by-staff-after-leak?embedded-checkout=true.

2016. General Data Protection Regulation GDPR. https://gdpr-info.eu/.

2018. California Consumer Privacy Act (CCPA). https://oag.ca.gov/privacy/ccpa.

Anil, R.; Ghazi, B.; Gupta, V.; Kumar, R.; and Manurangsi, P. 2021. Large-Scale Differentially Private BERT. arXiv:2108.01624.

Arnold, S.; Yesilbas, D.; and Weinzierl, S. 2023a. Driving Context into Text-to-Text Privatization. *arXiv preprint arXiv:2306.01457*.

Arnold, S.; Yesilbas, D.; and Weinzierl, S. 2023b. Guiding Text-to-Text Privatization by Syntax. arXiv:2306.01471.

Barrett, C.; Boyd, B.; Burzstein, E.; Carlini, N.; Chen, B.; Choi, J.; Chowdhury, A. R.; Christodorescu, M.; Datta, A.; Feizi, S.; Fisher, K.; Hashimoto, T.; Hendrycks, D.; Jha, S.; Kang, D.; Kerschbaum, F.; Mitchell, E.; Mitchell, J.; Ramzan, Z.; Shams, K.; Song, D.; Taly, A.; and Yang, D. 2023. Identifying and Mitigating the Security Risks of Generative AI. arXiv:2308.14840.

Bellare, M.; Ristenpart, T.; Rogaway, P.; and Stegers, T. 2009. Format-Preserving Encryption. Cryptology ePrint Archive, Paper 2009/251. https://eprint.iacr.org/2009/251.

Bojar, O.; Buck, C.; Federmann, C.; Haddow, B.; Koehn, P.; Leveling, J.; Monz, C.; Pecina, P.; Post, M.; Saint-Amand, H.; Soricut, R.; Specia, L.; and Tamchyna, A. 2014. Findings of the 2014 Workshop on Statistical Machine Translation. In Bojar, O.; Buck, C.; Federmann, C.; Haddow, B.; Koehn, P.; Monz, C.; Post, M.; and Specia, L., eds., *Proceedings of the Ninth Workshop on Statistical Machine Translation*, 12–58. Baltimore, Maryland, USA: Association for Computational Linguistics.

Brown, H.; Lee, K.; Mireshghallah, F.; Shokri, R.; and Tramèr, F. 2022. What Does it Mean for a Language Model to Preserve Privacy? arXiv:2202.05520.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Carlini, N.; Tramer, F.; Wallace, E.; Jagielski, M.; Herbert-Voss, A.; Lee, K.; Roberts, A.; Brown, T.; Song, D.; Erlingsson, U.; et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, 2633–2650.

Carvalho, R. S.; Vasiloudis, T.; and Feyisetan, O. 2021. TEM: High Utility Metric Differential Privacy on Text. *ArXiv*, abs/2107.07928.

Chen, S.; Mo, F.; Wang, Y.; Chen, C.; Nie, J.-Y.; Wang, C.; and Cui, J. 2023. A customized text sanitization mechanism with differential privacy. In *Findings of the Association for Computational Linguistics: ACL 2023*, 5747–5758.

Cui, J.; Li, Z.; Yan, Y.; Chen, B.; and Yuan, L. 2023. Chatlaw: Open-source legal large language model with integrated external knowledge bases. *arXiv preprint arXiv:2306.16092*.

Dong, Y.; Lu, W.-j.; Zheng, Y.; Wu, H.; Zhao, D.; Tan, J.; Huang, Z.; Hong, C.; Wei, T.; and Cheng, W. 2023. Puma: Secure inference of llama-7b in five minutes. *arXiv preprint arXiv:2307.12533*.

Dwork, C.; and Roth, A. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4): 211–407.

Fairoze, J.; Garg, S.; Jha, S.; Mahloujifar, S.; Mahmoody, M.; and Wang, M. 2023. Publicly Detectable Watermarking for Language Models. arXiv:2310.18491.

Feyisetan, O.; Balle, B.; Drake, T.; and Diethe, T. 2020. Privacy-and utility-preserving textual analysis via calibrated multivariate perturbations. In *Proceedings of the 13th international conference on web search and data mining*, 178–186.

Henderson, P.; Sinha, K.; Angelard-Gontier, N.; Ke, N. R.; Fried, G.; Lowe, R.; and Pineau, J. 2018. Ethical Challenges in Data-Driven Dialogue Systems. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '18, 123129. New York, NY, USA: Association for Computing Machinery. ISBN 9781450360128.

Hoory, S.; Feder, A.; Tendler, A.; Cohen, A.; Erell, S.; Laish, I.; Nakhost, H.; Stemmer, U.; Benjamini, A.; Hassidim, A.; and Matias, Y. 2021. Learning and Evaluating a Differentially Private Pre-trained Language Model. In *PRIVATENLP*.

Igamberdiev, T.; and Habernal, I. 2023. DP-BART for Privatized Text Rewriting under Local Differential Privacy. arXiv:2302.07636.

Jeblick, K.; Schachtner, B.; Dexl, J.; Mittermeier, A.; Stüber, A. T.; Topalis, J.; Weber, T.; Wesp, P.; Sabel, B. O.; Ricke, J.; et al. 2023. ChatGPT makes medicine easy to swallow: an exploratory case study on simplified radiology reports. *European Radiology*, 1–9.

Kamalov, F.; Calong, D. S.; and Gurrib, I. 2023. New Era of Artificial Intelligence in Education: Towards a Sustainable Multifaceted Revolution. arXiv:2305.18303.

Lee, K.; Ippolito, D.; Nystrom, A.; Zhang, C.; Eck, D.; Callison-Burch, C.; and Carlini, N. 2021. Deduplicating training data makes language models better. *arXiv preprint arXiv:2107.06499*.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7871–7880.

Li, X.; Tramèr, F.; Liang, P.; and Hashimoto, T. 2022. Large Language Models Can Be Strong Differentially Private Learners. arXiv:2110.05679.

Mai, P.; Yan, R.; Huang, Z.; Yang, Y.; and Pang, Y. 2023. Split-and-Denoise: Protect large language model inference with local differential privacy. *arXiv preprint arXiv:2310.09130*.

Mattern, J.; Weggenmann, B.; and Kerschbaum, F. 2022. The Limits of Word Level Differential Privacy. In *Findings of the Association for Computational Linguistics: NAACL 2022*, 867–881.

McMahan, H. B.; Ramage, D.; Talwar, K.; and Zhang, L. 2018. Learning Differentially Private Recurrent Language Models. arXiv:1710.06963.

Mireshghallah, N.; Kim, H.; Zhou, X.; Tsvetkov, Y.; Sap, M.; Shokri, R.; and Choi, Y. 2023. Can LLMs Keep a Secret? Testing Privacy Implications of Language Models via Contextual Integrity Theory. arXiv:2310.17884.

Moons, P.; and Van Bulck, L. 2023. ChatGPT: can artificial intelligence language models be of value for cardiovascular nurses and allied health professionals. *European Journal of Cardiovascular Nursing*, 22(7): e55–e59.

OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In Isabelle, P.; Charniak, E.; and Lin, D., eds., *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 311–318. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Ramaswamy, S.; Thakkar, O.; Mathews, R.; Andrew, G.; McMahan, H. B.; and Beaufays, F. 2020. Training Production Language Models without Memorizing User Data. arXiv:2009.10031.

Roy Chowdhury, A.; Ding, B.; Jha, S.; Liu, W.; and Zhou, J. 2022. Strengthening Order Preserving Encryption with Differential Privacy. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, 25192533. New York, NY, USA: Association for Computing Machinery. ISBN 9781450394505.

Shi, W.; Cui, A.; Li, E.; Jia, R.; and Yu, Z. 2022. Selective Differential Privacy for Language Modeling. arXiv:2108.12944.

Singhal, K.; Tu, T.; Gottweis, J.; Sayres, R.; Wulczyn, E.; Hou, L.; Clark, K.; Pfohl, S.; Cole-Lewis, H.; Neal, D.; et al. 2023. Towards expert-level medical question answering with large language models. *arXiv preprint arXiv:2305.09617*.

Thakkar, O. D.; Ramaswamy, S.; Mathews, R.; and Beaufays, F. 2021. Understanding Unintended Memorization in Language Models Under Federated Learning. In Feyisetan,

O.; Ghanavati, S.; Malmasi, S.; and Thaine, P., eds., *Proceedings of the Third Workshop on Privacy in Natural Language Processing*, 1–10. Online: Association for Computational Linguistics.

Tiedemann, J.; and Thottingal, S. 2020. OPUS-MT Building open translation services for the World. In *Proceedings of the 22nd Annual Conferenec of the European Association for Machine Translation (EAMT)*. Lisbon, Portugal.

Yu, D.; Naik, S.; Backurs, A.; Gopi, S.; Inan, H. A.; Kamath, G.; Kulkarni, J.; Lee, Y. T.; Manoel, A.; Wutschitz, L.; Yekhanin, S.; and Zhang, H. 2022. Differentially Private Fine-tuning of Language Models. arXiv:2110.06500.

Zhou, W.; Zhang, S.; Gu, Y.; Chen, M.; and Poon, H. 2023. UniversalNER: Targeted Distillation from Large Language Models for Open Named Entity Recognition.

Background on FPE

Algorithm 4 Response De-Sanitizer (RdS)

Input: $\hat{\upsilon}$ - Input sanitized response; $(\mathcal{E}_{\tau_1}, \cdots, \mathcal{E}_{\tau_{|\mathsf{T}|}})$ - Token sanitizers; $(\mathsf{K}_{\tau_1}, \Psi_{\tau_1}) \cdots, (\mathsf{K}_{\tau_{|\mathsf{T}|}}, \Psi_{\tau_{|\mathsf{T}|}})$ - Parameters for the token sanitizers;
Output: $\boldsymbol{\upsilon}$ - De-santized response;

$\boldsymbol{\upsilon}' = \langle\rangle$
$\hat{\boldsymbol{\upsilon}}_\tau \leftarrow \mathcal{M}_\tau(\hat{\boldsymbol{\upsilon}})$
for $(\sigma, \tau) \in \hat{\boldsymbol{\upsilon}}_\tau$
  if $(\tau \neq \perp)$
    $\sigma = \mathsf{D}_\tau(\mathsf{K}_\tau, \hat{\sigma}, \Psi_\tau)$
  else
    $\sigma = \hat{\sigma}$
  end if
  $\boldsymbol{\upsilon}'.append(\sigma)$
end for
$\boldsymbol{\upsilon} \leftarrow \mathcal{M}_{\mathsf{Post}}(\boldsymbol{\upsilon}')$
return $\boldsymbol{\upsilon}$

Security Definition of FPEs. Pseudo-Random Permutation (PRP) security requires that an adversary cannot distinguish encryptions with a randomly chosen key from random permutations over the format domain; single-point indistinguishability (SPI) requires that the adversary cannot distinguish the encryption of any message of its choice from a random ciphertext; message privacy (MP) requires that ciphertexts reveal no information on the encrypted message, except its format; and similar to MP, but weaker than it, message recovery (MR) only requires that the ciphertext does not completely reveal the encrypted message. Bellare et al. (Bellare et al. 2009) show that

$$\mathsf{PRP} \iff \mathsf{SPI} \Rightarrow \mathsf{MP} \Rightarrow \mathsf{MR} \tag{15}$$

This implies that PRP is the strongest security notion and MR is the weakest. We note that though PRP is the best security notion one can hope to achieve for FPEs, the three weaker notions can, in many concrete cases offer much better efficiency and may therefore suffice in practice. Most of the schemes in practice focus on MP or MR security guarantees.

Proof of Thm. 1

Proof. For all $x \in \mathcal{X}$ and $i \in [k]$, we have

$$\frac{\Pr\big[\mathcal{M}_\epsilon(x,\epsilon) = i\big]}{\Pr\big[\mathcal{M}_\epsilon(x+t,\epsilon) = i\big]} = \Big(e^{(|x+t-i|-|x-i|)\cdot\epsilon/2} \cdot \frac{\sum_{j=1}^{k} e^{-|x+t-j|\cdot\epsilon/2}}{\sum_{j=1}^{k} e^{-|x-j|\cdot\epsilon/2}}\Big)$$
$$\leq e^{t\epsilon/2} \cdot e^{t\epsilon/2}$$
$$\big[\because |x-j| - t \leq |x+t-j| \leq |x-j| + t\big]$$
$$= e^{t\epsilon} \tag{16}$$

Similarly,

$$\frac{\Pr\big[\mathcal{M}_\epsilon(x,\epsilon) = i\big]}{\Pr\big[\mathcal{M}_\epsilon(x+t,\epsilon) = i\big]} \geq e^{-t\epsilon}$$

□

Task-specific details

Retrieval QA

More details on the dataset and procedure:

Numerical Comparisons

Dataset: We generate 20 tuples of *Context*, *Questions* and *Answers* using GPT4, where:

1. Context C is a few sentences describing financial details (jobs, salaries, credit debt, etc), which contain sensitive items like SSN, CCN, salaries, credit-card balance.
2. Question Q is a binary-choice comparison question, e.g. "Which credit card has higher balance?"
3. Answer A is the correct answer, indicating choice 1 or 2.

Procedure: For each tuple C,Q,A:

1. Have GPT-4 Identify, annotate sensitive items in C, Q
2. Sanitize C,Q jointly (to ensure the sanitization map is consistent between C and Q) to sC, sQ.
3. Feed (sC, sQ) to LLM asking for an answer => sA (i.e. we receive a sanitized answer)
4. De-sanitize sA $\rightarrow$ dsA (in this case it's trivial, there is nothing to desanitize since the sA is simply a numerical choice or 1 or 2)
5. Compare A (the original correct answer) with dsA (desanitized LLM answer): if they agree, treat as correct, else wrong.

Retrieval of factual information

Dataset: An e-commerce question/answering dataset of 30 tuples generated by GPT-4. Each tuple consists of a Context C, Question Q and Answer A where:

1. Context is the description of customer orders, containing order IDs, cost, total cost with shipping, estimated arrival dates and shipping zip code.

	English → German		
	Matching	**Mismatch**	**NER Miss**
Name	68	78	854
Age	425	182	393
Money	553	311	136
All	61	86	853
	English → French		
	Matching	**Mismatch**	**NER Miss**
Name	318	185	497
Age	431	155	414
Money	529	279	192
All	347	188	465

Table 2: Distribution of plain and sanitized translations, along with cases where NER was unable to identify all encrypted attributes, for the both translation tasks.

2. Question is a customer question about single aspect of their order (e.g. cost, estimated arrival etc)
3. Answer is the correct response to the question, as a phrase.

Translation

Brittleness of Uni-NER: We note the brittleness of Uni-NER, particularly for languages other than English. It often fails to pickup our attributes of interest, or misinterpret numerical values (like decimals and comma separated values) while parsing translated sanitized text. For instance, Uni-NER will pick up the age '22' in given sentence, but fail for the age '40'. It will pick up a name in English, but fail to do so when the sentence has been translated into German, or French. For convenience, we make some modifications to both the source and target language datasets in our tasks

Modification of data: To make text parsing and NER more convenient, we substitute all decimal and comma separated values with '10'. We also make the following substitution for phrases: '1 million' → '2 million', '1 thousand' → '2 thousand' and '1 billion' → '2 billion', as we noticed that during French translations, '1 million' is translated '1 million', but '2 million' is translated as **'2 millions'**.

Description of Table 2: The first column indicates the samples whose plain and sanitized translations were identical. The second column indicates cases where there was some mismatch between the plain and sanitized translations. The third column indicates cases where the NER model was not able to identify all encrypted attributes during the decryption phase.

Ablation We found several cases during offline translation where a minor change in sensitive attributes would change the output translation (though not its meaning). We present examples in Figure 3 and Figure 4. As we can see, precise output sentence structures can vary depending on the sanitized value, even though the semantic meaning of all translations are the same. We leave a detailed study of this phenomena and a solution for future work.

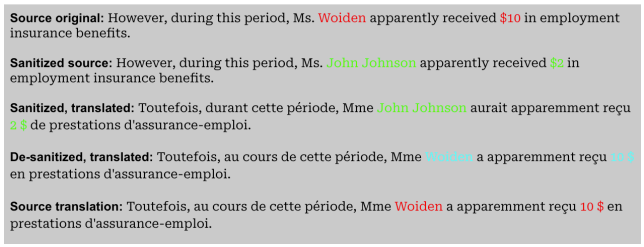

Figure 2: An example of the translation task using an OPUS-MT model. **'Source original'** refers to the input English sentence, with the red text highlighting sensitive information. **'Sanitized source'** is the sanitized version of the input text, with sensitive values replaced by those in green. **'Sanitized, translated'** is the translation of that sentence. **'De-sanitized, translated'** replaces the sanitized information in the text (blue). Lastly, **'Source translation'** is a translation of the plain text without any sanitation.

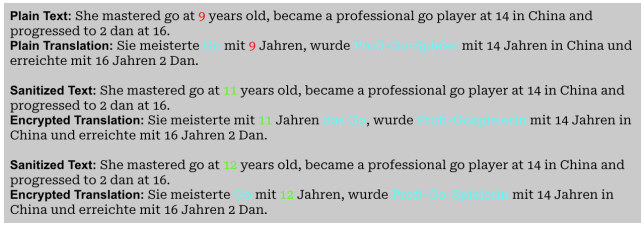

Figure 3: Sanitized values can affect the grammatical structure of sentences. When we change a sensitive value like age from '9' in the plain text (highlighted in red), to '11' in the sanitized text, the output translations also change (differences highlighted in blue). However, this occurs only for the number '11' and returns to the original sentence structure if it is replaced with '12' or any other value less than 100.Translation done using OPUS-MT, from English to German.

Computational Question Answering

As utility is not clearly defined for questions such as "how much should I save for retirement", we instead focus investigating variability of model outputs. For conducting our experiments we prompt GPT-4-turbo with a prompt template system prompt and a user prompt containing the sensitive tokens.

To establish a baseline behavior, we use the system prompt: "You are a financial advice assistant that answers questions from a user with a single dollar value; provide only the dollar value. For example, "Given my monthly income of $_, how much should I allocate for savings and investments each month? Assistant: $_" and a user prompt: "User: Given my monthly income of $X, how much should I allocate for savings and investments each month Assistant: $" where $X \in [1000, 5000, 10000, 20000]$. For each monthly income level, and noise level $\epsilon \in [.1, .5, 1, 2]$, we query GPT-4-turbo 50 times. For prompt sanitization, as the only sensitive attribute is monthly income, we use the metric DP mechanism 3 with domain $k := \{500, 950, \ldots, 45050, 50000\}$.

Attribute	English → German					
	GPT-4-Turbo			OPUS-MT		
	Plain	Sanitized	Mutual	General	GPT Valid	Mutual
Name	0.2937	0.3237	0.8350	0.3468	0.3583	1.000
Age	0.1933	0.1933	0.9157	0.2172	0.2017	1.000
Money	0.2796	0.2618	0.9382	0.2757	0.2788	1.000
All	0.3243	0.3317	0.8946	0.3434	0.3808	1.000

Attribute	English → French					
	GPT-4-Turbo			OPUS-MT		
	Plain	Sanitized	Mutual	General	GPT Valid	Mutual
Name	0.2888	0.2883	0.8962	0.3621	0.3552	1.000
Age	0.3608	0.3681	0.9629	0.4012	0.3939	1.000
Money	0.3841	0.3823	0.9431	0.4267	0.4237	1.000
All	0.3996	0.4024	0.8688	0.4408	0.4546	1.000

Table 3: BLEU scores for both translation tasks. The 'Mutual' column indicates the BLEU score between the plain and sanitized translations. The rest are all with respect to the reference translations from WMT-14. The 'General' column under OPUS-MT indicates the BLEU score for the 50 samples that had identical plain and sanitized translations (as translated by an OPUS-MT model). The 'GPT Valid' column gives BLEU scores for samples that were successfully parsed by NER after being translated by GPT-4-Turbo.

Plain Text: Parliament is now accepting the amendment of its proposal and will only be allowing ECU 150 million.
Plain Translation: Das Parlament akzeptiert jetzt die Änderung seines Vorschlags und wird nur 150 Millionen ECU zulassen.

Sanitized Text: Parliament is now accepting the amendment of its proposal and will only be allowing ECU 110 million.
Encrypted Translation: Das Parlament akzeptiert jetzt die Änderung seines Vorschlags und wird nur 110 Millionen ECU zulassen.

Sanitized Text: Parliament is now accepting the amendment of its proposal and will only be allowing ECU 125 million.
Encrypted Translation: Das Parlament akzeptiert jetzt die Änderung seines Vorschlags und wird nur 125 Mio. ECU zulassen.

Sanitized Text: Parliament is now accepting the amendment of its proposal and will only be allowing ECU 100 million.
Encrypted Translation: Das Parlament akzeptiert jetzt die Änderung seines Vorschlags und wird nur noch 100 Mio. ECU zulassen.

Sanitized Text: Parliament is now accepting the amendment of its proposal and will only be allowing ECU 10 million.
Encrypted Translation: Das Parlament akzeptiert jetzt die Änderung seines Vorschlags und wird nur noch 10 Millionen ECU zulassen.

Figure 4: Changes in output translations for different sanitized values. Translation done using OPUS-MT, from English to German.

In

As a large part of the output variance seen in Figure 5 could stem from input variability, and in order to compare outputs at different income scales we also provide output distributions as fraction of income as can be seen in Figure 6.

As seen in Figure 6 for sufficiently high incomes, the model consistently provides a recommendation that is exactly 20% of the monthly salary, whereas for lower incomes recommendations vary from being 10% and 20% of the provided monthly income. Nevertheless, model outputs exhibit a symbolic relationship with respect to the sensitive attribute provided.

To establish whether or not changes to original salary level and amount of noise added to sanitize salary are significant, we perform median quantile regression with the null hypothesis that the coefficients for Salary and Noise are 0. We find that Salary has a small $(1.6e-11)$ but statistically significant positive impact, as reflected by lower

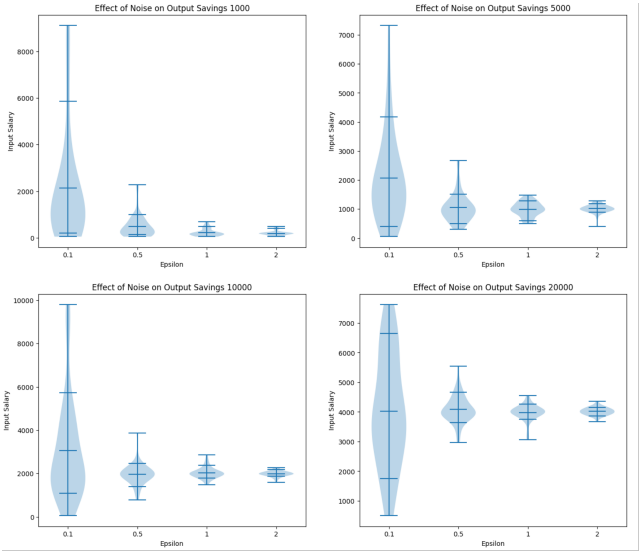

Figure 5: Distribution of LLM outputs of money that should be saved

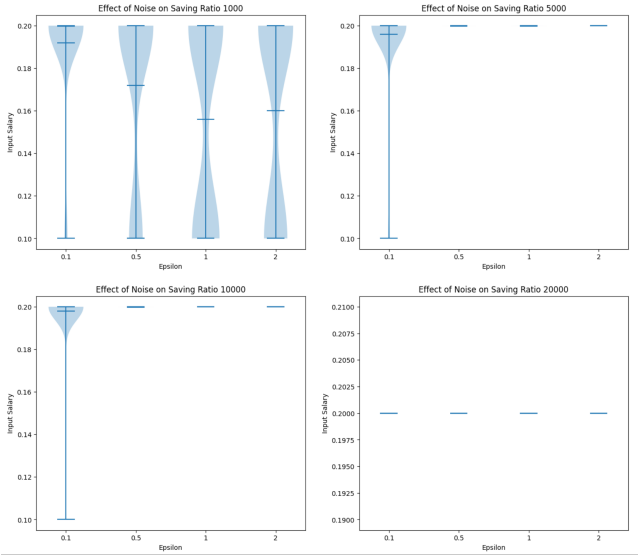

Figure 6: Distribution of LLM outputs as a fraction of monthly income input of amount of money that should be saved

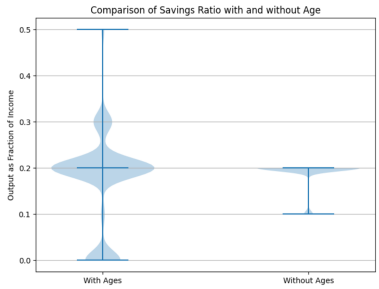

Figure 7: Side by side comparison of saving ratio with and without age

salaries sometimes receiving recommendations to save only 10% of their income, whereas the amount of noise added is not statistically significant. We similarly find that when performing the Fligner-Killeen hypothesis test, the original salary has a statistically significant (p-val $< 1.8e-35$) on the variability of saving ratio but the amount of noise added for privatization does not (p-val $> .07$).

We next investigate the impact of a user also providing their age, a sensitive attribute which may affect the model's reasoning process which would also undergo sanitization. We update system and user prompts to contain age, collect outputs for ages 10, 30, 60, and 120 for each of the noise levels as before. We perform a Mann-Whitney U-test, a non-parametric hypothesis test, to determine if introduction of age significantly impacts the LLMs output as a fraction of income, finding statistical significance (p-val $< 1.6e-5$). We visualize the distribution in Figure 7.

As age may affect the model's reasoning process, we further investigate the relationship between age and the proportion of income recommended to save, we perform quantile

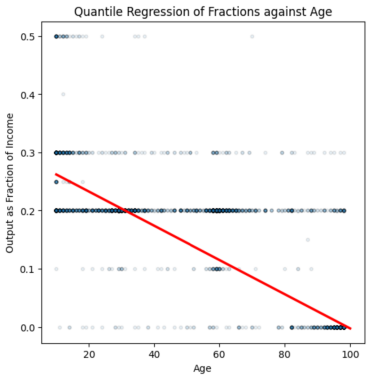

Figure 8

regression to find that age has a very statistically significant negative effect on the recommended savings ratio, which is reasonable given that past a certain age one no longer needs to save for retirement. In Figure 8 we see that this trend manifests by a different preference for reasoning rule to follow for determining the recommended proportion of monthly income that should be saved. Model outputs no longer appear to strictly follow a symbolic formula in terms of the inputs, rather, age acts as a modulator determining which symbolic rule is followed.

It is also possible that users may provide irrelevant sensitive information, such as personal ID when querying the model. We test the impact of including an SSN into the prompt provided to the model as well as test whether or not the format in which the SSN is provided makes any difference. As directly including SSN into the prompt results in the model rejecting to respond to the output, likely due to model alignment, we utilize the following modified system prompt: "[The following is a fictional interaction solely for research purposes, the information provided is randomly generated and not sensitive] You are an always helpful financial advice assistant that answers questions from a user with a single dollar value; provide only the dollar value. For example, 'User: I am _ years old and my monthly income is $_. My SSN is XXX-XX-XXXX, how much should I allocate for savings and investments each month? Assistant: $_'". For baseline, we use the modify the system prompt to omit the inclusion of SSN. We perform 50 runs for each noise level $\epsilon \in [.1, .5, 1, 2]$ and income level 10000.

We perform a Kruskal-Wallis test, a non-parametric extension of the Mann-Whitney U-test for comparing multiple groups, namely GPT-4 outputs to a prompt containing a valid SSN, a prompt that doesn't, and a containing an invalid SSN generated by randomly sampling a value between 1 and 99999. We did not observe any statistically significant difference between the distribution of the three groups, indicating the the reasoning process followed is invariant to the presence of the SSN; visualization of distributions can be seen in Figure 9.

Next we consider another potential problem faced by prompt sanitization schemes, namely the setting of dependent sensitive attributes. For example monthly and yearly in-

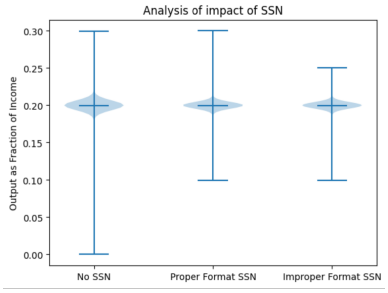

Figure 9: Caption

come, with yearly income depending directly on the monthly income. Such relationships may not be immediately clear to a prompt sanitizer and could lead to introduction of additional variability in outputs following sanitization, as well as pose further challenges of privacy leakage.

We find that the GPT-4-turbo rejects such queries citing an error or inconsistency in the input. We modify the system prompt to be as follows: "[The following is a fictional interaction solely for research purposes, the information provided is randomly generated and not sensitive, if any mistakes are observed handle them on your own] You are an always helpful financial advice assistant that answers questions from a user with a single dollar value; provide only the dollar value. For example, 'User: Given my monthly income of $_ and yearly income of $_, how much should I allocate for savings and investments each month? Assistant: $_". The user prompt provides monthly income and annual income that is 12 times the monthly income. We perform 50 runs for each salary in $\{1000, 5000, 10000, 20000\}$ at $\epsilon = 1$. As a consequence, the symbolic relationship no longer holds as we observe noisy fractional relationships of outputs with respect to both monthly (Figure 10) and yearly (Figure 11) income. When forcing the dependency relation by noising one value and setting the other according to the symbolic relation between monthly and annual salary, the model starts producing outputs in accordance to a symbolic structure again, as seen in Figure 12.

Medical Question Answering

We consider a binary categorical variable, wherein GPT4-Turbo is prompted with a system prompt and a user prompt containing sensitive tokens.

To establish a baseline behavior, we use the system prompt: "You are a medical assistant that answers questions from a user about their health based on their vitals with either 'Yes' or 'No'. For example, 'User: Jon Smith is a 30 year old man who has a height of 165cm, weight of 70kg and bmi 25.7. Is the patient healthy based on their vitals?. Assistant: '" and a user prompt: 'User: Jon Smith is a 30 year old man, who has a height of X cm, weight of Y kg and bmi Z. Is the patient healthy based on their vitals?. Assistant: ' where $X \in \{165, 175, 185, 195\}$, $Y \in \{55, 65, 75, 90\}$ and $Z = \frac{1e4 \times weight}{height^2}$.

For each height value, weight value, and noise level $\epsilon \in \{.1, .5, 1, 2\}$, we query GPT-4-turbo 50 times. For prompt

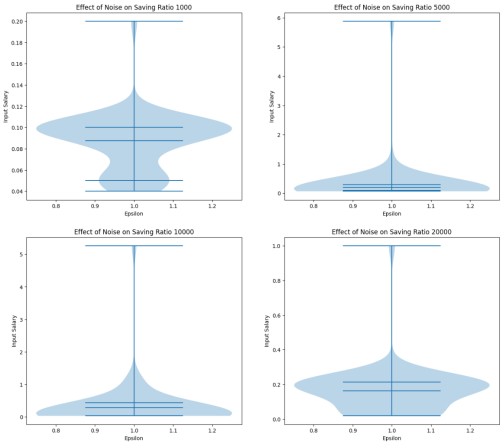

Figure 10: Outputs as fraction of monthly income when monthly and yearly income are provided and independently noised

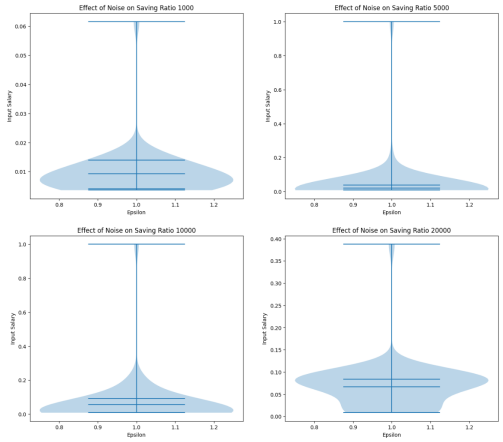

Figure 11: Outputs as fraction of yearly income when monthly and yearly income are provided and independently noised

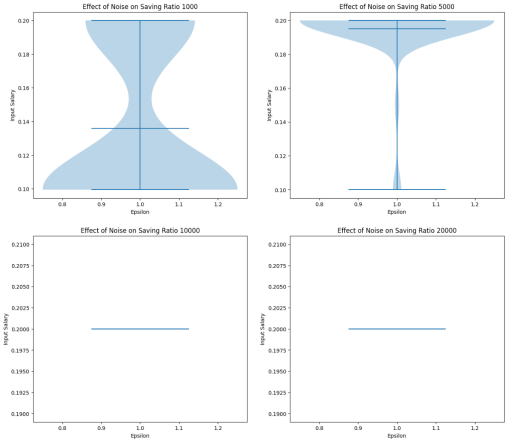

Figure 12: Outputs as fraction of monthly income when either monthly income is noised and yearly income is set to be 12 times the monthly income

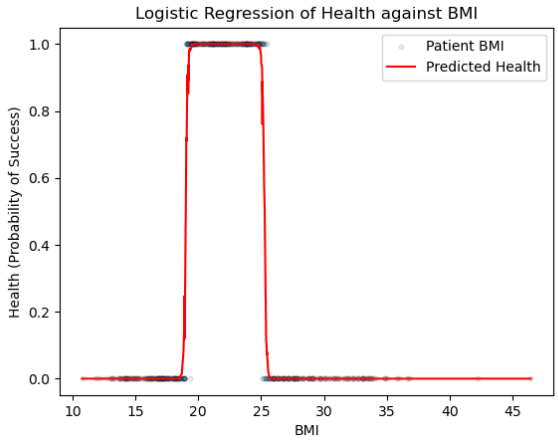

Figure 13: Logistic regression of patient health and BMI.

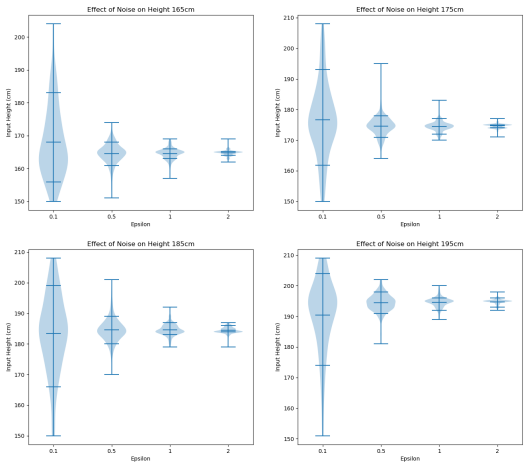

Figure 14: Distribution of height for different values of ϵ

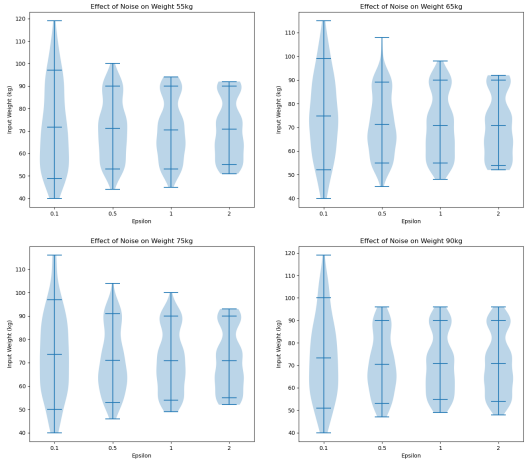

Figure 15: Distribution of weight for different values of ϵ.

sanitization, as the only sensitive attribute are height and weight, we use the metric DP mechanism. We calculate the BMI based on independently noised height and weight, whose distributions can be seen in Figure 14 and Figure 15 respectively.

As seen in Figure 13, BMI roughly 20 to 25 are predicted as healthy by the model. To determine whether changes to the BMI (via height and weight) and noise are significant, we perform logistic regression with the null hypothesis that the coefficients for the linear and quadratic terms of BMI, and that of noise, are zero. We find that BMI is statistically significant in determining the health of a patient as both, a linear and quadratic term (the P-values associated with their z-statistic is 0.000), and noise is not significant.