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

Amrita Roy Chowdhury[‖]*, David Glukhov[†]*, Divyam Anshumaan[‡]*,
Prasad Chalasani[§], Nicholas Papernot[†], Somesh Jha[‡] and Mihir Bellare[¶]
[‖]University of Michigan, Ann Arbor
[†]University of Toronto and Vector Institute
[‡]University of Wisconsin-Madison
[§]Langroid Incorporated
[¶]University of California, San Diego

***Abstract*—The rise of large language models (LLMs) has introduced new privacy challenges, particularly during *inference* where sensitive information in prompts may be exposed to proprietary LLM APIs. In this paper, we address the problem of formally protecting the sensitive information contained in a prompt while maintaining response quality. To this end, first, we introduce a cryptographically inspired notion of a *prompt sanitizer* which transforms an input prompt to protect its sensitive tokens. Second, we propose Pr$\epsilon\epsilon$mpt, a novel system that implements a prompt sanitizer, focusing on the sensitive information that can be derived solely from the individual tokens. Pr$\epsilon\epsilon$mpt categorizes sensitive tokens into two types: (1) those where the LLM's response depends solely on the format (such as SSNs, credit card numbers), for which we use format-preserving encryption (FPE); and (2) those where the response depends on specific values, (such as age, salary) for which we apply metric differential privacy (mDP). Our evaluation demonstrates that Pr$\epsilon\epsilon$mpt is a practical method to achieve meaningful privacy guarantees, while maintaining high utility compared to unsanitized prompts, and outperforming prior methods.**

I. Introduction

The recent advent of large language models (LLMs) have brought forth a fresh set of challenges for protecting users' data privacy. LLMs and their APIs present significant privacy concerns at *inference time*, which are fundamentally distinct from the well-documented risks of training data memorization [16, 40, 52, 85]. While the potential adversary in training data scenarios could be any API user, the threat during inference primarily stems from the model owner—typically the organization hosting the LLM. This inference stage poses a significant privacy risk, as *prompts* in natural language may include various types of sensitive information, from personally identifiable data like SSNs or credit card numbers to personal health or financial details.

*These authors contributed equally to this work.

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.231277
www.ndss-symposium.org

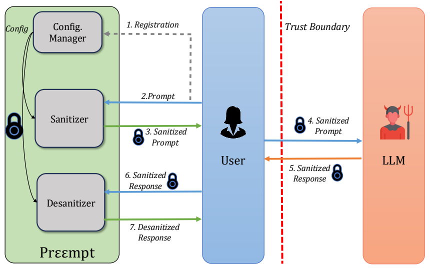

Fig. 1: Overview of Pr$\epsilon\epsilon$mpt: Users begin with a one-time registration to set up configurations, which are used in all subsequent interactions. Users can then submit prompts to Pr$\epsilon\epsilon$mpt and receive their sanitized versions, which are safe to be provided to the untrusted LLM. The LLM's responses (to the sanitized prompts) can then be desanitized to recover high-utility responses.

The ensuing privacy threat is exacerbated with the growing use of in-context learning, that involves presenting the LLM with a few training examples as part of the prompt during inference [14]. This has shifted some of the concerns around privacy of training data from training time to inference time. Furthermore, the consumer-facing nature and widespread accessibility [27, 62, 48, 80, 46] 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 users, leading to unwitting disclosure of sensitive information [10]. Consequently, certain countries, such as Italy [18], along with financial institutions [47, 88], government agencies [82, 63, 83], medical institutions [28] as well as companies, such as Samsung [77, 76], Amazon [4] and Apple [7], have prohibited the use of proprietary LLMs altogether, underscoring the significance of these privacy concerns.

Prior work on privacy-preserving LLMs [5, 54, 42, 73, 79, 60, 89], has primarily focused 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 risks that are not addressed by these mechanisms [29]. While recent research has begun to address the privacy of prompts, solutions based on homomorphic encryption and secure multi-party computation [39, 21, 43] are computationally expensive in practice, with state-of-the-art techniques taking over *sixteen* minutes for a single inference on BERT [70]. More efficient solutions either lack formal privacy guarantees [55, 91], require changes to current LLM APIs [58] or make impractical design choices [78, 49, 22] (see Sec. VI for more details).

To this end, we make the following contributions. First, we introduce a cryptographically inspired 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 a useful prediction. We provide a formal analysis of both its privacy and utility. Second, we propose Pr$\epsilon\epsilon$mpt[1], a system that instantiates a prompt sanitizer. To the best of our knowledge, Pr$\epsilon\epsilon$mpt is the *first* prompt sanitizer with formal privacy guarantees. As the first step in this direction, we focus on the sensitive information that can derived *solely* from the individual tokens. It is 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. This is because an adversary can exploit the sensitive tokens (such as SSN, credit card number) independently, without needing to process or access additional context from the prompts. Moreover, in many settings, sensitive information is often restricted to structured (e.g., tabular) data that can be extracted as tokens—for instance, in financial Q/A tasks as evaluated in our experiments. The task of handling privacy risks stemming from the contextual linguistic semantics[2] of the entire prompt [61, 13] 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). Consequently, we propose encrypting the former using format-preserving encryption [11]: a type of property-preserving encryption scheme where the ciphertext and the plaintext have the same format. For example, 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 using differential privacy (DP) [31], which is the state-of-the-art technique for achieving data privacy. Specifically, we employ a relaxation of DP, called metric DP [19]. Metric DP protects pairs of inputs that are "similar" based on a distance metric, meaning that the sanitized token will remain similar to the original token. This approach maintains the relevance of the responses generated to the original prompt while providing meaningful privacy guarantees.

We demonstrate the practicality of Pr$\epsilon\epsilon$mpt through empirical evaluation. Specifically, we evaluate four types of tasks: translation, retrieval augmented generation (RAG), long-context reading comprehension Q/A and multi-turn financial Q/A. We observe that Pr$\epsilon\epsilon$mpt's sanitization mechanism preserves the utility of responses across all tasks. For instance, the BLEU scores [72] for sanitized prompts are nearly identical compared to baseline unsanitized prompts for a German language translation task with GPT-4o. When prompted with Pr$\epsilon\epsilon$mpt sanitized prompts, all RAG tasks achieved 100% accuracy. Pr$\epsilon\epsilon$mpt is also quite successful in long-context and multi-turn conversation tasks. For example, responses based on Pr$\epsilon\epsilon$mpt processed reference texts used in long-context Q/A has a similarity score of 0.934 compared to responses based on unsanitized text, outperforming a contemporary method [81] (PAPILLON) without any additional overheads.

We provide a complete version of this paper, including references to the appendices at [25].

II. Background

Notation. Let V be the vocabulary (tokens) of a language model and V^* the set of possible strings over V (recall that a prompt and its response are strings over V). 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 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 $\mathbb{P}(\mathsf{V})$ denote the space of all probability distribution over V.

A. Language Model

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

[1]**Pr**ivacy-**Pre**serving Pro**mpt**

[2]This refers to cases where individual tokens may not be sensitive but, when considered in the context of the full prompt, could leak information because of the underlying natural language semantics.

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 $\perp$ (which we can be viewed as "end of sequence" (EOS) token), we sample a token σ from the distribution $f(\boldsymbol{\rho}, \boldsymbol{\sigma})$ (using a decoding algorithm, such as multinomial sampling, or greedy sampling of the single most likely next token) and append it to $\boldsymbol{\sigma}$. The algorithm stops once a special token $\perp$ is emitted. Once the decoding algorithm is fixed, we can model f as taking a prompt p in $\mathsf{V}^\star$ and outputting a string in $V^\star$. In a slight abuse of notation, we will henceforth use $f(\boldsymbol{\rho})$ to denote the response string of the LLM on the input prompt $\boldsymbol{\rho}$.

1) Tokens and Types: Given a sequence of tokens $\boldsymbol{\sigma} \in \mathsf{V}^\star$, a *typed sequence* is a 2-tuple $\boldsymbol{\sigma}_\mathcal{T} = \langle(\sigma_1, \tau_1), \cdots, (\sigma_n, \tau_n)\rangle$, where $\tau_i \in \mathsf{T}$ is the type of the substring σ_i of σ (we assume $\boldsymbol{\sigma} = \sigma_1 \cdot \sigma_2 \cdots \sigma_n$). Each type is associated with a domain. We also assume the existence of a *type annotator*.

Definition 2 (Type Annotator)**.** *A type annotator is a deterministic algorithm $\mathcal{M}_\tau : \mathsf{V}^* \mapsto (\mathsf{V}^\star \times \mathsf{T})^*$ that inputs a prompt $\boldsymbol{\rho}$ 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 $\perp$ (which denotes non-sensitive token). 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.

III. Prompt Sanitizer

Given an input prompt $\boldsymbol{\rho}$, a *prompt sanitizer* (denoted by PS) transforms the entire prompt to a sanitized one $\hat{\boldsymbol{\rho}}$ with the goal of protecting the sensitive tokens contained in $\boldsymbol{\rho}$. 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:*

- *Setup (S). The setup algorithm takes no input and outputs a secret key, as $\mathsf{K} \leftarrow \mathsf{S}$.*
- *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).*
- *Sanitization (E). The sanitization algorithm takes as input the secret key K and a type-annotated token sequence $\boldsymbol{\rho}_\tau \in (\mathsf{V}^\star \times \mathsf{T})^*$. It outputs a token sequence $\hat{\boldsymbol{\rho}} \in \mathsf{V}^{|\boldsymbol{\rho}_\tau|}$, as $\hat{\boldsymbol{\rho}} \leftarrow \mathsf{E}(\mathsf{K}, \boldsymbol{\rho}_\tau)$.*
- *Desanitization (D). Desanitization takes a string (token sequence) $\hat{\boldsymbol{v}} \in \mathsf{V}^*$ and processes it with the goal of reversing the effect of the sanitization algorithm, using the secret key K. This is represented as $\boldsymbol{v} \leftarrow \mathsf{D}(\mathsf{K}, \hat{\boldsymbol{v}})$ with $\boldsymbol{v} \in \mathsf{V}^{|\hat{\boldsymbol{v}}|}$.*

Given a prompt $\boldsymbol{\rho}$, the typical workflow of PS proceeds as follows:
(1) type annotate the prompt to obtain $\boldsymbol{\rho}_\tau = \mathcal{M}_\tau(\boldsymbol{\rho})$,
(2) sanitize the type annotated prompt using the secret key K as $\hat{\boldsymbol{\rho}} = \mathsf{E}(\mathsf{K}, \boldsymbol{\rho}_\tau)$,
(3) obtain the LLM's response on the sanitized prompt as $\hat{\boldsymbol{v}} = f(\hat{\boldsymbol{\rho}})$,
(4) desanitize the response to obtain $\boldsymbol{v} = \mathsf{D}(\mathsf{K}, \hat{\boldsymbol{v}})$.

In the above workflow, the desanitization algorithm restores information about the original prompt $\boldsymbol{\rho}$ in its output, $\boldsymbol{v}$. In the special case where we run the desanitization algorithm directly on the sanitized prompt $\hat{\boldsymbol{\rho}}$ (which can be useful for instance if the PS is used to store a set of sensitive prompts on an untrusted platform for later use), we ideally expect $\boldsymbol{v} = \boldsymbol{\rho}$.

We require that the sanitization and desanitization algorithms are *type preserving*, which 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 $(\tau_1, \cdots, \tau_n) = (\tau'_1, \cdots, \tau'_n)$.

A. Privacy Guarantee

The privacy game, denoted as $\mathbf{G}^{\text{pp}}_{\mathsf{PS},\mathcal{L}}$, is designed to capture an adversary's ability to distinguish between the sanitized outputs of two different prompts. In the game if the adversary picks two prompts that have a very different structure (e.g. different type or number of tokens), then the adversary can trivially distinguish between the corresponding sanitized prompts. To rule out pathological cases, we restrict the adversary to selecting two prompts with a "similar structure", formalized via a leakage function. Different instantiations of the leakage function lead to different instantiations of the game.

The game is defined as follows:

```
  $\underline{\text{INITIALIZE}}$:
1: $\mathsf{K} \leftarrow \mathsf{S}()$
2: $b \xleftarrow{\$} \{0,1\}$ $\triangleright$ Select a random bit
  $\underline{\text{SANITIZE}(\rho_0, \rho_1)}$: $\triangleright$ Adversary selects two prompts
3: $L_0 \leftarrow \mathcal{L}(\rho_0)$ ; $L_1 \leftarrow \mathcal{L}(\rho_1)$
  $\triangleright$ $\mathcal{L}$ is the leakage function associated with $\mathsf{PS}$
4: if $L_0 \neq L_1$ then return $\perp$
  $\triangleright$ Only prompt pairs with the same leakage are valid
5: $\hat{\rho}_0 \leftarrow \mathsf{E}(\mathsf{K}, \mathcal{M}_\tau(\rho_0))$ ; $\hat{\rho}_1 \leftarrow \mathsf{E}(\mathsf{K}, \mathcal{M}_\tau(\rho_1))$
6: return $\hat{\rho}_b$
  $\triangleright$ Return one of the sanitized prompts chosen at random
  $\underline{\text{FINALIZE}(b')}$:
7: return $[b' = b]$ $\triangleright$ $b'$ is the adversary's guess for $b$
```

We denote an adversary by $\mathcal{A}$. We model the information leakage from the sanitized prompts through a leakage function, $\mathcal{L}$. In particular, the leakage function $\mathcal{L}$ of a prompt sanitizer takes as input a prompt ρ and captures all the information about sensitive tokens that is leaked by $\hat{\rho} = \mathsf{E}(\rho, \mathsf{K})$, given a key K. In the above game, an adversary $\mathcal{A}$ aims to distinguish between the sanitized prompts of ρ_0 and ρ_1 based solely on the sanitized output and the leakage allowed by $\mathcal{L}$. The adversary is said to win the game if $b' = b$ and their advantage is formally defined as:

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

Intuitively, the game implies that even after observing a sanitized prompt, an adversary should not be able to reliably differentiate between two prompts with the *same* leakage. The definition of the leakage function, $\mathcal{L}$, is crucial and depends on the underlying sanitizer E. For example, if E sanitizes a token by redaction, for prompt ρ =“My age is 26”, the leakage function outputs all the non-sensitive tokens, i.e., $\mathcal{L}(\rho)$ = “My age is [ ]”—this represents the minimal possible leakage as redaction is the strongest sanitization mechanism. Alternatively, if E encrypts sensitive tokens, $\mathcal{L}$ might reveal the length of these tokens to $\mathcal{A}$. Note that leakage function is a standard notion in cryptography [33]. For instance, the leakage function for order-preserving encryption [57] is essentially the numerical ordering of the input dataset.

The restriction in the above game, requiring the pair of prompts to have the same leakage, aligns with standard notions in game-based cryptographic security definitions. For instance, this is similar to the definition of security in order-preserving encryption (IND-FA-OCPA [57]), where the adversary is restricted to selecting pairs of data sequences that maintain the same order.

B. Utility Guarantee

Let $\mathcal{Q} : \mathsf{V}^* \times \mathsf{V}^* \mapsto R_Q$ be a *quality oracle* that evaluates the quality of a candidate response $\boldsymbol{\upsilon}$ for a prompt ρ. Specifically, $\mathcal{Q}(\rho, \boldsymbol{\upsilon})$ is a measure of the response's goodness. Such a quality oracle has been used in prior work on LLMs [90].

Definition 4. *A prompt sanitizer PS satisfies (α, β) utility for a given prompt $\rho \in V^*$,*

$$\alpha = \mathbb{E}_f\Big[Q\big(\rho, f(\rho)\big)\Big] \tag{1}$$

$$\beta = \mathbb{E}_{f,PS}\Big[Q\Big(\rho, D\big(K, f(\hat{\rho})\big)\Big)\Big], \hat{\rho} = E\big(K, \mathcal{M}_\tau(\rho)\big) \tag{2}$$

where the randomness is defined over both the LLM, f, and PS.

The utility of the prompt sanitizer PS is evaluated by comparing the quality of the original response $f(\rho)$ with the one obtained through the PS pipeline. The above definition has two key characteristics. First, the utility is defined w.r.t to a specific prompt, as response quality can vary significantly across different prompts. For example, consider the following prompt: ρ = *“My age is 46. What is the average age of the population of New York?”* Here, a high-quality LLM's response should be invariant to the sensitive token ([*Age*]) in the prompt. This means that even after sanitization, we should be able to retrieve a correct and relevant response. On the other hand, for a conversational prompt used in a LLM-based chatbot to seek medical advice, the quality of the responses could vary significantly based on the specifics of the sanitization and desanitization algorithms of PS. Note that the quality oracle Q can take various forms based on the type of the prompt. For instance, it might be a human evaluator who assigns a quality score, or it could be a predefined analytical expression in case the prompt has some special structures. Second, utility is defined as an expectation, since in general both the LLM f and the prompt sanitizer PS, are probabilistic. Note that when the distribution of $f(\rho)$ matches the distribution of $\mathsf{D}(\mathsf{K}, f(\hat{\rho}))$, this represents the strictest form of utility. If R_Q is a metric space with a distance metric d_Q, we can quantitatively measure the mean degradation in the quality of the response as $d_Q(\alpha, \beta)$.

IV. Preεεmpt System Description

This section introduces Preεεmpt: a system that instantiates a sanitizer for prompts. First, we describe the primitives (FPE and mDP) in subsection IV-A. Next, two sections describe our threat model and design goals. Our system is described in subsection IV-D. We conclude the section with privacy and utility analysis. Our last subsection discusses other "strawman" solutions and why they don't address our threat model and goals.

A. Building Blocks

We start by describing the building blocks which will be used to sanitize the sensitive tokens.

1) Format-Preserving Encryption (FPE): Under a format preserving encryption (FPE) scheme, the plaintext and the ciphertext have the same format, that is, FPE ensures that the encrypted output is structurally similar to the original input, including properties such as length, character set, or format. This property allows applications to process ciphertexts and plaintexts in the same way. This backward compatibility makes FPE a popular tool for secure data analytics in practice. For instance, the ciphertext of a 16-digit credit card number encrypted under a FPE scheme would also be a 16-digit number[3]. As concrete examples, a plaintext Social Security number such as 055-46-6168 might be transformed into the ciphertext 569-83-4469, while an IP address like 76.217.83.75 could be encrypted as 97.381.64.35. Intuitively, FPE shuffles or re-encodes values within the space of all valid values of the same type. Without the decryption key, the ciphertext looks indistinguishable from any legitimate value of that format, effectively hiding the original data while remaining compatible with systems that expect specific formats.

Definition 5 (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 $x \leftarrow \mathsf{D}_{\mathcal{F}}(\mathsf{K}, \mathsf{N}, y)$.*

[3]One can add additional constraints, such as ensuring the last digit is the Luhn checksum or the first four digit corresponds to a bank.

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

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.

2) Metric Local Differential Privacy (mLDP): Differential privacy (DP) is a quantifiable measure of the stability of the output of a randomized mechanism to changes to its input. As a direct consequence of our threat model (Sec. IV-B), we work with the local model of DP (LDP) where each data point is individually randomized. Metric local differential privacy (mLDP) [19, 3, 75, 45] 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 6 (Metric Local Differential Privacy (mLDP) [19])**.** *A randomized algorithm $\mathcal{M} : \mathcal{X} \rightarrow \mathcal{Y}$ 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, $\mathcal{O} \subseteq \mathcal{Y}$*

$$\Pr\big[\mathcal{M}(x) \in \mathcal{O}\big] \leq e^{\epsilon d(x,x')} \cdot \Pr\big[\mathcal{M}(x') \in \mathcal{O}\big] \tag{3}$$

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; that is, only data points that are "close" to each other should be indistinguishable. Still, mLDP captures the privacy semantics of many real-world scenarios and is well suited to settings where releasing approximate information is acceptable. For example, it is often sufficient to reveal a coarse location, such as a city block, rather than exact GPS coordinates. Similarly, sharing an income range (e.g., \$60K–\$80K) can preserve utility without exposing precise figures. Alg. 3 in App. 1 of [25] outlines a mechanism for achieving ϵ-mLDP for the ℓ_1 distance using a variant of the exponential mechanism [31].

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

The proof of the theorem is standard and appears in the appendix. However, we next justify why $\mathcal{M}_{\epsilon}$ is an appropriate notion for our context. An input is more likely to be mapped to one which is close to it, which we formalize this using the following two properties.

Property 1.

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

Property 2.

$$|y_1 - x| < |y_2 - x| \iff \Pr\big[\mathcal{M}_\epsilon(x, \epsilon, [k]) = y_1\big] > \Pr\big[\mathcal{M}_\epsilon(x, \epsilon, [k]) = y_2\big], \quad \forall y_1, y_2 \in [k]$$

B. Threat Model

Pr$\epsilon\epsilon$mpt runs as an application on a user's (trusted) local device. Additionally, Pr$\epsilon\epsilon$mpt can support multiple users: envision it as an application maintained at the level of an organization and available to all of its employees. 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 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 $\hat{\boldsymbol{v}}$ from an LLM on $\hat{\boldsymbol{\rho}}$ and again uses Pr$\epsilon\epsilon$mpt to desanitize it into $\boldsymbol{v}$. This interaction constitutes two separates Pr$\epsilon\epsilon$mpt sessions: one for the $\boldsymbol{\rho} \rightarrow \hat{\boldsymbol{\rho}}$ transformation and the other for the $\hat{\boldsymbol{v}} \rightarrow \boldsymbol{v}$ transformation. The LLM is an untrusted third-party application which represents the adversary (Fig. 1).

In Pr$\epsilon\epsilon$mpt, we focus on tokens where the sensitive information can be derived *solely* from the individual token, with no extra context. Examples of such token types include SSN, credit card number, license number, age, money, A/C number, zipcode. Privacy issues stemming from the linguistic context of the prompts (e.g. a prompt indicating the users' mental health details) are beyond Pr$\epsilon\epsilon$mpt's scope (see App. 9 in [25] for more discussion).

C. Design Goals

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

- **Formal Guarantees.** Pr$\epsilon\epsilon$mpt should be able to provide a formal privacy guarantee on the sanitized prompts.
- **High Utility.** The responses based on sanitized prompts should be "close" to the responses based on the original prompt.
- **Stateless.** Finally, the sanitization and desanitization process should be stateless, i.e., 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 [38] and California's CCPA [15]. Additionally, these regulations grant individuals the Right to Deletion, allowing data owners to retract authorization previously granted for the use of their personal data. A stateful solution hinders the Right to Deletion and desanitization, while a stateless one offers flexibility and storage efficiency. For example, consider these two user action sequences:

$$A_1 = \langle \text{ Sanitize } \boldsymbol{\rho}_1;\ \text{Desanitize } \hat{\boldsymbol{v}}_1;\ \text{Sanitize } \boldsymbol{\rho}_2;\ \text{Desanitize } \hat{\boldsymbol{v}}_2;\ \text{Sanitize } \boldsymbol{\rho}_3;\ \text{Desanitize } \hat{\boldsymbol{v}}_3 \rangle$$

$$A_2 = \langle \text{ Sanitize } \boldsymbol{\rho}_1;\ \text{Sanitize } \boldsymbol{\rho}_2;\ \text{Desanitize } \hat{\boldsymbol{v}}_2,\ \text{Sanitize } \boldsymbol{\rho}_3,\ \text{Desanitize } \hat{\boldsymbol{v}}_1,\ \text{Desanitize } \hat{\boldsymbol{v}}_3 \rangle.$$

  Without perpetual retention of state information, a stateful solution restricts a user to a specific action sequence of sanitizing and desanitizing *in order* (such as, A_1). Moreover, multiple desanitization 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).

Note that while Pr$\epsilon\epsilon$mpt is stateless, conversation with the LLM can be stateful – the LLM is free to maintain a history of all (sanitized) prompts to better respond to user queries. We illustrate this experimentally in Sec. V.

D. System Modules

Pr$\epsilon\epsilon$mpt supports three types of sessions, namely, User Registration, Sanitization and Desanitization, which is taken care of by the following modules.

Configuration Manager. The configuration manager module of Pr$\epsilon\epsilon$mpt generates a secret key $\mathsf{K}_U \rightarrow \mathsf{G}(1^\kappa)$ for a given security parameter for an user U at the time of the registration for a FPE scheme $\mathcal{E}$. Subsequently, for any session involving user U, this module initializes all instances of the FPE scheme with the key K_U. Additionally, during registration, user U specifies the privacy parameter ϵ for $\mathcal{M}_\epsilon$, which is treated as the privacy budget for each individual sanitization session. The module also initializes the data domain (equivalently, format in the case of FPE) for each sensitive token type. The domains can either be predefined or computed based on some user-provided information. Lastly, various parameters (e.g. format and privacy parameter) can be dependent on the type τ.

Sanitizer. Recall that Pr$\epsilon\epsilon$mpt only sanitizes sensitive tokens that are alphanumeric or numeric (see Sec. IV-B). To this end, Pr$\epsilon\epsilon$mpt assumes that such sensitive tokens fall into two distinct categories:

- *Category I* (τ_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 names, Social Security Numbers (SSN), credit card numbers, Taxpayer Identification Numbers (TIN), passport numbers, bank account numbers, driver's license numbers, phone numbers, license numbers, and IP addresses. We provide empirical evidence of this assumption in Sec. V.
- *Category II* (τ_{II}). This category encompasses tokens where the LLM's response hinges on the specific *numerical value* itself, such as age, medical records[5], etc. That is, the LLM performs specific computations based on the *values* of these tokens.

Algorithm 1 Preϵmpt: Sanitization

```
    Input: $\boldsymbol{\rho}$ - Input prompt; $\mathsf{K}_U$ - Sanitization key; $\epsilon$ - Total budget
    Output: $\hat{\boldsymbol{\rho}}$ - Sanitized prompt;
 1: $\boldsymbol{\rho}' = \langle\rangle$
 2: $\boldsymbol{\rho}_\tau \leftarrow \mathcal{M}_\tau(\boldsymbol{\rho})$ ▷ $\mathcal{M}_\tau$ is instantiated with a named-entity recognizer
 3: $(\psi, t) \leftarrow \mathcal{M}_{\mathsf{Pre}}(\boldsymbol{\rho}, \boldsymbol{\rho}_\tau)$ ▷ $\Psi$ is a helper string encoding some extra information about the type of tokens
    ▷ $t$ is the number of tokens in $\boldsymbol{\rho}_\tau$ with type $\tau_{II}$
 4: for $(\sigma, \tau) \in \boldsymbol{\rho}_\tau$
 5:     if $(\tau \neq \perp)$
 6:         if $(\tau == \tau_{\mathrm{I}})$
 7:             $\hat{\sigma} = \mathsf{E}_{\mathcal{F}}(\mathsf{K}_U, \mathsf{N}_\tau, \sigma)$ ▷ $\mathsf{N}_\tau$ is the format of $\sigma$
 8:         else
 9:             $\hat{\sigma} = \mathcal{M}_\epsilon(\sigma, \frac{\epsilon}{t}, k_\tau)$ ▷ $[k_\tau]$ is the domain of $\sigma$
10:         end if
11:     else
12:         $\hat{\sigma} = \sigma$
13:     end if
14:     $\boldsymbol{\rho}'.append(\hat{\sigma})$
15: end for
16: $\hat{\boldsymbol{\rho}} \leftarrow \mathcal{M}_{\mathsf{Post}}(\boldsymbol{\rho}', \Psi)$ ▷ Performs some post-processing on the sanitized tokens
17: return $\hat{\boldsymbol{\rho}}$
```

A prompt is sanitized as follows. We first perform type annotation of the different tokens via $\mathcal{M}_\tau$. In addition to annotating the type of a token, $\mathcal{M}_\tau$ also indicates its category. For Preϵmpt, we instantiate $\mathcal{M}_\tau$ with a named-entity recognizer (NER). Next, Preϵmpt uses a pre-processor $\mathcal{M}_{\mathsf{Pre}}$ that takes $(\boldsymbol{\rho}, \boldsymbol{\rho}_\tau)$ as input and computes two things; 1) it determines the number of tokens belonging to the second category, denoted as t, 2) it computes a **helper string** Ψ to encode additional information about token types and provide flexibility during sanitization. Specifically, Ψ captures functional dependencies between the tokens. We present examples involving the helper string in Sec. IV-E and App. 7 of [25].

[5]Note that in some cases, for instance, language translation, all sensitive tokens are of τ_{I} since the LLM's response should not depend on the specific values.

In Preϵmpt, each sensitive token is sanitized individually. In particular, all tokens of the first category are sanitized using FPE with the user specific secret key K_U. On the other hand, all tokens of the second category are sanitized to satisfy $\frac{\epsilon}{t}$-mLDP using $\mathcal{M}_\epsilon$, where ϵ is the privacy parameter for the standard DP guarantee and t is the maximum distance between protected values. No operation is performed on tokens with non-sensitive types ($\tau = \perp$). Next, all sanitized tokens are concatenated and passed to a post-processor $\mathcal{M}_{\mathsf{Post}}$. The $\mathcal{M}_{\mathsf{Post}}$ enforces the functional dependencies encoded in Ψ. Furthermore, in Preϵmpt only the determinant is perturbed, and the sanitized versions of the dependent tokens are derived from this noisy encoding. If Ψ is empty (i.e. not functional dependencies are provided), then each token is handled independently and no functional dependencies are enforced, which can adversely affect utility. The full sanitization mechanism is outlined in Algorithm 1. Steps 3-17 in Alg. 1 instantiate the sanitization algorithm E of the prompt sanitizer (Def. 3).

Algorithm 2 Preϵmpt: Desanitization

```
    Input: $\hat{\boldsymbol{\upsilon}}$ - Input sanitized response; $\mathsf{K}_U$ - Sanitization key;
    Output: $\boldsymbol{\upsilon}$ - Desanitized response;
 1: $\boldsymbol{\upsilon} = \langle\rangle$
 2: $\hat{\boldsymbol{\upsilon}}_\tau \leftarrow \mathcal{M}_\tau(\hat{\boldsymbol{\upsilon}})$ ▷ $\mathcal{M}_\tau$ is instantiated with a named-entity recognizer
 3: for $(\sigma, \tau) \in \hat{\boldsymbol{\upsilon}}_\tau$
 4:     if $(\tau == \tau_I)$
 5:         $\sigma = \mathsf{D}_{\mathcal{F}}(\mathsf{K}_U, \mathsf{N}_\tau, \sigma)$ ▷ $\mathsf{N}_\tau$ is the format of $\sigma$
 6:     else
 7:         $\sigma = \hat{\sigma}$
 8:     end if
 9:     $\boldsymbol{\upsilon}.append(\sigma)$
10: end for
11: return $\boldsymbol{\upsilon}$
```

Desanitizer. Desanitization (Alg. 2) begins with the same type annotator. All sensitive tokens of category I can be desanitized using the decryption algorithm of the FPE scheme. However, tokens sanitized with $\mathcal{M}_\epsilon$

cannot be desanitized without retaining additional state information and are hence, left untouched by default. Steps 3-11 in Algorithm 2 correspond to the desanitization algorithm D of the prompt sanitizer (Def. 3).

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 auxiliary information. In this scenario, Pr$\epsilon\epsilon$mpt will exclusively desanitize tokens that appeared in the prompt. Note that the only thing required to desanitize is the secret key K_U: Pr$\epsilon\epsilon$mpt does not store any sensitive information post the termination of a session thereby making our solution stateless.

E. Setting up Parameters

In this section, we provide guidelines on how to choose the parameters for Pr$\epsilon\epsilon$mpt.

Setting up ϵ. If a user desires a privacy parameter of ϵ for the standard DP guarantee and wishes to protect values that differ by at most distance l, then use ϵ'-mLDP with $\epsilon' = \frac{\epsilon}{l}$. Any ϵ-mLDP protocol is $\epsilon \cdot l_{max}$-DP where the distance between any pair of inputs is at most l_{max}.

Using the helper string Ψ. The helper string Ψ encodes auxiliary knowledge that captures functional dependencies among sensitive tokens. These dependencies typically fall into two categories: (1) *Common knowledge*, such as mathematical or definitional identities (e.g., *Annual Salary* $= 12\times$*Monthly Salary*), and (2) *Domain-specific knowledge*, such as medical or financial formulas (e.g., *BMI* $= \frac{\textit{Weight}}{\textit{Height}^2}$).

For common knowledge, the dependencies can be stored in a knowledge base and appended directly to the prompt to aid reasoning or consistency. Since these are standardized and widely accepted relationships, they can be programmatically injected with little ambiguity.

Handling domain-specific dependencies is more complex. One possible approach is to use a local LLM to infer dependencies between sensitive attributes identified by NER, producing structured representations of causal or computational links. These dependencies can then be modeled as a directed acyclic graph (DAG), where each node corresponds to a sensitive attribute and edges represent dependencies (e.g., computation or inference). The root nodes represent the base sensitive values and are directly noised using mLDP. These values are then propagated along the DAG via the encoded dependencies, ensuring that related fields (such as income and tax, or weight and BMI) are sanitized consistently.

F. Privacy and Utility Analysis

Privacy Analysis. The formal privacy guarantee of Pr$\epsilon\epsilon$mpt is given as follows:

Theorem 2. *Let S be the set of all token pairs of type τ_{II} that are different in the prompt pairs $(\boldsymbol{\rho}_0, \boldsymbol{\rho}_1)$ in the privacy game $\mathbf{G}^{\mathrm{pp}}_{\mathsf{PS},\mathcal{L}}$. Then, for Pr$\epsilon\epsilon$mpt we have:*

$$\mathbf{Adv}^{\mathrm{pp}}_{\mathit{Pr\epsilon\epsilon mpt},\mathcal{L}}(\mathcal{A}) \leq e^{l\epsilon} + negl(\kappa) \tag{4}$$

where $l = \max_{(\sigma_0,\sigma_1)\in S}\{|\sigma_0 - \sigma_1|\}$ and κ is the security parameter of the underlying FPE scheme.

Proof Sketch. First, we compute the adversary's advantage in Pr$\epsilon\epsilon$mpt when the two prompts $(\boldsymbol{\rho}_0, \boldsymbol{\rho}_1)$ differ by only a single token, denoted as $\mathbf{Adv}^{\mathrm{pp}}_{\mathrm{Pr\epsilon\epsilon mpt},\mathcal{L}=1}(\mathcal{A})$. Next, using the classic hybrid argument [66], we establish an upper bound on the adversary's advantage in the general case, expressed in terms of its advantage when the prompts differ by just a single token. Finally, Eq. 4 can be derived by substituting $\mathbf{Adv}^{\mathrm{pp}}_{\mathrm{Pr\epsilon\epsilon mpt},\mathcal{L}=1}(\mathcal{A})$ into this result. The full proof is presented in App. 10 of [25].

Practical Privacy Considerations.
Error due to NER. Pr$\epsilon\epsilon$mpt's privacy guarantee is cryptographic and *orthogonal* to NER. Specifically, Pr$\epsilon\epsilon$mpt uses NER as a black-box and the above theorem aligns with the standard $\mathcal{F}_{\mathrm{NER}}$-hybrid model of security where $\mathcal{F}_{\mathrm{NER}}$ is an ideal functionality for NER [67]. Importantly, the performance of NER should *not* be conflated with the efficacy of the sanitization scheme. While the practical performance depends on properly identifying sensitive tokens, this is inherent to our task. As NER continues to improve, so will Pr$\epsilon\epsilon$mpt's practical performance *without* any modifications to the design. Additionally, domain-specific pattern matching with regular expressions can achieve high performance for structured data.

Nevertheless, we provide Theorem 4 (App. A), which formalizes how to analyze privacy in the presence of NER errors. The main idea is to model errors within the leakage function $\mathcal{L}$ of our privacy game $\mathbf{G}^{\mathrm{pp}}_{\mathsf{PS},\mathcal{L}}$. If the false negative rate of the NER is λ, we define a leakage function $\mathcal{L}_{\mathrm{NER}}$ that additionally leaks up to $\lambda\%$ of the sensitive tokens. To construct the two prompts $\boldsymbol{\rho}_0$ and $\boldsymbol{\rho}_1$ for the corresponding privacy game $\mathbf{G}^{\mathrm{pp}}_{\mathsf{PS},\mathcal{L}_{\mathrm{NER}}}$, we proceed as follows: we start with the original prompt pair as in the unmodified game $\mathbf{G}^{\mathrm{pp}}_{\mathsf{PS},\mathcal{L}}$, and then modify $\boldsymbol{\rho}_1$ by replacing $\lambda\%$ of its sensitive tokens with the corresponding tokens from $\boldsymbol{\rho}_0$. In other words, this construction models the scenario where the adversary gains access to a fraction of sensitive tokens that were missed by the NER.

Correlated Tokens. To mitigate correlation attacks, Pr$\epsilon\epsilon$mpt adopts a conservative approach by dividing the privacy budget equally among *all* tokens of type τ_{II}. This ensures that, via composition, the total privacy loss remains bounded by ϵ. However, leveraging the helper string Ψ can yield a better privacy-utility tradeoff. For example, in the prompt, "My age is X, I was born in Y. I am X years old.", $[Age : X]$, $[Year : Y]$ and $[Age : X]$ are the sensitive tokens. By default, Pr$\epsilon\epsilon$mpt distributes the privacy budget equally (ϵ/3) among all type τ_{II} tokens, however, the helper string Ψ can indicate that X and Y represent the same ground-truth and that X is repeated. Using Ψ, Pr$\epsilon\epsilon$mpt applies ϵ-mLDP to the first occurrence of X, yielding $\hat{X} = 25$ (suppose). Pr$\epsilon\epsilon$mpt then derives the corresponding $\hat{Y} = 2000$ by post-processing and reuses $\hat{X}$ for the second occurrence of age. This incurs no additional privacy loss due to the post-processing immunity of mLDP [32]. Thus, the resulting sanitized prompt is given by: "My age is 25, I was born in 2000. I am 25 years old.". We present additional illustrative examples in App. 7 of [25].

Utility Analysis. We analyze Pr$\epsilon\epsilon$mpt's utility below.
Prompts with Perfect Utility. Recall that, for assessing utility, we compare the responses of the LLM to the original prompt $\boldsymbol{\rho}$ and the sanitized prompt $\hat{\boldsymbol{\rho}}$ produced by Pr$\epsilon\epsilon$mpt. For many practically useful prompts, the response of the LLM remains the *same* for both $(\boldsymbol{\rho}, \hat{\boldsymbol{\rho}})$ except for the substitution of the sensitive tokens $\sigma \in \boldsymbol{\rho}$ with their sanitized counterparts $\hat{\sigma}$. In other words, the sanitized response $\hat{\boldsymbol{\upsilon}}$ generated from $\hat{\boldsymbol{\rho}}$ preserves perfect utility (after desanitization). We refer to such prompts as *invariant prompts*, where the LLM's response should be invariant to the specific values (or small variations) of the sensitive tokens. This property holds in particular for prompts containing only type τ_{I} tokens. Translation is exemplar: all sensitive tokens will be classified as type τ_{I} and sanitized using FPE since the LLM's translation should not depend on their specific values. As a result, sanitized tokens can be perfectly desanitized from the translated text. The quality score (the output of Q) can be evaluated using metrics such as BLEU score [71].

We now turn to the case of invariant prompts that include sensitive tokens of type τ_{II}. One such example is a factual information retrieval task for RAG. Consider the following prompt in the context of financial documents: "Please return all bank accounts with balance greater than \$2000." Here the two sensitive tokens [*Bank A/C*] and [*Bank Balance*] are sanitized via FPE and mLDP, respectively. mLDP, by construction, noisily maps an input to a value that is close to it (as per Properties 1 and 2 in Sec. IV-A2). As a consequence, the bank balance is perturbed only slightly, allowing correct numeric comparisons with high probability. This is precisely the rationale for our choice of mLDP: sanitized tokens preserve ordinal relationships and remain close to their original values, enabling useful computations while still providing strong privacy guarantees. The quality score here is the accuracy of the answers (count of the correct bank A/Cs returned). The above discussion is validated by our experimental results in Sec. V. Formally, we have:

Theorem 3. *For invariant prompts, Pr$\epsilon\epsilon$mpt satisfies* (α, α)*- utility where* $\alpha = \mathbb{E}_f\Big[Q\big(\boldsymbol{\rho}, f(\boldsymbol{\rho})\big)\Big] = \mathbb{E}_{f,Pr\epsilon\epsilon mpt}\Big[Q\Big(\boldsymbol{\rho}, \mathsf{D}_{Pr\epsilon\epsilon mpt}\big(\mathsf{K}, f(\hat{\boldsymbol{\rho}})\big)\Big)\Big]$.

Other Prompts. Given the complex and open-ended nature of prompts and responses, it is challenging to assign a utility score for any general prompt. Nevertheless, we provide some guidelines for when Pr$\epsilon\epsilon$mpt is likely to perform well. Note that Pr$\epsilon\epsilon$mpt introduces only small perturbations in the sanitized prompt $\boldsymbol{\rho}$. Hence, intuitively, Pr$\epsilon\epsilon$mpt should perform well where small changes in the original prompt result in only *limited* changes to the generated response. There can be two natural ways to capture these changes in the response. First, consider cases where the prompt satisfies Lipschitz continuity [65], as given by $d_{\mathsf{V}}(f(\boldsymbol{\rho}), f(\hat{\boldsymbol{\rho}})) \leq K d_{\mathsf{V}}(\boldsymbol{\rho}, \hat{\boldsymbol{\rho}})$ for some $K \in \mathbb{R}_{>0}$ and distance metric $d_{\mathsf{V}} : \mathsf{V}^* \times \mathsf{V}^* \mapsto \mathbb{R}_{>0}$. Distances defined over a document embedding space could be apt for d_{V}. For example, when using an LLM as a financial advisor with a prompt "My monthly salary is \$12,000. Suggest a monthly savings plan.", the response should ideally remain consistent (and hence, very close in the embedding space) even if the salary value is slightly altered to \$11,500 (via $\mathcal{M}_\epsilon$). A second way of bounding changes in the response is when the operations of the LLM on the sensitive tokens can be expressed as a symbolic computation. For example; "My height is 158 cm and weight is 94lb. Compute my BMI." The BMI is computed via a fixed formula (i.e., a symbolic computation). These type of prompts ensure that the responses on $\hat{\boldsymbol{\rho}}$ can deviate from the original response in only a well-structured and predictable manner. Additionally, if this symbolic mapping is known, Pr$\epsilon\epsilon$mpt could leverage this information during desanitization to improve utility further.
Usability. A key advantage of Pr$\epsilon\epsilon$mpt is its ease of use: after type annotation, Pr$\epsilon\epsilon$mpt employs predefined sanitizers to protect sensitive tokens without any manual configuration of custom rules or execution of ad hoc san-

itization strategies. In the current prototype, Ψ is treated as an optional user input. Importantly, while Ψ can be used to improve performance—for example, through better privacy budget allocation—omitting it does not affect the privacy guarantees of Pr$\epsilon\epsilon$mpt. As discussed in Sec. IV-E, Ψ can also be automatically generated from the input prompt to capture functional dependencies between sensitive tokens, further enhancing usability.

G. Comparison with Strawman Solutions

Strawman Solution I: Redaction. One intuitive solution is to redact all sensitive tokens from the prompt. While this approach ensures perfect privacy, it severely impacts utility dependent on those sensitive tokens and can sometimes lead to a complete loss of functionality. We present an example in App. 5.1 of [25].

Strawman Solution II: Substitution. An intuitive solution is to replace sensitive tokens with others of the same type using a lookup table. However, this method is not stateless, as desanitization requires access to the lookup tables, leading to scalability and security issues (see Sec. IV-C). The table size grows linearly with the number of sensitive tokens, and a separate table is needed for each user to prevent information leakage. In contrast, Pr$\epsilon\epsilon$mpt only requires a fixed-size key per user, regardless of prompt number or length.

Strawman Solution III: Suppression. Consider the following sanitization strategy for the tokens of the second category ($\tau_{\texttt{II}}$): a numerical token is sanitized by simply setting its k lowest order digits to 0, the intuition being 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, it is difficult to formally quantify its privacy guarantees. Prior work shows that such ad-hoc approaches are often vulnerable to attacks [30, 26, 64]. In contrast, the mLDP-based approach used in Pr$\epsilon\epsilon$mpt offers a principled way of balancing this privacy/utility trade-off.

Additional Baseline: LLMs assisted obfuscation and deobfuscation. One could also attempt to use a LLM to obfuscate and deobfuscate sensitive information based on rules in the system prompt, and maintaining a state to recover information, such as [81]. However, the probabilistic nature of the LLM and lack of specifications preclude any rigorous privacy analysis.

V. Experiments

We evaluate the following questions:

Q1. How does Pr$\epsilon\epsilon$mpt impact utility of realistic tasks compared to unsanitized performance?
Q2. How does Pr$\epsilon\epsilon$mpt compare against prior LLM based sanitization approaches?
Q3. What is the impact of different technical design choices on Pr$\epsilon\epsilon$mpt's utility?

A. Utility Loss from Pr$\epsilon\epsilon$mpt Sanitization

We tackle **Q1** by applying Pr$\epsilon\epsilon$mpt to four tasks: translation, retrieval-augmented generation (RAG), multi-turn financial question answering (Q/A), and long-context reading comprehension Q/A. These tasks represent a broad spectrum of real-world LLM applications where input prompts are likely to contain sensitive information.

Models. We use GPT-4o [68], Gemini-1.5 [36], and OPUS-MT for translations, RAG, and question-answering tasks. For named entity recognition (NER), we use Uni-NER [92], Llama-3 8B Instruct [2], and Gemma-2 9B Instruct [37]. We also use Llama-3 as a Q/A model for the long-context task.

Translation. Translation is a common use case for language models. However, business or bureaucratic emails containing sensitive information face major privacy concerns pertaining to leakage of sensitive information [56]. For this task, we employ an LLM for named entity recognition (NER) of sensitive tokens belonging to the types of ([*Name*], [*Age*] and [*Money*]). We evaluate Pr$\epsilon\epsilon$mpt's performance on 50 English-French and English-German samples obtained from WMT-14 [12] dataset. These samples are single sentences containing one or two PII values. We seek exact translations in this context and use BLEU scores as the quality oracle $\mathcal{Q}$ to assess their similarity to reference translations. We use FPE to sanitize [*Name*] and [*Money*], and use *mLDP* for [*Age*].

Results. We report the BLEU scores for the translations of the original sentences and the ones obtained via Pr$\epsilon\epsilon$mpt in Table III. We observe that the BLEU scores are nearly identical in both cases, with only minor differences due to the performance variation of the translation model, nuances of language and NER. Details regarding the statistics of PII values, NER performance, details regarding encryption and ablations with larger privacy budgets can be found in App. 4.2 of [25]. We present comparisons with Papillon [81], a contemporary privacy preserving framework in Section V-B.

Retrieval-Augmented Generation (RAG). Retrieval-augmented generation is also commonly employed for a variety of LLM use cases [35], including extraction of information from potentially sensitive documents. Typically, documents are split, embedded, and indexed in a vector database. At query time, the most relevant shards are retrieved based on lexical and semantic similarity, then provided as context to the LLM for answer generation. Our experiments focus on this final step: generating answers given a query and its relevant context. We consider two types of question-answering scenarios: numerical comparisons, and retrieval of factual information. We assess these settings by using GPT-4 [68] to generate tuples of *Context* C, questions *Questions* Q, and answers *Answers* A; jointly sanitizing C and Q so that copies of the same sensitive attribute appearing in both C and Q are replaced with the same token, and comparing the desanitized LLM responses with A. Our numerical comparison questions involve comparing credit card balances and determining which is higher, while factual retrieval questions require returning specific aspects about a generated e-commerce order. The quality oracle $\mathcal{Q}$ is the accuracy of the answers.

Results. We observe that Pr$\epsilon\epsilon$mpt achieves 100% accuracy for both the RAG tasks. Additional experimental details can be found in App. 4.5 of [25].

Long-Context Q/A. LLMs can be tasked with not only retrieving specific information from long documents, but also integrating and reasoning about the information they contain. To simulate this, we use NarrativeQA [51], a long-context, reading comprehension task. We answer questions about book or movie summaries, including character-related queries, using only the provided context. Character names are treated as sensitive attributes and sanitized using FPE. Summaries have 534 words on average with a standard deviation of 210.

To assess the impact of Pr$\epsilon\epsilon$mpt on reasoning and reading comprehension, we use semantic textual similarity (STS) [74] between the answers based on the original summaries and the answers based on Pr$\epsilon\epsilon$mpt summaries. This score also acts as the quality oracle $\mathcal{Q}$. We do not use BLEU scores for this experiment, as the reference answers only have a few words and do not capture any paraphrased response. We discuss the performance of NER and present examples in App. 4.5 of [25]. We present comparisons with Papillon [81], a contemporary privacy preserving framework in Sec. V-B.

Results. We report the STS scores in Table I. We find that Pr$\epsilon\epsilon$mpt captures a significant amount of semantics of the plaintext response, with the GPT-4o response having an STS score of 0.934. For context, if we don't desanitize the LLM response, the score drops to 0.523 with respect to the plain responses. If the LLM gives a completely irrelevant response (such as the answer to an unrelated question), the score drops to around 0.146.

TABLE I: Semantic Textual Similarity scores of different methods for the Long-Context Q/A task. Higher value implies more similarity with the reference answer. "Plain Responses" refer to the responses for unsanitized inputs, and "References" indicate the ground truth responses. We find that Pr$\epsilon\epsilon$mpt has a particularly high utility with respect to GPT-4o, outperforming prior methods. Pr$\epsilon\epsilon$mpt uses Gemma-2 9B Instruct as the NER model for Gemini-1.5, and UniNER for Llama-3 and GPT-4o.

STS Score	**Pr$\epsilon\epsilon$mpt**			**Papillon**
	Llama-3	Gemini-1.5	GPT-4o	GPT-4o
Plain Responses	0.839	0.849	0.934	0.854
References (GT)	0.514	0.722	0.510	0.458

This demonstrates the robustness of the metric. Further details and ablations can be found in App 4.5 of [25].

Multi-Turn Financial Q/A. LLMs are also frequently used in multi-turn conversational settings, and may be tasked with performing numerical reasoning over sensitive information. Thus, we assess Pr$\epsilon\epsilon$mpt on a financial multi-turn question answering benchmark ConvFinQA [23].The dataset consists of financial reports written by experts [24] followed by a sequence of conversational, numerical-reasoning questions guiding the model through solving a multi-step problem. Each prompt includes background text and a table with yearly financial data spanning over several years. All numerical information (except years) is extracted using regex and sanitized using mLDP. To handle repeated values within parentheses in the table, we use the helper string Ψ in the regex updating of text to ensure that this structure is preserved in the sanitized text. ConvFinQA performance is typically reported in terms of exact match accuracy of responses. Since sanitization introduces noise in the numerical values, exact matching is no longer an appropriate evaluation criterion; instead, we measure utility after sanitization with the relative error of the prediction. Moreover, we observed that for this dataset, the answers returned by a model are occasionally correct up to the target answers sign or magnitude—often due to the questions being underspecified rather than model error. To account for the sensitivity of relative error to incorrect magnitudes of predictions, we check if the relative error of the magnitude and sign adjusted response is less than .1 of the correct answer. If the adjusted error is sufficiently small, we record it instead.

Results. We report the 25th, 50th, and 75th percentiles of the relative error in GPT-4o's answers for sanitized and clean prompts. We further report the 25th, 50th, and 75th percentiles of "consistency", measured as the

TABLE II: Performance evaluation on ConvFinQA benchmark with varying degrees of prompt sanitization (ϵ represents the privacy parameter for mLDP). Higher relative error indicates larger deviation from ground truth, while lower prediction consistency indicates a low relative discrepancy between sanitized and unsanitized responses. "Base" here indicates the baseline.

Impact of Prεεmpt on ConvFinQA Performance						
ϵ	**Relative Error**			**Prediction Consistency**		
	25th	Median	75th	25th	Median	75th
0.1	0.0581	0.4000	4.4115	0.0698	0.3661	0.9994
0.5	0.0154	0.0776	1.0000	0.0167	0.1345	0.9898
1.0	0.0075	0.0408	0.9881	0.0084	0.0736	0.9899
2.0	0.0040	0.0244	0.8686	0.0044	0.0447	0.9899
Base	0.0000	0.0000	12.6749	-	-	-

relative difference between the model's prediction on the sanitized query compared to the prediction on the clean query. We report the results in Table II, observing a clear trend of performance improvement with a larger privacy budget, however, we note that at the 75th percentile, consistency does not change much. We also observe that the relative error of the sanitized prompts at the 75th percentile is lower than for unsanitized prompts, suggesting that addition of noise regularizes model behavior and prevents large outlier responses. We provide additional results with higher privacy budgets in App. 4.6 [25].

B. Comparison of Prεεmpt with Prior Methods

We compare Prεεmpt with Papillon [81], a contemporary privacy-preserving framework. It uses a local LLM to create a proxy of the user query that omits all PII values, while attaining high utility with respect to a remote, task-performing model. We consider two tasks in our setting: *Translation* and *Long-Context Q/A*. We use GPT-4o for all steps of Papillon, with Llama-3.1 8B Instruct [2] as the local model.

Translation. We consider 100 samples for each sensitive attribute ([*Name*],[*Age*],[*Money*]) for both languages, for a total of 600 samples. Following Papillon, we create optimized prompts for each attribute-language pair.

Results. We report BLEU scores in Table III. We find that Prεεmpt significantly outperforms Papillon, except for the [*Age*] and [*Money*] PII categories for the English-French translation task, where it is comparable. Furthermore, the average leakage of privacy due to NER failure is 71% of unique PII values compared to 97% for Prεεmpt. We detail their relative performance in App. 5.2 of [25].

TABLE III: BLEU scores for the English→German and English→French translation tasks, with UniNER-7B-PII for NER. All scores are w.r.t the reference translations from WMT-14. Higher value implies more similarity with the reference translation. We find that Prεεmpt has nearly identical performance with the translations of unmodified sentences and also outperforms prior methods.

English → German, NER: UniNER-7B-PII					
Attribute	**Gemini-1.5**		**GPT-4o**		
	Plain	Prεεmpt	Plain	Prεεmpt	Papillon
Name	0.334	0.341	0.287	0.278	0.175
Age	0.235	0.252	0.243	0.231	0.135
Money	0.245	0.274	0.217	0.200	0.153
English → French, NER: UniNER-7B-PII					
Name	0.423	0.408	0.432	0.419	0.290
Age	0.486	0.490	0.480	0.479	0.409
Money	0.329	0.333	0.294	0.279	0.299

Long-Context Q/A. We consider 50 samples for prompt optimization. Each sample contains a unique summary, a question based on it, and the corresponding answer.

Results. We report the STS scores in Table I. We find that Prεεmpt performs somewhat better than Papillon. However, Papillon as implemented, is mostly unsuccessful in preventing leakage for long context tasks. We observed that 80% of all prompts passed to the remote model contain character identities. These prompts are just the questions and do not include the summary. As the summaries are based off Wikipedia entries, the remote model is able to identify those characters and correctly respond to the query. We discuss NER performance and examples of successes and failures in App. 5.3 of [25].

C. Impact of Design Choices on Prεεmpt Utility

To address **Q3**, we examine design choices for two components: NER and encryption format.

Named-Entity Recognizer (NER). Uni-NER [92] is an LLM trained for generic named entity recognition. We finetune it on 10 high-risk categories from the AI4Privacy dataset [1]. We evaluate the NER as a type annotator on a held out subset of the dataset, consisting of text in English, German and French, with 50 samples per category, per language. We tabulate results for the following categories: "Money", "Name", "Age", "SSN", "Credit Card Number", "Zipcode", "Date", "Password", "Sex", "Phone Number". We make comparisons with off-the-shelf proprietary and open-source models, including: GPT-4.1 [69], Claude 4 Sonnet [6], Gemini 2.5 [50], Llama-3.1 8B Instruct [2] and Gemma-2 9B Instruct [37] (details in App. 6.1 of [25]). As our experiments only deal with *Name*, *Age* and *Money*, we use another Uni-NER model, specifically finetuned on these attributes.

TABLE IV: Named entity recognition (NER) F1 scores for English (E), German (G), and French (F). Our finetuned version of UniNER either matches or outperforms all other models on almost every sensitive attribute. "CCN" and "PN" stand for *Credit Card Number* and *Phone Number* respectively.

Attribute	**Part A: Open-source Models**									**Part B: Closed-source Models**								
	Uni-NER-7B-PII			**Gemma-2 9B Inst**			**Llama-3.1 8B Inst**			**GPT-4.1**			**Gemini-2.5**			**Claude 4 Sonnet**		
	E	G	F	E	G	F	E	G	F	E	G	F	E	G	F	E	G	F
Name	1.00	1.00	1.00	.907	.893	.846	.836	.766	.715	.843	.883	.845	.742	.903	.840	.791	.867	.872
Age	1.00	1.00	1.00	1.00	.951	.990	.960	.884	.822	.970	1.00	.990	.990	.990	.990	.980	1.00	.990
Money	.940	.860	.880	.940	.827	.824	.820	.710	.820	.882	.941	.959	.990	1.00	1.00	.990	.980	1.00
SSN	.990	1.00	.990	.640	.760	.653	.843	.871	.827	.875	.959	.960	.990	1.00	1.00	.969	.969	.929
CCN	.980	.960	1.00	.952	.962	.873	.916	.926	.855	.971	.971	.980	.980	.990	.970	.980	.980	.980
Zipcode	1.00	.990	.980	.980	.990	.980	.952	.925	.936	.980	.962	.981	.990	.980	.990	.990	.942	.990
Date	1.00	1.00	1.00	.778	.708	.649	.960	.846	.895	1.00	.980	.970	.860	.733	.784	1.00	.971	.980
Password	.980	1.00	1.00	.885	.887	.833	.238	.087	.163	1.00	.970	.950	.980	.810	.970	.990	.970	.922
Sex	1.00	1.00	1.00	.971	.943	.980	.926	.673	.830	.962	.945	.971	.971	.925	.990	.971	.954	.980
PN	.980	1.00	1.00	.952	.971	.962	.926	.971	.971	.980	.970	.990	.990	1.00	.990	.980	.980	.990

Results. As seen in Table IV (App. 6.1 of [25]), finetuning Uni-NER on these high-risk categories yields 100% F-1 scores across most attributes, often exceeding the performance of state-of-the-art proprietary models.

Encryption Format. Pr$\epsilon\epsilon$mpt assumes that the LLM's performance depends on preserving the format of type $\tau_{\mathtt{I}}$ tokens. We validate this assumption by evaluating the LLM on two other sanitization algorithms: (1) that does not preserve the format at all, and (2) that preserves an incorrect format. For the first case, we sanitize the type $\tau_{\mathtt{I}}$ tokens with AES, which replaces the sensitive tokens with 16 bytes of random strings. In the second case, we randomly substitute the tokens without maintaining the correct format (e.g. replacing a 5-digit ZIP code with a randomly chosen 8-digit value). We assess this in the context of a RAG task by generating 31 tuples of contexts (C), questions (Q), and answers (A) corresponding to a factual retrieval task. For each tuple, we evaluate the percentage of correct, desanitized answers using GPT-4.

Results. We observe that our model achieves 100% accuracy in factual information retrieval when employing FPE. However, performance drops to 70.97% with AES encryption and 77.42% with random substitution using incorrect formats. This confirms that format preservation is crucial for the LLM's performance. We provide additional results for the Translation and Multi-Turn Financial Q/A tasks with different privacy budgets in App. 4.2 and App. 4.6 of [25] respectively.

VI. Related Work

A line of work proposes to sanitize the prompts via substitution using a local LLM[78, 49, 22]. However, such solutions cannot be stateless if they intend to provide utility by desanitizing LLM responses. Cryptographic methods have also been explored for protecting user privacy at inference [43, 39, 21]. However, these approaches impose high computational and communication overheads. One line of approach for protecting privacy at inference involves employing DP for in-context learning by generating a synthetic dataset [29, 84, 41, 86]. However, these approaches are only applicable when a large collection of data is available, and are different from sanitizing an individuals sensitive information when they are submitting a simple query to an LLM. More similar to our setting are local DP based approaches. However, a key difference from our work is the way in which noise is added. A line of prior work employs metric DP by adding noise to text embeddings, and then decoding the private embeddings back into text [34]: this violates the definition of a prompt sanitizer as this might not preserve the types of the tokens (Sec. III). Another approach noisily sample a token from a pre-defined list of "similar" tokens [17, 8, 9, 20] which require carefully selecting the list of similar tokens. Another line of work generates a noisy paraphrase of the prompts [59, 87, 53, 44]. However, these methods suffer from the curse of dimensionality as the amount of noise grows proportionally with the length of the generated text leading to poor utility. Table V provides a summary comparing Pr$\epsilon\epsilon$mpt with prior work.

VII. Conclusion

LLMs introduce new challenges for protecting sensitive information at inference time. We address this by introducing a cryptographically inspired primitive—the prompt sanitizer—which transforms prompts to protect sensitive tokens. We then present Pr$\epsilon\epsilon$mpt, a system that implements this primitive with provable privacy guarantees. Experiments show that Pr$\epsilon\epsilon$mpt maintains high utility across both structured and open-ended prompts.

TABLE V: A comparison of prompt sanitization frameworks as per the design goals in Sec. IV-C. We find that Prεεmpt is the only framework that has all the desirable qualities of a secure and high utility prompt sanitizer.

Method	Stateless	Formal Privacy Guarantee	High Utility	Resource-Efficient
Prεεmpt (Ours)	✓	✓	✓	✓
Papillon [81]	✗	✗	✓	✓
Substitution-based [78, 49, 22]	✗	✗	✓	✓
Cryptography-based [43, 39, 21]	✓	✓	✓	✗
DP-based noising of text [29, 84, 41, 86]	✓	✓	✗	✗
DP-based noising of text embeddings [34]	✓	✓	✗	✓
DP-based noising of tokens [17, 8, 9, 20]	✓	✓	✗	✓
DP-based text paraphrasing [59, 87, 53, 44]	✓	✓	✗	✓

References

[1] ai4Privacy. *pii-masking-200k (Revision 1d4c0a1)*. 2023. DOI: 10.57967/hf/1532. URL: https://huggingface.co/datasets/ai4privacy/pii-masking-200k.

[2] Abhimanyu Dubey et al. *The Llama 3 Herd of Models*. 2024. arXiv: 2407.21783 [cs.AI]. URL: https://arxiv.org/abs/2407.21783.

[3] M. Alvim et al. "Invited Paper: Local Differential Privacy on Metric Spaces: Optimizing the Trade-Off with Utility". In: *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*. 2018, pp. 262–267.

[4] *Amazon Implements Restrictions on the Use of Generative AI Tools by Employees*. https://www.neatprompts.com/p/amazon-implements-restrictions-on-the-use-of-generative-ai-tools-by-employees. 2024.

[5] Rohan Anil et al. *Large-Scale Differentially Private BERT*. 2021. arXiv: 2108.01624 [cs.LG].

[6] Anthropic. *Claude 4 Sonnet*. 2025. URL: https://www.anthropic.com/claude/sonnet.

[7] *Apple restricts employees from using ChatGPT over fear of data leaks*. https://www.theverge.com/2023/5/19/23729619/apple-bans-chatgpt-openai-fears-data-leak. 2023.

[8] Stefan Arnold et al. "Driving Context into Text-to-Text Privatization". In: *arXiv preprint arXiv:2306.01457* (2023).

[9] Stefan Arnold et al. *Guiding Text-to-Text Privatization by Syntax*. 2023. arXiv: 2306.01471 [cs.CL].

[10] Clark Barrett et al. *Identifying and Mitigating the Security Risks of Generative AI*. 2023. arXiv: 2308.14840 [cs.AI].

[11] Mihir Bellare et al. *Format-Preserving Encryption*. Cryptology ePrint Archive, Paper 2009/251. https://eprint.iacr.org/2009/251. 2009. URL: https://eprint.iacr.org/2009/251.

[12] Ondřej Bojar et al. "Findings of the 2014 Workshop on Statistical Machine Translation". In: *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Ed. by Ondřej Bojar et al. Baltimore, Maryland, USA: Association for Computational Linguistics, June 2014, pp. 12–58. DOI: 10.3115/v1/W14-3302. URL: https://aclanthology.org/W14-3302.

[13] Hannah Brown et al. *What Does it Mean for a Language Model to Preserve Privacy?* 2022. arXiv: 2202.05520 [stat.ML].

[14] Tom Brown et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

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

[16] Nicholas Carlini et al. "Extracting training data from large language models". In: *30th USENIX Security Symposium (USENIX Security 21)*. 2021, pp. 2633–2650.

[17] Ricardo Silva Carvalho et al. "TEM: High Utility Metric Differential Privacy on Text". In: *ArXiv* abs/2107.07928 (2021). URL: https://api.semanticscholar.org/CorpusID:236034456.

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

[19] Konstantinos Chatzikokolakis et al. "Broadening the scope of differential privacy using metrics". In: *PETS*. 2013.

[20] Sai Chen et al. "A customized text sanitization mechanism with differential privacy". In: *Findings of the Association for Computational Linguistics: ACL 2023*. 2023, pp. 5747–5758.

[21] Tianyu Chen et al. "The-x: Privacy-preserving transformer inference with homomorphic encryption". In: *arXiv preprint arXiv:2206.00216* (2022).

[22] Yu Chen et al. "Hide and seek (has): A lightweight framework for prompt privacy protection". In: *arXiv preprint arXiv:2309.03057* (2023).

[23] Zhiyu Chen et al. *ConvFinQA: Exploring the Chain of Numerical Reasoning in Conversational Finance Question Answering*. 2022. arXiv: 2210.03849 [cs.CL]. URL: https://arxiv.org/abs/2210.03849.

[24] Zhiyu Chen et al. *FinQA: A Dataset of Numerical Reasoning over Financial Data*. 2022. arXiv: 2109.00122 [cs.CL]. URL: https://arxiv.org/abs/2109.00122.

[25] Amrita Roy Chowdhury et al. *Prεεmpt: Sanitizing Sensitive Prompts for LLMs*. 2025. arXiv: 2504.05147 [cs.CR]. URL: https://arxiv.org/abs/2504.05147.

[26] Aloni Cohen. "Attacks on Deidentification's Defenses". In: *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 1469–1486. ISBN: 978-1-939133-31-1. URL: https://www.usenix.org/conference/usenixsecurity22/presentation/cohen.

[27] Jiaxi Cui et al. "Chatlaw: Open-source legal large language model with integrated external knowledge bases". In: *arXiv preprint arXiv:2306.16092* (2023).

[28] *Doctors banned from using ChatGPT to write medical notes*. https://www.ausdoc.com.au/news/doctors-banned-from-using-chatgpt-to-write-medical-notes/. 2023.

[29] Haonan Duan et al. "Flocks of stochastic parrots: differentially private prompt learning for large language models". In: *Proceedings of the 37th International Conference on Neural Information Processing Systems*. NIPS '23. New Orleans, LA, USA: Curran Associates Inc., 2024.

[30] Cynthia Dwork et al. "Exposed! A Survey of Attacks on Private Data". In: *Annual Review of Statistics and Its Application* 4 (2017), pp. 61–84. URL: https://api.semanticscholar.org/CorpusID:26766335.

[31] Cynthia Dwork et al. "The Algorithmic Foundations of Differential Privacy". In: *Found. Trends Theor. Comput. Sci.* 9.3&#8211;4 (Aug. 2014), pp. 211–407. ISSN: 1551-305X.

[32] Cynthia Dwork et al. "The Algorithmic Foundations of Differential Privacy". In: *Found. Trends Theor. Comput. Sci.* 9.3–4 (Aug. 2014), pp. 211–407. ISSN: 1551-305X. DOI: 10.1561/0400000042. URL: https://doi.org/10.1561/0400000042.

[33] Stefan Dziembowski et al. "Leakage-resilient cryptography". In: *2008 49th Annual IEEE Symposium on Foundations of Computer Science*. IEEE. 2008, pp. 293–302.

[34] Oluwaseyi Feyisetan et al. "Privacy-and utility-preserving textual analysis via calibrated multivariate perturbations". In: *Proceedings of the 13th international conference on web search and data mining*. 2020, pp. 178–186.

[35] Yunfan Gao et al. "Retrieval-augmented generation for large language models: A survey". In: *arXiv preprint arXiv:2312.10997* (2023).

[36] Petko Georgiev et al. Gemini Team. *Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context*. 2024. arXiv: 2403.05530 [cs.CL]. URL: https://arxiv.org/abs/2403.05530.

[37] Morgane Riviere et al. Gemma Team. *Gemma 2: Improving Open Language Models at a Practical Size*. 2024. arXiv: 2408.00118 [cs.CL]. URL: https://arxiv.org/abs/2408.00118.

[38] *General Data Protection Regulation GDPR*. https://gdpr-info.eu/. 2016.

[39] Meng Hao et al. "Iron: Private inference on transformers". In: *Advances in neural information processing systems* 35 (2022), pp. 15718–15731.

[40] Peter Henderson et al. "Ethical Challenges in Data-Driven Dialogue Systems". In: *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*. AIES '18. New Orleans, LA, USA: Association for Computing Machinery, 2018, pp. 123–129. ISBN: 9781450360128. DOI: 10.1145/3278721.3278777. URL: https://doi.org/10.1145/3278721.3278777.

[41] Junyuan Hong et al. "Dp-opt: Make large language model your privacy-preserving prompt engineer". In: *arXiv preprint arXiv:2312.03724* (2023).

[42] Shlomo Hoory et al. "Learning and Evaluating a Differentially Private Pre-trained Language Model". In: *PRIVATENLP*. 2021. URL: https://api.semanticscholar.org/CorpusID:235097653.

[43] Xiaoyang Hou et al. *CipherGPT: Secure Two-Party GPT Inference*. Cryptology ePrint Archive, Paper 2023/1147. 2023. URL: https://eprint.iacr.org/2023/1147.

[44] Timour Igamberdiev et al. *DP-BART for Privatized Text Rewriting under Local Differential Privacy*. 2023. arXiv: 2302.07636 [cs.CR].

[45] Jacob Imola et al. *Metric Differential Privacy at the User-Level*. 2024. arXiv: 2405.02665 [cs.CR]. URL: https://arxiv.org/abs/2405.02665.

[46] Katharina Jeblick et al. "ChatGPT makes medicine easy to swallow: an exploratory case study on simplified radiology reports". In: *European Radiology* (2023), pp. 1–9.

[47] *JPMorgan Chase Restricts Staffers' Use Of ChatGPT*. https://www.forbes.com/sites/siladityaray/2023/02/22/jpmorgan-chase-restricts-staffers-use-of-chatgpt/. 2023.

[48] Firuz Kamalov et al. *New Era of Artificial Intelligence in Education: Towards a Sustainable Multifaceted Revolution*. 2023. arXiv: 2305.18303 [cs.CY].

[49] Zhigang Kan et al. "Protecting user privacy in remote conversational systems: A privacy-preserving framework based on text sanitization". In: *arXiv preprint arXiv:2306.08223* (2023).

[50] Koray Kavukcuoglu. *Gemini 2.5: Our most intelligent AI model*. 2025. URL: https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/#gemini-2-5-thinking.

[51] Tomáš Kočiský et al. "The NarrativeQA Reading Comprehension Challenge". In: *Transactions of the Association for Computational Linguistics* 6 (2018). Ed. by Lillian Lee et al., pp. 317–328. DOI: 10.1162/tacl_a_00023. URL: https://aclanthology.org/Q18-1023/.

[52] Katherine Lee et al. "Deduplicating training data makes language models better". In: *arXiv preprint arXiv:2107.06499* (2021).

[53] Mike Lewis et al. "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*. 2020, pp. 7871–7880.

[54] Xuechen Li et al. *Large Language Models Can Be Strong Differentially Private Learners*. 2022. arXiv: 2110.05679 [cs.LG].

[55] Guo Lin et al. *EmojiCrypt: Prompt Encryption for Secure Communication with Large Language Models*. 2024. arXiv: 2402.05868 [cs.CL]. URL: https://arxiv.org/abs/2402.05868.

[56] Chenyang Lyu et al. *A Paradigm Shift: The Future of Machine Translation Lies with Large Language Models*. 2024. arXiv: 2305.01181 [cs.CL]. URL: https://arxiv.org/abs/2305.01181.

[57] M. Maffei et al. "On the Security of Frequency-Hiding Order-Preserving Encryption". In: *Cryptology and Network Security*. Ed. by Srdjan Capkun et al. Cham: Springer International Publishing, 2018, pp. 51–70. ISBN: 978-3-030-02641-7.

[58] Peihua Mai et al. "Split-and-Denoise: Protect large language model inference with local differential privacy". In: *arXiv preprint arXiv:2310.09130* (2023).

[59] Justus Mattern et al. "The Limits of Word Level Differential Privacy". In: *Findings of the Association for Computational Linguistics: NAACL 2022*. 2022, pp. 867–881.

[60] H. Brendan McMahan et al. *Learning Differentially Private Recurrent Language Models*. 2018. arXiv: 1710.06963 [cs.LG].

[61] Niloofar Mireshghallah et al. *Can LLMs Keep a Secret? Testing Privacy Implications of Language Models via Contextual Integrity Theory*. 2023. arXiv: 2310.17884 [cs.AI].

[62] Philip Moons et al. "ChatGPT: can artificial intelligence language models be of value for cardiovascular nurses and allied health professionals". In: *European Journal of Cardiovascular Nursing* 22.7 (Feb. 2023), e55–e59. ISSN: 1474-5151. DOI: 10.1093/eurjcn/zvad022. eprint: https://academic.oup.com/eurjcn/article-pdf/22/7/e55/52017892/zvad022.pdf. URL: https://doi.org/10.1093/eurjcn/zvad022.

[63] *More federal agencies join in temporarily blocking or banning ChatGPT*. https://fedscoop.com/more-federal-agencies-join-in-temporarily-blocking-or-banning-chatgpt/. 2024.

[64] Arvind Narayanan et al. "Myths and fallacies of "Personally Identifiable Information"". In: *Commun. ACM* 53.6 (June 2010), pp. 24–26. ISSN: 0001-0782. DOI: 10.1145/1743546.1743558. URL: https://doi.org/10.1145/1743546.1743558.

[65] Jorge Nocedal et al. *Numerical Optimization*. 2e. New York, NY, USA: Springer, 2006.

[66] Goldreich Oded. *Foundations of Cryptography: Volume 2, Basic Applications*. 1st. New York, NY, USA: Cambridge University Press, 2009. ISBN: 052111991X, 9780521119917.

[67] Goldreich Oded. *Foundations of Cryptography: Volume 2, Basic Applications*. 1st. USA:

Cambridge University Press, 2009. ISBN: 052111991X.

[68] OpenAI. *GPT-4 Technical Report*. 2023. arXiv: 2303.08774 [cs.CL].

[69] OpenAI. *Introducing GPT-4.1 in the API*. 2025. URL: https://openai.com/index/gpt-4-1/.

[70] Q. Pang et al. “BOLT: Privacy-Preserving, Accurate and Efficient Inference for Transformers”. In: *2024 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2024, pp. 133–133. DOI: 10.1109/SP54263.2024.00130. URL: https://doi.ieeecomputersociety.org/10.1109/SP54263.2024.00130.

[71] Kishore Papineni et al. “BLEU: a method for automatic evaluation of machine translation”. In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. ACL ’02. Philadelphia, Pennsylvania: Association for Computational Linguistics, 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: https://doi.org/10.3115/1073083.1073135.

[72] Kishore Papineni et al. “Bleu: a Method for Automatic Evaluation of Machine Translation”. In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Ed. by Pierre Isabelle et al. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: https://aclanthology.org/P02-1040.

[73] Swaroop Ramaswamy et al. *Training Production Language Models without Memorizing User Data*. 2020. arXiv: 2009.10031 [cs.LG].

[74] Nils Reimers et al. “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks”. In: *Conference on Empirical Methods in Natural Language Processing*. 2019. URL: https://api.semanticscholar.org/CorpusID:201646309.

[75] Amrita Roy Chowdhury et al. “Strengthening Order Preserving Encryption with Differential Privacy”. In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’22. Los Angeles, CA, USA: Association for Computing Machinery, 2022, pp. 2519–2533. ISBN: 9781450394505. DOI: 10.1145/3548606.3560610. URL: https://doi.org/10.1145/3548606.3560610.

[76] *Samsung Bans ChatGPT Among Employees After Sensitive Code Leak*. https://www.forbes.com/sites/siladityaray/2023/05/02/samsung-bans-chatgpt-and-other-chatbots-for-employees-after-sensitive-code-leak/. 2023.

[77] *Samsung Bans Staff’s 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. 2023.

[78] Zhili Shen et al. *The Fire Thief Is Also the Keeper: Balancing Usability and Privacy in Prompts*. 2024. arXiv: 2406.14318 [cs.CR]. URL: https://arxiv.org/abs/2406.14318.

[79] Weiyan Shi et al. *Selective Differential Privacy for Language Modeling*. 2022. arXiv: 2108.12944 [cs.CL].

[80] Karan Singhal et al. “Towards expert-level medical question answering with large language models”. In: *arXiv preprint arXiv:2305.09617* (2023).

[81] Li Siyan et al. “PAPILLON: PrivAcy Preservation from Internet-based and Local Language MOdel ENsembles”. In: *arXiv preprint arXiv:2410.17127* (2024).

[82] *Social Security Administration issues temporary block on generative AI*. https://fedscoop.com/social-security-administration-temporary-block-generative-ai/. 2023.

[83] *Space Force Pumps the Brakes on ChatGPT-Like Technology With Temporary Ban*. https://www.airandspaceforces.com/space-force-chatgpt-technology-temporary-ban/. 2023.

[84] Xinyu Tang et al. “Privacy-preserving in-context learning with differentially private few-shot generation”. In: *arXiv preprint arXiv:2309.11765* (2023).

[85] Om Dipakbhai Thakkar et al. “Understanding Unintended Memorization in Language Models Under Federated Learning”. In: *Proceedings of the Third Workshop on Privacy in Natural Language Processing*. Ed. by Oluwaseyi Feyisetan et al. Online: Association for Computational Linguistics, June 2021, pp. 1–10. DOI: 10.18653/v1/2021.privatenlp-1.1. URL: https://aclanthology.org/2021.privatenlp-1.1.

[86] Zhiliang Tian et al. “Seqpate: Differentially private text generation via knowledge distillation”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 11117–11130.

[87] Saiteja Utpala et al. “Locally differentially private document generation using zero shot prompting”. In: *arXiv preprint arXiv:2310.16111* (2023).

[88] *Workers' ChatGPT Use Restricted At More Banks—Including Goldman, Citigroup*. https://www.forbes.com/sites/brianbushard/2023/02/24/workers-chatgpt-use-restricted-at-more-banks-including-goldman-citigroup/. 2023.

[89] Da Yu et al. *Differentially Private Fine-tuning of Language Models*. 2022. arXiv: 2110.06500 [cs.LG].

[90] Hanlin Zhang et al. "Watermarks in the Sand: Impossibility of Strong Watermarking for Generative Models". In: *Forty-first International Conference on Machine Learning*. 2024.

[91] Mengke Zhang et al. *LatticeGen: A Cooperative Framework which Hides Generated Text in a Lattice for Privacy-Aware Generation on Cloud*. 2024. arXiv: 2309.17157 [cs.CL]. URL: https://arxiv.org/abs/2309.17157.

[92] Wenxuan Zhou et al. "UniversalNER: Targeted Distillation from Large Language Models for Open Named Entity Recognition". In: (2023). arXiv: 2308.03279 [cs.CL].

Appendix

A. Error due to NER

Pr$\epsilon\epsilon$mpt protects sensitive tokens under error due to NER as follows:

Theorem 4. *Let S be the set of all token pairs of type τ_{II} that are different in the prompt pairs $(\boldsymbol{\rho}_0, \boldsymbol{\rho}_1)$ in the privacy game $\mathbf{G}^{\mathrm{pp}}_{\mathsf{PS},\mathcal{L}_{NER}}$. Then, for Pr$\epsilon\epsilon$mpt we have:*

$$\mathbf{Adv}^{\mathrm{pp}}_{Pr\epsilon\epsilon mpt,\mathcal{L}_{NER}}(\mathcal{A}) \leq e^{l\epsilon} + negl(\kappa) \tag{5}$$

where $l = \max_{(\sigma_0,\sigma_1)\in S}\{|\sigma_0 - \sigma_1|\}$ and κ is the security parameter of the underlying FPE scheme.

Proof. Note that the proof of Theorem 2 is modular. The additional leakage introduced by NER errors is already explicitly captured in the prompts constructed with the modified leakage function. Consequently, the remainder of the proof proceeds in exactly the same way. Our security argument remains unchanged, as it relies on a hybrid argument over prompts that differ in only a single token. So our starting point is two prompts which differ in a single token that falls in the $(1-\lambda)\%$ tokens that were not replaced. $\square$