Leveraging Soft Prompts for Privacy Attacks in Federated Prompt Tuning

Quan Minh Nguyen
University of Florida

Min-Seon Kim
North Carolina State University

Hoang M. Ngo
University of Florida

Trong Nghia Hoang
Washington State University

Hyuk-Yoon Kwon
Seoul National University of Science and Technology

My T. Thai*
University of Florida
mythai@cise.ufl.edu

Abstract

Membership inference attacks (MIAs) pose a serious privacy threat in federated learning (FL). While MIAs have been extensively studied in standard FL, the recent shift toward federated fine-tuning introduces new and largely unexplored attack surfaces. In this work, we show that federated prompt-tuning, which adapts pre-trained foundation models using lightweight input prefixes, exposes a novel and effective vector for membership inference. We propose PROMPTMIA, a membership inference attack tailored to federated prompt-tuning, in which a malicious server introduces adversarially crafted prompts and exploits their updates during collaborative training to determine whether a target data point belongs to a client's private dataset. We formalize this threat via a security game and demonstrate that PROMPTMIA achieves consistently high attack advantage across diverse benchmark datasets, substantially outperforming current SOTA federated MIAs. We also provide a theoretical lower bound on the attack advantage that explains the observed empirical behavior. Finally, we show that existing MIA defenses are often ineffective against PROMPTMIA, highlighting the need for defense mechanisms specifically tailored to prompt-tuning in federated settings.

1 Introduction

Federated Learning (FL) [1] is a prominent training paradigm that enables data holders to collaboratively train a shared model without exposing their private data. Despite this, FL methods are vulnerable to adversarial threats [2, 3]. One prominent threat is membership inference attack (MIA) [4], where the adversarial server attempts to determine whether a specific data record was included in a client's training dataset. Existing research has predominantly studied such privacy threats in settings where the attack surface is tied to clients' gradients or full model parameters exchanged during training [5, 6]. This classical view has, however, overlooked new attack surfaces that arise in more recent FL paradigms that instead communicate and aggregate external adaptation modules of pre-trained foundation models, e.g. soft-prompts prepended to input embeddings. [7].

To raise awareness of the potential existence of unexplored privacy threats in recent FL paradigms, this paper reveals a novel attack surface via manipulating the clustering and aggregation of local sets of soft prompts in federated prompt-tuning (FPT) [7]. To the best of our knowledge, this is the first work that identifies an unexplored privacy risk in soft prompt optimization for representation fine-tuning, particularly in FPT. This work is orthogonal to and complement existing research on privacy risks in prompting large language models with exemplars and/or task instructions that contain private data for

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

in-context learning (ICL). Such studies focus largely on designing adversarial queries that expose whether a particular data point was included in private instruction prompts used for ICL [8, 9]. In contrast, soft prompt optimization does not include private data in the prompts themselves. Moreover, the attack model in in-context learning originates from end users external to the training system whereas in FPT, it arises from the server within the training network. Existing defenses against prompt risk in in-context learning [10, 11, 12] are therefore not applicable to soft prompt risk in FPT.

To highlight the privacy risk that arises within soft prompt aggregation in FPT, we develop PROMPTMIA, a novel membership inference attack (MIA) that exploits FPT's prompt update and selection mechanism as a new attack surface. This is substantiated with the following contributions:

(I) We develop an algorithm that optimizes adversarial prompts to be preferentially selected and updated by local clients when their datasets contain a target data point, while remaining unselected otherwise. PROMPTMIA is substantiated via running this algorithm on the server to create and add adversarial prompts to the shared prompt pools. The server can keep track of changes made to these prompts between communication iterations to infer membership information in a single communication round, without the need for training a shadow model or external data (Section 3.2.2).

(II) We analyze the theoretical attack advantage of PROMPTMIA through the lens of a security game. In this view, the advantage of the attacker is characterized in terms of the true and false positive rates which measure the chance that the adversarial prompts are correctly and incorrectly selected, respectively. Our analysis shows that PROMPTMIA achieves perfect true positive rate and provably low false positive rate, resulting in a high attack advantage (Section 3.3).

(III) We run extensive experiments to validate the effectiveness of PROMPTMIA across 7 datasets including CIFAR-10, CIFAR-100 [13], TinyImageNet [14], MNIST-M[15], Fashion-MNIST[16], CINIC-10 [17], MMAFEDB[18] and 3 vision transformer architectures including ViT-B/32 [19], DeiT-B/16 [20], and ConViT [21]. Our empirical results consistently show that PROMPTMIA outperforms SOTA federated MIAs [22, 23, 24], achieving $> 90\%$ attack success rate on all datasets, which corroborates the aforementioned high attack advantage (Section 4.1). Our code is available at `https://github.com/GivralNguyen/Prompt-MIA`.

(IV) While the primary objective of this work is not to propose novel defense mechanisms or conduct an exhaustive evaluation of all possible defense strategies against PROMPTMIA, we conduct additional analyses and experiments to examine how PROMPTMIA interacts with FL clients implementing standard defenses such as outlier detection [25, 26, 27, 28], input noise perturbation [29], regularization-based defenses [30, 31, 32] (Section 4.2), as well as defenses that modify the prompt selection or aggregation protocol, such as Secure Aggregation [33], randomized key indices, and randomized prompt selection via dropout, and monitoring protocols such as cross-round key-drift auditing (Appendix A.6). We show that these defenses are often either provably or empirically ineffective. We do not claim that no future audit or secure protocol can defeat the attack. Rather, these results underscore the urgent need for more research on privacy defenses against this new prompt-based attack surface.

2 Federated Prompt-Tuning

To address the resource bottleneck in traditional full model fine-tuning, recent literature in federated learning (FL) has increasingly adopted parameter-efficient tuning methods, notably prompt tuning. Prompt-tuning [34, 35] reformulates the downstream task adaptation problem as input modification rather than weight updates. This method focuses on optimizing learnable soft-prompts which are appended to the input embedding in a transformer architecture. These prompts provide beneficial context for downstream tasks, similar to how hints can be provided to assist puzzle solving. In Federated Prompt-Tuning (FPT), a central server maintains a global prompt pool that clients collaboratively optimize, while keeping the underlying large-scale foundation model frozen. Let the global prompt pool of size M be denoted as $\mathcal{P}_{\text{GLOBAL}} = \{P_i\}_{i=1}^{M}$ with a corresponding set of learnable keys $\mathcal{K}_{\text{GLOBAL}} = \{k_i\}_{i=1}^{M}$. During training or inference, given an input x, the frozen pre-trained model f extracts the `[CLS]` representation to form a deterministic query $q(x) = f(x)[0, :]$. Using cosine distance γ, the client retrieves the top-N contextually relevant prompts from $\mathcal{P}_{\text{GLOBAL}}$ (Eq. 1):

$$\hat{\mathcal{K}}_x = \underset{\{s_i\}_{i=1}^{N} \subseteq [M]}{\operatorname{argmin}} \sum_{i=1}^{N} \gamma\big(q(x), k_{s_i}\big) = \underset{\{s_i\}_{i=1}^{N} \subseteq [M]}{\operatorname{argmax}} \sum_{i=1}^{N} \kappa\big(q(x), k_{s_i}\big), \qquad [M] = \{1, \ldots, M\}. \quad (1)$$

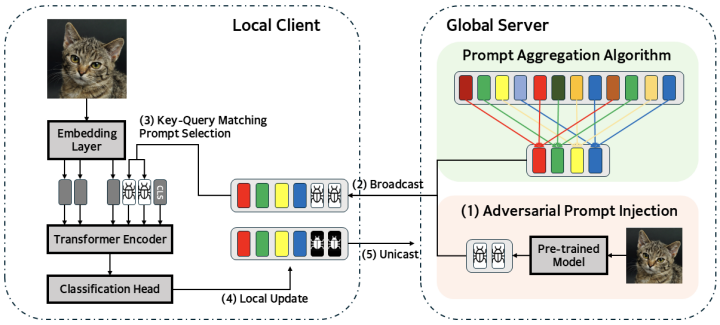


Figure 1: PROMPTMIA workflow: (1) the server injects adversarial prompts designed for a target sample into the global prompt pool; (2) the modified pool is broadcast to clients; (3) each client performs query-key matching, which selects all adversarial prompts if the target sample is present; (4) selected prompts are locally updated and (5) returned to the server. By monitoring which prompts are updated, the server infers the target's membership in client data.

i.e., the N prompts whose associated keys are closest to the query feature $q(x)$ under the cosine distance γ, or equivaliently, the N prompts whose associated keys have the highest cosine similarity to $q(x)$. These selected prompts are prepended to the input embedding x_e, forming the adapted input $x_p = [P_{s_1}; \cdots; P_{s_N}; x_e]$. Let $h_{\text{CLS}}(x_p)$ denote the final `[CLS]` representation of the prompt-augmented input. Let g denote the optional frozen pretrained classification head. With the trainable classifier f_c^ϕ parameterized by ϕ, the training objective is defined as:

$$\min_{\mathcal{P},\mathcal{K},\phi} \mathcal{L}\left(f_c^\phi(g(h_{\text{CLS}}(x_p))), y\right) + \lambda \sum_{i \in \hat{\mathcal{K}}_x} \gamma\left(q(x), k_i\right). \tag{2}$$

The first term is the softmax cross-entropy, while the second is a surrogate that pulls the selected keys to align with the corresponding query features. Given input image x and global key-prompt pool $(\mathcal{K}_{\text{GLOBAL}}, \mathcal{P}_{\text{GLOBAL}})$, the client selects $\hat{\mathcal{K}}_x$ using Eq. 1 and updates $\hat{\mathcal{K}}_x$, the corresponding prompts $\hat{\mathcal{P}}_x$ along with ${f_c}^\phi$ according to Eq. 2. The server aggregates the trained prompts from participating clients to update $(\mathcal{K}_{\text{GLOBAL}}, \mathcal{P}_{\text{GLOBAL}})$ using any of the existing FL algorithms (e.g., FEDAVG [1], FEDPROX [36] or PFPT [7]). See Appendix A.1 for more details.

3 MIAs against Federated Prompt-Tuning

In this section, we present the formulation and workflow of PROMPTMIA. An overview of the attack is shown in Fig. 1. Unlike prior approaches that rely on model weights, gradients [22, 23], or auxiliary datasets [24] to train shadow models, PROMPTMIA operates by injecting adversarial prompts $\mathcal{P}_{\text{ADV}}$ into the shared prompt pool $\mathcal{P}_{\text{GLOBAL}}$ prior to a training round. These prompts are designed to be selected only when target sample $\mathcal{T}$ is present in a client's data. As clients select and update prompts (Eq. 1), the server can monitor which injected prompts are modified and use this as membership signal. PROMPTMIA exploits prompt updates as a privacy channel to infer membership in a single round, reducing both computational cost and adversarial assumptions. PROMPTMIA achieves perfect true positive rate and provably low false positive rate, resulting in a high attack success rate.

3.1 Prompt based AMIs as Security Games

We formalize the prompt-based active membership inference attack (AMI) threat models through security games between a challenger (client) and an adversary (server) in the FPT setting.

(I) The adversary is denoted by $\mathcal{A}$, and the corresponding security games are represented as $\mathsf{Exp}^{\text{AMI}}(\mathcal{A})$. In $\mathsf{Exp}^{\text{AMI}}(\mathcal{A})$, the adversarial server $\mathcal{A}$ consists of three components: $\mathcal{A}_{\mathsf{INIT}}$, $\mathcal{A}_{\mathsf{ATTACK}}$, and $\mathcal{A}_{\mathsf{GUESS}}$. At the beginning of the games, a random bit b determines whether a target sample $\mathcal{T}$ belongs to the client's data $\mathcal{D}$ (with $b = 1$ indicating membership).

(II) In the initialization phase $\mathcal{A}_{\mathsf{INIT}}$, the server constructs the current round's aggregated global key-prompt pool $(\mathcal{K}_{\text{GLOBAL}}, \mathcal{P}_{\text{GLOBAL}}) = \{ (k_i, P_i) \}_{i=1}^{M}$ using any existing FL aggregation algorithm.

(III) In the attack phase $\mathcal{A}_{\mathsf{ATTACK}}$, the server constructs a set of N adversarial keys $\mathcal{K}_{\text{ADV}} = \{ k_{a_m} \}_{m=1}^{N}$, where each k_{a_m} is an adversarial key vector explicitly generated for target sample $\mathcal{T}$ given query function q. It then selects a subset of N prompts $\{(k_j, P_j)\}_{j \in S} \subseteq (\mathcal{K}_{\text{GLOBAL}}, \mathcal{P}_{\text{GLOBAL}})$ where $S \subseteq \{1, \ldots, M\}$ and $|S| = N$. For each index $j \in S$, the server replaces the original key k_j with an adversarial key from $\mathcal{K}_{\text{ADV}}$. While only the key values are changed, for ease of notation, we define the set of adversarial prompts as $(\mathcal{K}_{\text{ADV}}, \mathcal{P}_{\text{ADV}}) = \{ (k_{a_m}, P_{a_m}) \}_{m=1}^{N}$. Correspondingly, the set of remaining prompts are benign prompts $(\mathcal{K}_{\text{BENIGN}}, \mathcal{P}_{\text{BENIGN}}) = \{ (k_{b_n}, P_{b_n}) \}_{n=1}^{M-N}$. Together, we obtain the modified global prompt pool $(\tilde{\mathcal{K}}, \tilde{\mathcal{P}}) = \big(\mathcal{K}_{\text{ADV}} \cup \mathcal{K}_{\text{BENIGN}},\ \mathcal{P}_{\text{ADV}} \cup \mathcal{P}_{\text{BENIGN}}\big)$.

$\mathcal{A}_{\mathsf{ATTACK}}$ then distributes $(\tilde{\mathcal{K}}, \tilde{\mathcal{P}})$ to participating clients (instead of $(\mathcal{K}, \mathcal{P})$ like in regular FPT). Each client t selects $(\hat{\mathcal{K}}_t, \hat{\mathcal{P}}_t)$ from $(\tilde{\mathcal{K}}, \tilde{\mathcal{P}})$ using Eq. 1 and updates them based on their local data $\mathcal{D}$.

(IV) In $\mathcal{A}_{\mathsf{GUESS}}$, the server uses $\hat{\mathcal{P}}_t$ to guess b, effectively identifying whether $\mathcal{T} \in \mathcal{D}$. The advantage of the adversarial server $\mathcal{A}^{\mathcal{D}}$ in the security game is given by:

$$\mathbf{Adv}^{\text{AMI}}(\mathcal{A}) = \Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0] \tag{3}$$

where b is the true membership label, and b' is the adversary's prediction. We define the *attack success rate* as:

$$\mathbf{ASR}^{\text{AMI}}(\mathcal{A}) = \frac{1}{2}(\Pr[b' = 1 \mid b = 1] + \Pr[b' = 0 \mid b = 0]) = \tfrac{1}{2}\big(1 + \mathbf{Adv}^{\text{AMI}}(\mathcal{A})\big). \tag{4}$$

3.2 MIA using Adversarial Prompt Injection

Given the security game laid out in Section 3.1, the adversary's objective is to inject a set of N adversarial keys-prompts $(\mathcal{K}_{\text{ADV}}, \mathcal{P}_{\text{ADV}})$ into the global prompt pool such that, if $\mathcal{T} \in \mathcal{D}$, the top-$N$ selected prompts will always be the adversarial set. The membership signal is defined as the event that **all adversarial prompts are selected and subsequently updated**. To build intuition, we first introduce a *Naive Prompt Injection* attack and analyze its weaknesses, before introducing the more robust method, PROMPTMIA, which achieves perfect true positive rate and provably low false positive rate, resulting in a high attack advantage (Section 3.3).

3.2.1 Naive Prompt Injection Attack

In this approach, the adversary constructs adversarial keys $\mathcal{K}_{\text{ADV}} = \{ k_{a_m} \}_{m=1}^{N}$ and inserts them into the global prompt pool. The goal is to ensure that if target data $\mathcal{T} \in \mathcal{D}$, the top-$N$ selected prompts using Eq. 1 always coincide with $\mathcal{K}_{\text{ADV}}$. A straightforward strategy is to align each adversarial key k_{a_m} directly with the client's query vector $q(\mathcal{T})$ so as to maximize cosine similarity. We ensure that:

$$\kappa(q(\mathcal{T}), k_{a_m}) = 1 \ \text{ or } \ k_{a_m} = q(\mathcal{T}), \quad \forall\, k_{a_m} \in \mathcal{K}_{\text{ADV}}. \tag{5}$$

where κ is the cosine similarity operator. The server selects a subset $S \subseteq [M]$ with $|S| = N$, and for each $j \in S$ performs key modificaton:

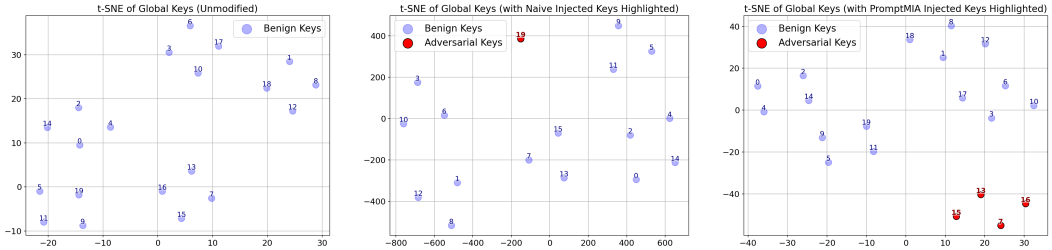
$$(k_j, P_j) \ \mapsto \ (k_{a_m}, P_j), \qquad m = 1, \ldots, N, \tag{6}$$

However, this naive attack suffers from fundamental weaknesses. Since all adversarial keys collapse to the same vector $q(\mathcal{T})$ due to Eq. 5, clients can easily detect this redundancy through dimensionality reduction techniques (e.g., t-SNE, PCA) (see Fig. 2b), which would reveal a tight cluster of identical keys, or simply by directly inspecting key values. In addition, simple defenses such as discarding any key whose similarity exceeds a suspicious threshold (e.g., excluding $\kappa(q(\mathcal{T}), k_{a_m}) > 1 - r$ for small $r > 0$) can render the attack ineffective.

Finally, because all adversarial keys are identical, the attack suffers from a high false positive rate (see Section 4.1) : even when the target sample is absent, adversarial prompts may still be selected whenever the similarity of one adversarial key (and as a result, all adversarial keys) to some non-target query $q(x)$, $x \in \mathcal{D}$, $x \neq \mathcal{T}$, is higher than all benign keys, or $\kappa(q(x), k_{a_m}) > \max_{k_b \in \mathcal{K}_{\text{BENIGN}}} \kappa(q(x), k_b)$, significantly increasing FPR and reducing the attack's advantage.

3.2.2 PromptMIA

To build up on and overcome the limitations of the naive prompt injection attack, we impose two requirements on adversarial key generation. First, adversarial keys $\mathcal{K}_{\text{ADV}}$ must achieve higher cosine



(a) t-SNE of global keys without adversarial key injection.

(b) t-SNE visualization shows naively injected adversarial keys overlapping

(c) t-SNE of global keys after PromptMIA adversarial keys injection.

Figure 2: Comparison of the global key distributions produced by a ViT-B/32 model trained on CIFAR-10 after 60 global epochs, visualized using t-SNE. Blue keys are benign keys, and red keys are adversarial keys.

similarity with the target query $q(\mathcal{T})$ than any benign key $\mathcal{K}_{\text{BENIGN}}$. Second, adversarial keys must be sufficiently diverse from one another to not be trivially exposed. Formally, the server selects a subset $S \subseteq [M]$ of size N, and for each index $j \in S$, the original key k_j is replaced with an adversarial key $k_{a_m} \in \mathcal{K}_{\text{ADV}}$ (see Eq. 6). k_{a_m} is constrained to lie within a cosine similarity interval with $q(\mathcal{T})$:

$$\kappa(q(\mathcal{T}), k_{a_m}) \in \left[\max_{k_b \in \mathcal{K}_{\text{BENIGN}}} \kappa(q(\mathcal{T}), k_b) + \delta_{\min}, \quad \max_{k_b \in \mathcal{K}_{\text{BENIGN}}} \kappa(q(\mathcal{T}), k_b) + \delta_{\min} + \Delta \right], \forall\, k_{a_m} \in \mathcal{K}_{\text{ADV}} \tag{7}$$

where $\delta_{\min}$ ensures that all adversarial keys have higher cosine similarity to $q(\mathcal{T})$ than benign keys, while Δ introduces controlled variability to prevent all adversarial keys from collapsing to the same vector. Algorithm 1 describes the procedure for generating a single adversarial key with a specified cosine similarity s to the target query. Building on this primitive, Algorithm 2 constructs a set of N adversarial keys $\mathcal{K}_{\text{ADV}}$ that satisfies Eq. 7. Details of these algorithms are given in Appendix A.2. PROMPTMIA allows the adversary to generate an adversarial key set $\mathcal{K}_{\text{ADV}}$ that will always be selected and updated when $\mathcal{T} \in \mathcal{D}$, while not being easily detectable (see Fig. 2c and Section 4.2). The server generates and distributes the modified global prompt pool $(\tilde{\mathcal{K}}, \tilde{\mathcal{P}}) = \left(\mathcal{K}_{\text{ADV}} \cup \mathcal{K}_{\text{BENIGN}},\ \mathcal{P}_{\text{ADV}} \cup \mathcal{P}_{\text{BENIGN}}\right)$ to all participating clients. Given client t with local dataset $\mathcal{D}$ and input data $x \in \mathcal{D}$, the client computes $q(x)$ and selects top-N keys $\hat{\mathcal{K}}_x$ using Eq. 1 and update only the selected prompts using Eq. 2. If $x \equiv \mathcal{T}$, then we have the top-N keys are exactly the adversarial keys $\mathcal{K}_{\text{ADV}}$, or $\hat{\mathcal{K}}_x \equiv \mathcal{K}_{\text{ADV}}$, which corresponds precisely to the membership condition.

3.3 Theoretical Analysis of PromptMIA

In this section, we provide theoretical guarantees that PROMPTMIA achieves a perfect true positive rate (TPR) alongside a provably low false positive rate (FPR). Specifically, TPR is the adversary's success in identifying the target sample $\mathcal{T}$ when it is a member ($b = 1$), and FPR is the adversary's error in identifying $\mathcal{T}$ as a member when it is a non-member ($b = 0$).

(I) First, the PROMPTMIA attack is constructed to ensure perfect identification when the target sample is present. Theorem 3.1 establishes that the TPR of PROMPTMIA equals 1. The proof follows directly from the construction of adversarial keys in Algorithm 2, which ensures that $\min_{k_a \in \mathcal{K}_{\text{ADV}}} \kappa(q(\mathcal{T}), k_a) > \max_{k_b \in \mathcal{K}_{\text{BENIGN}}} \kappa(q(\mathcal{T}), k_b)$. Thus, if the client possesses $\mathcal{T} \in \mathcal{D}$, the top-$N$ selected keys must be exactly the set $\mathcal{K}_{\text{ADV}}$.

Theorem 3.1 (True Positive Rate)**.** *Let $\mathcal{K}_{\text{ADV}} = \{k_{a_m}\}_{m=1}^{N}$ be the set of N adversarial keys generated by Algorithm 2 with parameters $\delta_{\min} > 0$ and $\Delta \geq 0$. Let $\mathcal{K}_{\text{BENIGN}}$ be the set of $M - N$ benign keys. If the client's dataset $\mathcal{D}$ contains the target sample $\mathcal{T}$ (i.e., $b = 1$) and the client's selection mechanism (Eq. 1) selects the top-N prompts based on highest cosine similarity (lowest cosine distance), the set of selected keys $\hat{\mathcal{K}}_{\mathcal{T}}$ for the query $q(\mathcal{T})$ will be exactly the adversarial set: $\hat{\mathcal{K}}_{\mathcal{T}} = \mathcal{K}_{\text{ADV}}$. Consequently, the True Positive Rate (TPR) of* PROMPTMIA *is 1, or TPR* $= \Pr[b' = 1 \mid b = 1] = 1$

(II) We now analyze the FPR, $\Pr[b' = 1 \mid b = 0]$, which is more complex. A false positive arises whenever a non-member query $q(x)$ selects all N adversarial keys $\{k_{a_j}\}_{j=1}^{N}$ as its top-N choices than

to any benign key $\{k_{b_i}\}_{i=1}^{M-N}$. To mathematically derive a closed-form bound for this probability, it is necessary to formalize the geometric and statistical properties of the non-member query space. Because the exact distribution of high-dimensional query vectors is analytically intractable, we establish two foundational assumptions grounded in the inherent optimization dynamics of FPT.

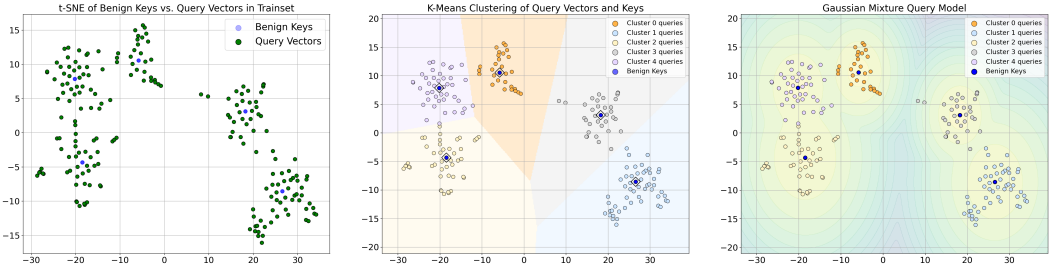


Figure 3: **Left:** t-SNE projection of the global keys and query vectors from the train set. **Center:** K-Means clustering of keys with queries. **Right:** Gaussian Mixture Query Model. Visualizations are generated from a ViT-B32 model trained on CIFAR-10 for 60 global epochs.

Assumption 1 (Benign Keys as Cluster Centroids). *We assume that the $K = M - N$ benign keys $k_{b_i} \in \mathcal{K}_{\text{BENIGN}}$ act as the effective centroids of the non-member query vector distribution. Each non-member query $q(x)$ is assumed to belong to the cluster of its nearest benign key.*

First, the objective (Eq. 2) includes a surrogate loss $\gamma(q(x), k_i)$ that explicitly pulls selected keys to align with their corresponding query features. Over multiple training rounds, this optimization forces the benign keys to stabilize and act as cluster centers, which formally justifies Assumption 1.

Assumption 2 (Gaussian Mixture Query Model). *We model the global marginal distribution of non-member queries $q(x)$ as a Gaussian Mixture Model (GMM). Specifically, the conditional distribution of queries belonging to benign cluster i is modeled as a spherical Gaussian centered at that key: $q(x) \mid (x \in \text{cluster } i) \sim \mathcal{N}(k_{b_i}, \sigma_i^2 I)$, where σ_i^2 is the local cluster variance.*

Second, Assumption 2 is motivated by the local alignment structure induced by benign keys. Queries assigned to the same benign key tend to concentrate around that key in the embedding space, while different benign keys define distinct local regions. Thus, we approximate each local region by a spherical Gaussian centered at its benign key, forming a tractable Gaussian mixture model for analysis. This is consistent with the standard use of Gaussian mixtures for density estimation and clustering [37, 38]. In addition, this approximation captures the empirical multi-cluster geometry visualized in Fig. 3.

Next, we introduce the adversarial key into this setting. For a non-member query $q(x)$ drawn from cluster i, we define a *cluster-flip* event E_i as the case when $q(x)$ selects all N adversarial keys as its N nearest centroids. Formally, E_i is the intersection of $N \times (M - N)$ *race* events A_{jl}, where each A_{jl} denotes that $q(x)$ is closer to an adversarial key k_{a_j} than to a benign key k_{b_l}. A cluster-flip event thus arises only if the adversary wins all of these races simultaneously. The probability of the joint event E_i can then be upper bounded by the probability of the single race with the lowest success probability, as formally stated in Lemma 3.2.

Lemma 3.2. *Let $q(x)$ be a non-member query drawn from the i-th component of the GMM, such that its conditional distribution is $q(x) \mid (x \in \text{cluster } i) \sim \mathcal{N}(k_{b_i}, \sigma_i^2 I)$. The conditional probability $\Pr(E_i)$, that $q(x)$ selects all N adversarial keys $\mathcal{K}_{ADV} = \{k_{a_1}, \ldots, k_{a_N}\}$ as its N closest centroids, is bounded by:*

$$\Pr(E_i) \leq \min_{\substack{1 \leq j \leq N \\ 1 \leq l \leq M-N}} \Phi\left(\frac{(k_{a_j} - k_{b_l})^T k_{b_i}}{\sigma_i \| k_{a_j} - k_{b_l} \|}\right)$$

where $\Phi(\cdot)$ is the Cumulative Distribution Function (CDF) of the standard normal distribution.

With this lemma, we can bound the FPR. The FPR is the probability of the event E_{FP} that a single, random non-member query $q(x)$ results in a cluster-flip event. Using the Law of Total Probability, FPR can be expressed as a weighted sum of cluster-flip probabilities, where the weights correspond to the prior probabilities of clusters. Based on the bound on the probability of cluster-flip events

for each cluster established in Lemma 3.2, we can derive a bound on the FPR which is the largest (worst-case) cluster-flip probability across all clusters, stated in Theorem 3.3.

Theorem 3.3 (False Positive Rate)**.** *The per-sample False Positive Rate (FPR) is bounded by:*

$$FPR = \Pr[b' = 1 \mid b = 0] \leq \max_{1 \leq i \leq M-N} \left(\min_{\substack{1 \leq j \leq N \\ 1 \leq l \leq M-N}} \Phi\left(z_{ijl}\right) \right)$$

where $\mathcal{K}_{ADV} = \{k_{a_1}, \ldots, k_{a_N}\}$ *is the set of* N *adversarial keys, and* $z_{ijl} = \frac{(k_{a_j} - k_{b_l})^T k_{b_i}}{\sigma_i \| k_{a_j} - k_{b_l} \|}$

Theorem 3.3 provides some insights on conditions under which the attack is most effective. First, the bound is tighter when the adversarial target $q(\mathcal{T})$ is highly distinctive. A distinctive target ensures that its corresponding adversarial keys are geometrically well separated from benign key clusters, resulting in small $\Phi\left(z_{ijl}\right)$. In addition, the FPR is lower when the benign data forms tight and compact clusters in the query space around the benign keys (i.e., minimizing σ_i^2). That will decreases the likelihood that any non-member query will randomly stray into the adversary's region. Additionally, combining Theorems 3.1 and 3.3 yields the **bound on the Advantage** stated in Corollary A.1. The detailed proofs are provided in Appendix A.8.

On the tightness of FPR bound established Theorem 3.3: In practice, the upper bound for the per-sample FPR remains ≈ 0 across all datasets, which corresponds to our 100% per-sample attack success rate (Sec. 4.1). We examine the upper FPR bound accross various settings in Appendix A.9.

4 Experimental Results

We evaluate PROMPTMIA on four datasets: CIFAR-10, CIFAR-100 [13], TinyImageNet [14], and a synthetic 4-dataset benchmark constructed by pooling MNIST-M [39], Fashion-MNIST [16], CINIC-10 [17], and MMAFEDB [2] and three different models: ViT-B32 [19], ConViT [21] and DeiT [20]. Experiments follow three axes: (1) measuring attack effectiveness via advantage and success rate and comparison against other federated MIAs; (2) testing robustness of PROMPTMIA against MIA defenses; and (3) conducting ablations on key hyperparameters (M, N, $\delta_{\min}$, Δ) as well as federated settings. Details about experimental settings are given in Appx. A.7.

4.1 PROMPTMIA PERFORMANCE EVALUATION

We evaluate the performance of PROMPTMIA against FPT using two metrics: Advantage (Eq. 3) and Attack Success Rate (Eq. 4). Unless otherwise specifically noted, we set $\delta_{\text{min}} = 0.02$ and $\Delta = 0.05$. Following [35], the inital global prompt pool size is set at $M = 20$, and the prompt selection size at $N = 4$. In the case of batched update, given batch $B = \{(x_i, y_i)\}_{i=1}^{\ell}$, for each sample the client computes the per-sample selected key set $\hat{\mathcal{K}}_{x_i}$ and corresponding per-sample loss $\mathcal{L}_{x_i}$ (defined in Eq. 2) and updates the batch-level set of chosen keys and prompts as: $\hat{\mathcal{K}}_B = \hat{\mathcal{K}}_B \bigcup \hat{\mathcal{K}}_{x_i}$ and $\hat{\mathcal{P}}_B = \hat{\mathcal{P}}_B \bigcup \mathcal{P}_{x_i}$. Batch-wise loss $\mathcal{L}_B$ is calculated by accumulating $\mathcal{L}_{x_i}$. The client updates the selected keys and prompts as $\hat{\mathcal{K}}_B \leftarrow \mathcal{K}_B - \mu \nabla_{\hat{\mathcal{K}}_B} \mathcal{L}_B$ and $\hat{\mathcal{P}}_B \leftarrow \hat{\mathcal{P}}_B - \mu \nabla_{\hat{\mathcal{P}}_B} \mathcal{L}_B$, where μ is the learning rate. After receiving the client's updates, the server infers membership by checking whether all adversarial prompts are selected, i.e. $\mathbb{I}_{\{\mathcal{T} \in \mathcal{D}\}} = 1$ if $\mathcal{K}_{\text{ADV}} \subseteq \hat{\mathcal{K}}_B$, and 0 otherwise. Fig. 4 shows average results of three models over four different datasets. **Detailed results and error bands on individual models** are given in Appx. A.11. PROMPTMIA achieves near-perfect attack success rates across all models and datasets at small batch sizes, and maintains $> 90\%$ ASR at larger batch sizes. In contrast, *naive prompt injection attack* collapses as the batch size increases, reaching Advantage of only $\approx 20\%$ against FourDataset when batch size $= 256$.

PROMPTMIA vs. existing FedMIAs. We also compare PROMPTMIA against existing MIAs. Most MIAs rely on the observation that ML models behave differently on training data and on unseen data [22], and exploit signals such as gradient norms, gradient cosine similarity, or loss values. Gradient-based methods are not directly applicable to FPT, where clients exchange prompts rather than gradients. We therefore compare PROMPTMIA against MIAs that use loss values as

[2] `https://www.kaggle.com/datasets/yuulind/mmafedb-clean`

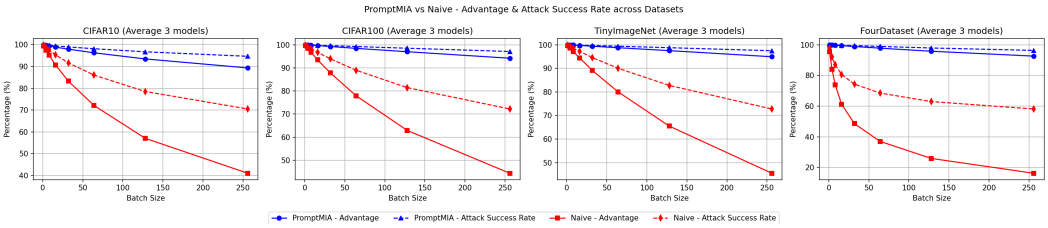


Figure 4: Performance of PromptMIA vs Naive **averaged across three models**. Each subplot shows Advantage and A.S.R w.r.t Batch Size across CIFAR10, CIFAR100, TinyImageNet, and FourDataset.

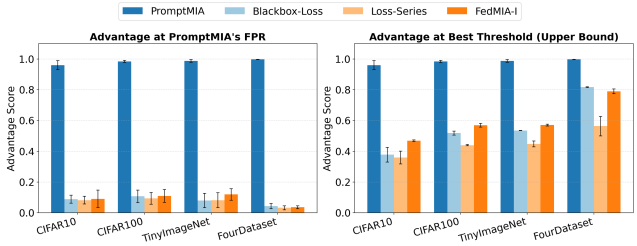


Figure 5: Comparison of PROMPTMIA vs. existing federated MIAs (batch size = 64).

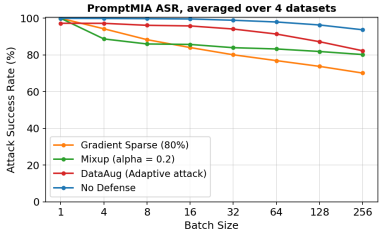


Figure 6: Impact of regularization-based MIA defenses.

the membership signal: Blackbox-Loss [24], Loss-Series [23], and FedMIA-I [22]. Since these attacks output a continuous score, their Advantage depends on the decision threshold, and we report two settings (Fig. 5). (i) At PROMPTMIA's operating point, each baseline's threshold is set to match PROMPTMIA's FPR on the same model. Here PROMPTMIA reaches an Advantage of 0.96–1.00, whereas the baselines remain at or below 0.12. (ii) At the best threshold, each baseline uses the threshold that maximizes its Advantage, chosen using the true membership labels. This is an upper bound that no real attacker can achieve. Even then, the baselines reach only 0.36–0.82, and PROMPTMIA, which uses no such tuning, outperforms all of them on every dataset. Structured comparision of PROMPTMIA and existing FedMIAs are given in Appx. A.4.

4.2 MIA Defenses against PROMPTMIA

Although the primary objective of this work is not to propose novel defense mechanisms or conduct an exhaustive evaluation of all possible defense strategies against PROMPTMIA, we examine how standard MIA defenses interact with PROMPTMIA. Details about theses defenses are given in Appendix A.5 and A.6.

(I) Modern MIA Defenses: Most existing MIAs rely on models overfitting their training data [22, 23, 24, 6], so modern MIA defenses are mostly regularization-based: gradient sparsification [30], mixup [31], and data augmentation [32]. Fig. 6 evaluates these defenses in a realistic setting, where the client applies the defense both during training and in the attacked round. Results are averaged over four datasets (CIFAR-10, CIFAR-100, TinyImageNet, and FourDataset) with ViT-B/32. *Gradient sparsification* keeps only the top 20% of gradient entries, which can prune the updates to the injected prompts. *Mixup* and *data augmentation* represent challenging scenarios, since they depart from the security game of Sec. 3.1: the client no longer trains on the target itself, but on a modified version of it (a blend with another image, or an augmented copy). Even so, PROMPTMIA remains a significant threat under both gradient sparsification and mixup, including at large batch sizes. For data augmentation, we propose a simple modification that makes the attack adaptive to the defense by constructing adversarial keys using augmented copies of the target sample (Appx. A.3). Overall, modern regularization-based MIA defenses do not eliminate the threat posed by PROMPTMIA.

(II) Another line of defense is to use **Anomaly Detection Methods** to filter out adversarial prompts. We run four classical outlier detectors on the M-key global pool to flag the N injected adversarial keys. We use an injection rate of 10%, i.e., each round is attacked with probability 0.1 (Appendix A.12). Since PROMPTMIA succeeds in a single round, the true injection rate, and thus the attainable precision, is even lower. As shown in Table1, these methods are ineffective against PROMPTMIA,

with Figures 15 and 16 visualizing their behavior. Although evaluating deep learning-based anomaly detection algorithms [40, 41] would be interesting, we leave this for future work; moreover, the prompt pool is typically small, which raises doubts about the effectiveness of such data-hungry methods. Moreover, the number of prompts in the prompt pool is typically small, which raises doubts about the effectiveness of deep learning based methods.

(III) Input Noise Perturbation. Inspired by PixelDP [29], we add fresh Gaussian noise $\eta \sim \mathcal{N}(0, \sigma^2 I)$ to normalized input images [29] during every local training step and during attack evaluation. We set $\sigma = L\sqrt{2\ln(1.25/\delta)}/\epsilon$, with $\delta = 1/n_{\text{train}}$ and $L = 1$, where n_{train} denotes the number of training samples. Smaller values of ϵ produce stronger input perturbations. Similar to the data augmentation setting, we evaluate an adaptive variant of PROMPTMIA that constructs adversarial keys using simulated noisy copies of the target (see Appendix A.3). In particular, we measure the attack success rate (ASR) under $\epsilon \in \{3, 5, 8\}$ using ViT-B/32 across four datasets, as shown in Fig. 7. Table 2 reports the corresponding privacy–utility trade-off, assessed through attack success and classification accuracy. Defending against PROMPTMIA with input noise perturbation presents a pronounced privacy–utility trade-off. Stronger noise (e.g., $\epsilon = 3$) reduces attack success at larger batch sizes, but at a substantial cost to model accuracy: on CIFAR-100, accuracy drops from 73.5% without noise to 42.5%. Conversely, moderate noise levels ($\epsilon = 5$ or 8) better preserve accuracy but offer limited protection, leaving the model highly vulnerable to adaptive PROMPTMIA.

(IV) System-level Defenses: We also analyze the impact of system-level defenses that modify the prompt selection or aggregation mechanism against PROMPTMIA such as randomized key indices, secure aggregation [33], randomized prompt selection via prompt dropout and cross-round key-drift auditing. In Appendix A.6, we show that these defenses are either provably or empirically ineffective.

Table 1: Outlier detection results **averaged over datasets and models**. See Appx. A.12.

Method	Precision	Recall	F1
IsolationForest[25]	0.04(0.01)	1.00(0.00)	0.08(0.02)
LocalOutlierFactor[26]	0.00(0.00)	0.00(0.00)	0.00(0.00)
OneClassSVM[28]	0.05(0.01)	0.49(0.03)	0.08(0.02)
EllipticEnvelope[27]	0.03(0.01)	0.24(0.03)	0.05(0.02)

Table 2: Model accuracy (%) under input noise training with different ϵ (ViT-B32).

Dataset	$\epsilon = 3$	$\epsilon = 5$	$\epsilon = 8$	Non-DP
CF-10	82.7 (0.6)	87.9 (0.5)	89.9 (0.5)	93.5 (0.1)
CF-100	42.5 (0.2)	54.4 (0.8)	61.8 (0.8)	73.5 (0.6)
TinyIN	69.3 (0.1)	74.3 (0.1)	77.7 (0.4)	81.5 (0.2)
4-dataset	38.9 (1.1)	53.3 (1.3)	60.2 (0.8)	68.1 (1.3)

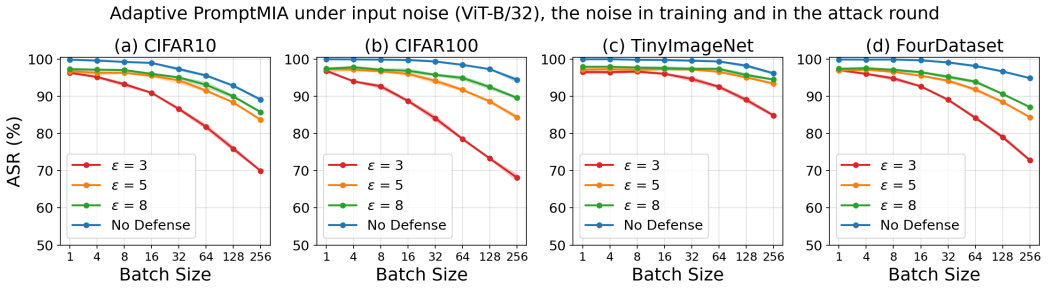


Figure 7: ASR of adaptive PROMPTMIA (ViT-B/32) under Input Noise Perturbation with different ϵ.

4.3 Additional Experiments

In Appx. A.13, we analyze how key hyperparameters impact PROMPTMIA's performance. The most interesting finding is that the ASR of PROMPTMIA is much higher on the models that have been trained for a few epochs rather than randomly initialized keys (Fig. 17e). This corresponds to our theoretical findings in Section 3.3 that FPR is lower when the benign data forms tight and compact clusters in the query space around the benign keys, which happens naturally during the training process (Appx. A.14). We provide additional experiments on various settings such as non-iid and imbalalanced data (Appx. A.10), very large batch size (Appx. A.15), text and multimodel data (Appx. A.16), large number of clients (Appx. A.17), target distinctiveness (Appx. A.18) and downstream task accuracy (Appx. A.19). We also discuss additional metrics such as Attack Sparsity, Computational Overhead and Scalability in Appx. A.20. We discuss PROMPTMIA's limitations and future research directions in Appx. A.21.

5 Related Work

Membership Inference Attacks against Federated Learning. The goal of MIAs in FL is to identify if a specific data point was part of a client's training set. Active Membership Inference (AMI) attacks involve a dishonest server poisoning the global models, e.g., maliciously modifying model parameters, before dispatching them to clients [42, 43]. Our work follows the standard AMI protocol. Recently, FedMIA [22] proposed a three-step attack that leverages updates from all clients that can be integrated as an extension to existing attacks.

Federated Fine-Tuning of Foundation Models. Federated fine-tuning avoids updating the entire model and instead allows clients to fine-tune only a subset of parameters, such as LoRA-[44, 45, 46] and adapter-based approaches [47, 48]. Recently, prompt-based federated fine-tuning approaches have been proposed [49, 7, 50, 51], which update soft prompts instead of full model weights. Our work focuses on MIAs under FPT setting since these approaches are architecturally orthogonal.

Relation to broader privacy attacks. In in-context learning, adversaries can craft malicious prompts to extract confidential information in discrete prompts [8, 9]. Active attacks such as backdoor injection [52] or representation steering [53, 54] are typically designed to manipulate model behavior or internal representations rather than infer membership (Appx. A.22). Our work is the first to investigate the privacy risks of soft prompts in FPT.

6 Ethical Considerations

Intended use. PROMPTMIA is intended solely for privacy auditing and security evaluation of federated prompt-tuning (FPT) systems. We publish it to expose a new attack surface and to dispel the assumption that FPT guarantees data protection. While FPT reduces privacy risk by not sharing raw data, it does not eliminate that risk. All our experiments use public benchmark datasets in simulated federations; no deployed system was attacked, and no inference was performed against any real individual's data.

Practical safeguards. Clients should not treat a central aggregator as a trusted party, and should adopt additional safeguards in sensitive deployments despite their costs. Mechanisms that prevent the server from observing individual client updates, such as secure aggregation (Appendix A.6) or multi-party management of the prompt pool, reduce the attack from client-level to group-level membership inference, at additional communication and computation cost.

Responsible disclosure. PROMPTMIA targets the design of the prompt-selection protocol in FPT rather than a defect in a particular implementation, and we are not aware of a commercial FPT deployment with a maintainer to whom conventional embargoed disclosure would apply. In place of vendor notification, we publish the attack so that the risk is known to researchers and implementers before such systems reach wider deployment. We release the code at `https://github.com/GivralNguyen/Prompt-MIA` with a clear intended-use statement, and we document the safeguards above together with their costs and limits, so that practitioners are not left with a demonstrated attack and no guidance.

7 Conclusion

In conclusion, we show that FPT introduces a new and critical privacy vulnerability. Through PROMPTMIA, we demonstrate that a malicious server can leverage adversarial prompts to reliably infer client membership, achieving high attack success rates across multiple datasets. Our theoretical analysis explains the attack's robustness by establishing a lower bound on the attack's advantage. Finally, evaluation of PROMPTMIA against existing MIA defenses reveals their limitations in this setting, underscoring the need for new defense strategies specifically designed for FPT.

Acknowledgments and Disclosure of Funding

This work is partially supported by NSF grant IIS-2416606 and NIH/NIDDK grant R01DK142809. Any opinions, findings, and conclusions expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[2] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 27–38, 2017.

[3] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.

[4] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[5] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 691–706. IEEE, 2019.

[6] Jiacheng Li, Ninghui Li, and Bruno Ribeiro. Effective passive membership inference attacks in federated learning against overparameterized models. In *The Eleventh International Conference on Learning Representations*, 2023.

[7] Pei-Yau Weng, Minh Hoang, Lam Nguyen, My T Thai, Lily Weng, and Nghia Hoang. Probabilistic federated prompt-tuning with non-iid and imbalanced data. *Advances in Neural Information Processing Systems*, 37:81933–81958, 2024.

[8] Rui Wen, Zheng Li, Michael Backes, and Yang Zhang. Membership inference attacks against in-context learning. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 3481–3495, 2024.

[9] Haonan Duan, Adam Dziedzic, Nicolas Papernot, and Franziska Boenisch. Flocks of stochastic parrots: Differentially private prompt learning for large language models. *Advances in Neural Information Processing Systems*, 36:76852–76871, 2023.

[10] Tong Wu, Ashwinee Panda, Jiachen T Wang, and Prateek Mittal. Privacy-preserving in-context learning for large language models. *arXiv preprint arXiv:2305.01639*, 2023.

[11] Junyuan Hong, Jiachen T Wang, Chenhui Zhang, Zhangheng Li, Bo Li, and Zhangyang Wang. Dp-opt: Make large language model your privacy-preserving prompt engineer. *arXiv preprint arXiv:2312.03724*, 2023.

[12] Xinyu Tang, Richard Shin, Huseyin A Inan, Andre Manoel, Fatemehsadat Mireshghallah, Zinan Lin, Sivakanth Gopi, Janardhan Kulkarni, and Robert Sim. Privacy-preserving in-context learning with differentially private few-shot generation. *arXiv preprint arXiv:2309.11765*, 2023.

[13] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[14] Yann Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

[15] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of machine learning research*, 17(59):1–35, 2016.

[16] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[17] Luke N Darlow, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

[18] M. A. Mahmoudi. Mma facial expression dataset. https://www.kaggle.com/datasets/mahmoudima/mma-facial-expression, June 2020. Accessed: 2026-04-30.

[19] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, G Heigold, S Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

[20] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.

[21] Stéphane d'Ascoli, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *International conference on machine learning*, pages 2286–2296. PMLR, 2021.

[22] Gongxi Zhu, Donghao Li, Hanlin Gu, Yuan Yao, Lixin Fan, and Yuxing Han. Fedmia: An effective membership inference attack exploiting" all for one" principle in federated learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 20643–20653, 2025.

[23] Yuhao Gu, Yuebin Bai, and Shubin Xu. Cs-mia: Membership inference attack based on prediction confidence series in federated learning. *Journal of Information Security and Applications*, 67:103201, 2022.

[24] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)*, pages 268–282. IEEE, 2018.

[25] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008.

[26] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

[27] Peter J. Rousseeuw and Katrien Van Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41(3):212–223, 1999.

[28] Larry M Manevitz and Malik Yousef. One-class svms for document classification. *Journal of machine Learning research*, 2(Dec):139–154, 2001.

[29] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE symposium on security and privacy (SP)*, pages 656–672. IEEE, 2019.

[30] Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. *Advances in Neural Information Processing Systems*, 31, 2018.

[31] Yangsibo Huang, Zhao Song, Kai Li, and Sanjeev Arora. Instahide: Instance-hiding schemes for private distributed learning. In *International conference on machine learning*, pages 4507–4518. PMLR, 2020.

[32] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[33] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[34] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[35] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 139–149, 2022.

[36] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[37] Jonathan Q. Li and Andrew R. Barron. Mixture density estimation. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'99, page 279–285, Cambridge, MA, USA, 1999. MIT Press.

[38] Lianmeng Jiao, Thierry Denœux, Zhun ga Liu, and Quan Pan. Egmm: An evidential version of the gaussian mixture model for clustering. *Applied Soft Computing*, 129:109619, 2022.

[39] Seunghun Lee, Sunghyun Cho, and Sunghoon Im. Dranet: Disentangling representation and adaptation networks for unsupervised cross-domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15252–15261, 2021.

[40] Nguyen Hoang Khoi Do, Truc Nguyen, Malik Hassanaly, Jung Taek Seo, My T Thai, et al. Swift hydra: Self-reinforcing generative framework for anomaly detection with multiple mamba models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[41] Minqi Jiang, Chaochuan Hou, Ao Zheng, Songqiao Han, Hailiang Huang, Qingsong Wen, Xiyang Hu, and Yue Zhao. Adgym: Design choices for deep anomaly detection. *Advances in Neural Information Processing Systems*, 36:70179–70207, 2023.

[42] Truc Nguyen, Phung Lai, Khang Tran, NhatHai Phan, and My T Thai. Active membership inference attack under local differential privacy in federated learning. *arXiv preprint arXiv:2302.12685*, 2023.

[43] Minh Vu, Truc Nguyen, My T Thai, et al. Analysis of privacy leakage in federated large language models. In *International Conference on Artificial Intelligence and Statistics*, pages 1423–1431. PMLR, 2024.

[44] Jiaxing Qi, Zhongzhi Luan, Shaohan Huang, Carol Fung, Hailong Yang, and Depei Qian. Fdlora: Personalized federated learning of large language model via dual lora tuning. *arXiv preprint arXiv:2406.07925*, 2024.

[45] Ziyao Wang, Zheyu Shen, Yexiao He, Guoheng Sun, Hongyi Wang, Lingjuan Lyu, and Ang Li. Flora: Federated fine-tuning large language models with heterogeneous low-rank adaptations. *Advances in Neural Information Processing Systems*, 37:22513–22533, 2024.

[46] Boyu Fan, Xiang Su, Sasu Tarkoma, and Pan Hui. Helora: Lora-heterogeneous federated fine-tuning for foundation models. *ACM Transactions on Internet Technology*, 25(2):1–22, 2025.

[47] Dongqi Cai, Yaozong Wu, Shangguang Wang, Felix Xiaozhu Lin, and Mengwei Xu. Fedadapter: Efficient federated learning for modern nlp. *arXiv preprint arXiv:2205.10162*, 2022.

[48] Sajjad Ghiasvand, Yifan Yang, Zhiyu Xue, Mahnoosh Alizadeh, Zheng Zhang, and Ramtin Pedarsani. Communication-efficient and tensorized federated fine-tuning of large language models. *arXiv preprint arXiv:2410.13097*, 2024.

[49] Shangchao Su, Mingzhao Yang, Bin Li, and Xiangyang Xue. Federated adaptive prompt tuning for multi-domain collaborative learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 15117–15125, 2024.

[50] Sikai Bai, Jie Zhang, Song Guo, Shuaicheng Li, Jingcai Guo, Jun Hou, Tao Han, and Xiaocheng Lu. Diprompt: Disentangled prompt tuning for multiple latent domain generalization in federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27284–27293, 2024.

[51] Yu Feng, Zhen Tian, Yifan Zhu, Zongfu Han, Haoran Luo, Guangwei Zhang, and Meina Song. Cp-prompt: Composition-based cross-modal prompting for domain-incremental continual learning. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 2729–2738, 2024.

[52] Shuai Zhao, Meihuizi Jia, Zhongliang Guo, Leilei Gan, Xiaoyu Xu, Xiaobao Wu, Jie Fu, Yichao Feng, Fengjun Pan, and Luu Anh Tuan. A survey of recent backdoor attacks and defenses in large language models. *arXiv preprint arXiv:2406.06852*, 2024.

[53] Zhengxuan Wu, Aryaman Arora, Atticus Geiger, Zheng Wang, Jing Huang, Dan Jurafsky, Christopher D Manning, and Christopher Potts. Axbench: Steering llms? even simple baselines outperform sparse autoencoders. *arXiv preprint arXiv:2501.17148*, 2025.

[54] Zhengxuan Wu, Qinan Yu, Aryaman Arora, Christopher D Manning, and Christopher Potts. Improved representation steering for language models. *arXiv preprint arXiv:2505.20809*, 2025.

[55] Claude E Shannon. Communication theory of secrecy systems. *The Bell system technical journal*, 28(4):656–715, 1949.

[56] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 739–753. IEEE, 2019.

[57] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *2022 IEEE symposium on security and privacy (SP)*, pages 1897–1914. IEEE, 2022.

[58] Oliver Kramer. Scikit-learn. In *Machine learning for evolution strategies*, pages 45–53. Springer, 2016.

[59] Ignazio Gallo, Gianmarco Ria, Nicola Landro, and Riccardo La Grassa. Image and text fusion for upmc food-101 using bert and cnns. In *2020 35th International Conference on Image and Vision Computing New Zealand (IVCNZ)*, pages 1–6. IEEE, 2020.

[60] Thu Hang Phung, Duong M Nguyen, Thanh Trung Huynh, Quoc Viet Hung Nguyen, Trong Nghia Hoang, and Phi Le Nguyen. Federated prompt-tuning with heterogeneous and incomplete multimodal client data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3936–3946, 2025.

A Appendix

A.1 Federated Prompt Tuning

Figure 8 illustrates the workflow of federated prompt tuning. In federated prompt tuning, a central server maintains a global pool of prompts and keys. Each client selects the top-N prompts most similar to its input features, updates those prompts and a lightweight classifier locally, and sends the updated prompts back. The server then aggregates the clients' prompts to refine the global prompt pool without sharing raw data.

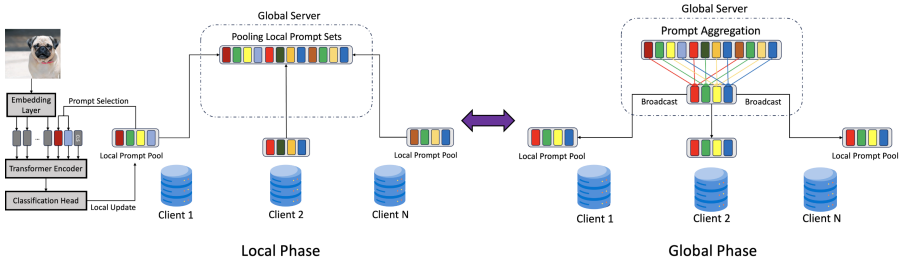


Figure 8: (Local Phase) each client samples and fine-tunes a subset of global summarizing prompts using a prompt-selection strategy; (Global Phase) the server aggregates all local prompt sets to refine the global prompt pool.

A.2 Generating adversarial keys set $\mathcal{K}_{\text{ADV}}$

To mount PROMPTMIA, the adversarial server must generate a set of keys $\mathcal{K}_{\text{ADV}}$ that are more similar to the target query vector $q(\mathcal{T})$ than all benign keys $\mathcal{K}_{\text{BENIGN}}$ while remaining diverse enough to avoid detection.

GENKEYWITHSIMILARITY (Alg. 1) constructs a single adversarial key with a desired cosine similarity s to the target query. It first *normalizes* the target query $q(\mathcal{T})$ to obtain the unit vector $\hat{q} = q(\mathcal{T})/\|q(\mathcal{T})\|$. Then it *samples a random vector* $r \in \mathbb{R}^{D_k}$ and *removes the component of* r *that lies along* $\hat{q}$ by computing $o = r - \langle r, \hat{q}\rangle \hat{q}$. This ensures that o is orthogonal to the target direction. Next, o is normalized to $\hat{o} = o/\|o\|$, producing a unit vector exactly perpendicular to $\hat{q}$. To construct a vector with a desired cosine similarity s to $\hat{q}$, the algorithm forms

$$\hat{v} = s \cdot \hat{q} + \sqrt{1 - s^2} \cdot \hat{o},$$

Finally, $\hat{v}$ is *rescaled* to match the original norm of $q(\mathcal{T})$, producing the adversarial key $k_a = \hat{v} \cdot \|q(\mathcal{T})\|$.

GENADVKEYSET (Alg. 2) builds the entire adversarial key set $\mathcal{K}_{\text{ADV}}$. It computes the maximum similarity $s_{\max}$ between the target query and existing benign keys, then samples N similarity scores uniformly from the interval

$$[s_{\max} + \delta_{\min},\ s_{\max} + \delta_{\min} + \Delta],$$

ensuring that each adversarial key is slightly closer to the target query than any benign key by at least $\delta_{\min}$ but not so close that all keys collapse to the target query. Each sampled similarity s_m is used to generate a key via GENKEYWITHSIMILARITY, producing a diverse set of keys that satisfy the required similarity bounds.

Algorithm 1 GENKEYWITHSIMILARITY$(q(\mathcal{T}), s)$

Require: Target query vector $q(\mathcal{T}) \in \mathbb{R}^{D_k}$, desired cosine similarity $s \in (0, 1)$
Ensure: Adversarial key $k_a \in \mathbb{R}^{D_k}$ such that $\kappa(k_a, q(\mathcal{T})) \approx s$ and $\|k_a\| = \|q(\mathcal{T})\|$

$\hat{q} \leftarrow q(\mathcal{T})/\|q(\mathcal{T})\|$ {Normalize the target vector}
$r \sim \mathcal{U}(0,1)^{D_k}$; $\hat{r} \leftarrow r/\|r\|$ {Sample a random vector}
$o \leftarrow \hat{r} - \langle \hat{r}, \hat{q} \rangle \cdot \hat{q}$ {Remove component along $\hat{q}$}
$\hat{o} \leftarrow o/\|o\|$ {Normalize orthogonal component}
$\hat{v} \leftarrow s \cdot \hat{q} + \sqrt{1 - s^2} \cdot \hat{o}$ {Combine to enforce similarity s}
$k_a \leftarrow \hat{v} \cdot \|q(\mathcal{T})\|$ {Rescale to match original norm}
return k_a {Adversarial key}

Algorithm 2 GENADVKEYSET$(q(\mathcal{T}), \mathcal{K}_{\text{BENIGN}}, \delta_{\min}, \Delta, N)$

Require: Target query vector $q(\mathcal{T}) \in \mathbb{R}^{D_k}$, benign key set $\mathcal{K}_{\text{BENIGN}}$, margins $\delta_{\min}, \Delta$, number of adversarial keys N

Ensure: $\kappa(q(\mathcal{T}), k_{a_m}) \in \Big[\max_{k_b \in \mathcal{K}_{\text{BENIGN}}} \kappa(q(\mathcal{T}), k_b) + \delta_{\min},\ \max_{k_b \in \mathcal{K}_{\text{BENIGN}}} \kappa(q(\mathcal{T}), k_b) + \delta_{\min} + \Delta\Big], \forall\, k_{a_m} \in \mathcal{K}_{\text{ADV}}$

$\hat{q} \leftarrow q(\mathcal{T})/\|q(\mathcal{T})\|$ {Normalize target query}
$s_{\max} \leftarrow \max_{k_b \in \mathcal{K}_{\text{BENIGN}}} \kappa(\hat{q}, k_b)$ {Maximum similarity to benign keys}
for $m = 1$ to N **do**
  Sample $s_m \sim \mathcal{U}(s_{\max} + \delta_{\min},\ s_{\max} + \delta_{\min} + \Delta)$
  $k_{a_m} \leftarrow$ GENKEYWITHSIMILARITY$(q(\mathcal{T}), s_m)$
end for
return Adversarial key set $\mathcal{K}_{\text{ADV}} = \{k_{a_m}\}_{m=1}^{N}$

A.3 Adaptive PROMPTMIA against input-transforming defenses

Input-transforming defenses modify each training sample before it reaches the model, so a client selects prompts (Eq. 1) using the query of a *transformed* target rather than the clean query $q(\mathcal{T})$ around which the adversarial keys $\mathcal{K}_{\text{ADV}}$ were built (Eq. 7). Because the transformation is random and redrawn at every local step, a key tuned to $q(\mathcal{T})$ may no longer be selected, which weakens the attack. Following Kerckhoffs's principle [55] that a system's security must not rely on the secrecy of its mechanism (i.e., security through obscurity), as well as the standard assumption in modern membership inference that the adversary has full knowledge of the client's training protocol [56, 57], we adapt PROMPTMIA as follows. The server knows the client's defense pipeline, since it is part of the training protocol, but not the specific random transformation applied in any given step. In both settings below, the server estimates the *expected* transformed query and builds keys that remain selected under the transformation's randomness; the two settings differ only in how the transformed copies are generated.

Concretely, the server draws K transformed copies $\tilde{\mathcal{T}}_1, \ldots, \tilde{\mathcal{T}}_K$ of the target and takes their mean query direction

$$c = \frac{\bar{q}}{\|\bar{q}\|}, \qquad \bar{q} = \frac{1}{K} \sum_{k=1}^{K} \frac{q(\tilde{\mathcal{T}}_k)}{\|q(\tilde{\mathcal{T}}_k)\|}. \tag{8}$$

It overwrites the N keys nearest c with adversarial keys constructed as in Alg. 2, but centered on c instead of $q(\mathcal{T})$. The only remaining choice is how strongly to align the keys with c: rather than fixing the margin at the lower end of Eq. 7, the server raises the target similarity to the *smallest* value at which at least 95% of the K copies still select all N adversarial keys, , falling back to fully aligned keys (similarity 1) if no value reaches this coverage. Requiring high coverage makes the keys robust to the random transformation, while choosing the smallest sufficient similarity keeps the keys as far from c as possible, limiting the false positives induced by non-member batches. We use $K = 32$.

A.3.1 Data augmentation

For data augmentation [32], each copy $\tilde{\mathcal{T}}_k$ is an independent augmentation of the target drawn from the client's augmentation pipeline, so the client selects prompts with $q(\tilde{\mathcal{T}})$ rather than $q(\mathcal{T})$. The server applies the construction above directly, sampling the K copies from the known augmentation distribution.

A.3.2 Input noise perturbation

For the input-noise defense of Sec. 4.2 [29], each copy is a noisy target $\tilde{\mathcal{T}}_k = \mathcal{T} + \eta_k$ with $\eta_k \sim \mathcal{N}(0, \sigma^2 I)$ added to the normalized input, using the same σ as the client's trained model. The adaptive construction is otherwise identical: the server forms c from Eq. 8 over the K noisy copies, overwrites the N keys nearest c, and calibrates the similarity to the 95%-coverage threshold, again with $K = 32$.

A.4 Comparision of PROMPTMIA and existing Federated MIAs.

In Table 3, we provide a structured comparison summarizing existing MIA approaches in federated learning, along with their assumptions, attack surfaces, and key differences from PromptMIA. To summarize, the attack surface of PromptMIA is fundamentally different from traditional (sets of global prompts v.s. gradient vectors/ loss values). Existing MIAs are based on the observation that ML models behave differently on training data and unseen data and can be circumvented through regularization-based defenses. PromptMIA also significantly outperforms these methods (Fig. 5). To ensure a fair comparison, we report Blackbox-Loss, Loss-Series, and FedMIA-I both at their best threshold, which maximizes Advantage over the full ROC curve and is an upper bound for these attacks since it is chosen using the evaluation labels, and at a threshold matched to PROMPTMIA's FPR on the same model, using the largest FPR that does not exceed it.

#	Attack	Attack Surface	Attack Methodology	Assumptions
1	Blackbox-Loss [24] Loss-Series [23]	Data Loss	Model tends to have lower loss on training points than on non-training points. [2] uses loss values over multiple rounds.	Bounded loss and Gaussian error distributions
2	Grad-Cosine [6]	Gradient vector	The distributions of the cosine similarity of instances from the training data (members) and from non-members are different.	Gradient vectors of different instances are orthogonal & their cosine similarity is Gaussian.
3	FedMIA [22]	Data Loss (Fed-MIA-I), Gradient vector (Fed-MIA-II)	Update distribution of clients trained with target data is different from non-target clients	Update distribution follows Gaussian distribution
4	**PromptMIA (ours)**	Global prompt pool in FPT	Attacker injects adversarial prompts that will always be updated when target data is present.	Each query vector is modeled as a Gaussian mixture over benign keys

Table 3: Comparison of PROMPTMIA and existing Membership Inference Attacks.

A.5 Membership Inference Defenses

Although the primary objective of this work is not to propose novel defense mechanisms or conduct an exhaustive evaluation of all possible defense strategies against PROMPTMIA, we examine how standard MIA defenses interact with PROMPTMIA.

Noise Perturbation Rather than obfuscating gradient updates, clients can add calibrated noise directly to the input image, which provides (ϵ, δ)-DP per noisy access with respect to input changes of bounded ℓ_2 norm [29]. This introduces randomness to Eq. 1:

$$\mathcal{K}_x = \underset{\{s_i\}_{i=1}^N \subseteq [M]}{\operatorname{argmax}} \sum_{i=1}^{N} \kappa\big(q(x+\eta), k_{s_i}\big), \qquad \eta \sim \mathcal{N}(0, \tilde{\sigma} I). \tag{9}$$

Increasing the noise variance strengthens the protection by making the prompt selection less predictable and reducing the effectiveness of adversarially crafted keys. However, this also comes at the high cost of model utility.

Regularization-based MIA Defenses: We evaluate PROMPTMIA against regularization-based MIA Defenses, namely Gradient Sparsification [30], MixUp [31] and Data Augmentation[32]. For Gradient Sparsification, we use 80% sparsification rate. For MixUp, we use $\alpha_{\text{mixup}} = 0.2$. For Data Augmentation, we employ a combination of random cropping, horizontal flipping, and color jittering.

Anomaly detection techniques We test commonly used anomaly detection techniques in machine learning against PROMPTMIA. Specifically, we consider the following classical approaches: `IsolationForest` [25], `LocalOutlierFactor` [26], `OneClassSVM` [28], and `EllipticEnvelope` [27]. These methods are widely used, well established in the literature, and directly available in SCIKIT-LEARN library [58]. In our implementation, we use SCIKIT-LEARN version 1.5.2. Although it would be interesting to evaluate deep learning-based anomaly detection algorithms [40, 41] (e.g., autoencoders, VAEs, GAN-based models) against PROMPTMIA, we leave this for future work. Moreover, the number of prompts in the prompt pool is typically small (20 in our experiments), which raises doubts about the effectiveness of deep learning based methods in this setting. We therefore restrict our evaluation to the aforementioned classical methods. We briefly describe the anomaly detection algorithms used in this paper:

- **Isolation Forest** detects anomalies by recursively partitioning the feature space with random splits. Each sample's path length, or the number of splits needed to isolate it in a random tree is shorter for outliers, as they are easier to separate from the bulk of the data. Averaging this path length over a forest of random trees yields an anomaly score: points with shorter average paths are more likely to be anomalous.
- **Local Outlier Factor (LOF)** detects anomalies by comparing the local density of each sample to that of its k-nearest neighbors. Normal points have similar density to their neighbors, while outliers lie in sparser regions. The LOF score is the ratio of the average neighbor density to the sample's own density; values LOF > 1 indicate potential outliers. LOF captures both local and global structure, making it effective in datasets with varying densities.
- **One-Class Support Vector Machine (OCSVM)** is an unsupervised anomaly detection method derived from the Support Vector Machine framework. Instead of separating multiple classes, OCSVM learns a decision boundary that encloses the majority (normal) data in feature space, labeling points outside this region as anomalies or novelties. It works by maximizing the margin around normal data to create a robust "normalcy region" using a kernel function (commonly the radial basis function) to capture non-linear patterns.
- **Elliptic Envelope** is an outlier detection method that assumes inliers follow a known distribution, typically Gaussian. It fits a robust estimate of the data's mean and covariance (using the Minimum Covariance Determinant estimator) to capture the central elliptical shape of normal data while ignoring outliers. Points are then scored by their Mahalanobis distance from this fitted ellipse, with distant points flagged as anomalies. This approach is effective when the normal data distribution is approximately Gaussian.

A.6 System-level Defenses

We also analyze the impact of system-level defenses that modify the prompt selection or aggregation mechanism against PROMPTMIA.

- **Randomized key indices.** We consider the setting where, after training, clients randomly permute the indices of their prompt keys before sending updates to the server to prevent the server from knowing exactly which prompts were updated. However, this defense is

fundamentally ineffective against PROMPTMIA. The server already stores the previous prompt pool and can check whether each adversarial prompt appears in the client-updated pool via content matching, which is unaffected by index permutation. It then knows that the client selected and updated all adversarial prompts when no match is found. Index randomization is thus provably ineffective as it cannot prevent the server from knowing which prompts were updated.

- **Secure Aggregation.** Another strategy is to apply secure aggregation [33] so that individual client updates are hidden from the server. The intuition is that, if the server only observes an aggregated result rather than each client's prompt updates, it may be unable to determine whether a particular adversarial prompt was selected. However, current secure aggregation protocols are developed for linear aggregation (that is, weighted averaging) of local model parameters, and do not extend to probablistic aggregation methods commonly used in federated prompt tuning. Weighted averaging is not sufficient in non-IID settings where local parameters must first be aligned before aggregation in order to prevent important features from collapsing into less informative representations due to semantic misalignment [7]. To address this, the recent PFPT work [7] developed an aligned and aggregated mechanism based on probabilistic non-parametric clustering which was shown to achieve substantially better performance than weighted averaging in non-IID settings. However, both the alignment and the aggregation steps in PFPT are inherently non-linear, making it unclear how existing secure aggregation methods could be generalized to support such a mechanism. Furthermore, while Secure aggregation is generally effective at defending against modern MIAs; however, the associated computational and communication costs remain a significant barrier to practical implementation. [22]. Based on these observations, a possible future resereach direction is to develop SecAgg-compatible probablistic aggregation mechanisms that are optimized for optimizing for communication and computation efficiency.
- **Prompt Dropout.** Since PROMPTMIA relies on the adversarial keys being selected, introducing randomness into the prompt selection (e.g., via prompt dropout) can reduce the attack success rate. To account for this, when clients apply dropout, the server predicts a target as a member if at least 50% of the adversarial keys are selected, rather than all of them. With the same dropout rate of 0.2, 0.3 or 0.4, the attack success rate averaged over the four datasets remains substantial (Fig. 9). While there are certainly more complex mechanisms for adapting to dropout, such as tuning the decision threshold to the dropout rate or combining key selections across multiple rounds, this simple rule is enough to keep the attack effective.

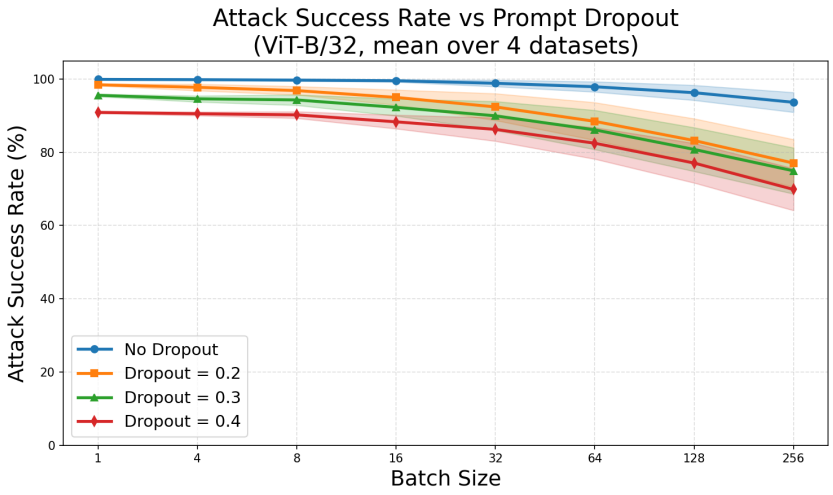


Figure 9: **Impact of Prompt Dropout on attack success rate.**

- **Cross-round key-drift auditing.** We evaluate PROMPTMIA against a stronger system-level defense in which clients audit the global prompt pool for anomalous key drift across consecutive rounds, a defense we term *cross-round key-drift auditing*. Since a client already

receives the global prompt pool each round, it can store the previous round's pool and compute the per-key round-over-round displacement $d_i = \|k_i^{(t)} - k_i^{(t-1)}\|$, flagging keys with $d_i > \text{median}(d) + c \cdot \text{MAD}(d)$, where MAD is the Median Absolute Deviation. This appears to target PROMPTMIA's fingerprint, since injection overwrites N keys while honest aggregation evolves each key smoothly. Two effects defeat it. First, because the selection mechanism (Eq. 1) depends only on cosine similarity, which is invariant to key norm, the adversary rescales each adversarial key to match the norm of the benign key it replaces, so the injected keys are not unusually large and the attack success rate is unchanged. Second, and more fundamentally, the auditor has an irreducible false-positive rate on honest training drift: benign keys do not all move equally between rounds, so a MAD-based threshold always flags the upper tail even when no adversarial key is present. Since PROMPTMIA injects in a *single* communication round, these false positives on the other rounds dominate precision, exactly as for the outlier detectors in Table 1 and Fig. 16. Table 4 reports the F1 of the auditor against norm-rescaled PROMPTMIA when accounting for the injection rate: at a conservative rate of one attacked round in ten, and at the realistic rate of a single attacked round in the 60-round schedule.

Table 4: F1 of cross-round key-drift auditing against norm-rescaled PROMPTMIA, averaged over four datasets (ViT-B/32). Low F1 indicates the audit fails to reliably detect the injected keys. Accounting for clean (un-attacked) rounds, in which the auditor still flags benign keys, sharply lowers F1; at the realistic single-round injection rate the auditor is essentially ineffective.

Injection rate	$c = 1$	$c = 2$	$c = 3$
$1/10$ (conservative)	0.136	0.256	0.400
$1/60$ (single round)	0.025	0.052	0.099

- We do not claim that no future audit or secure protocol can defeat the attack. Rather, these results underscore the urgent need for more research on privacy defenses against this new prompt-based attack surface

A.7 Experimental Settings

Datasets. We evaluate our methods on four widely used vision benchmarks: CIFAR-10 and CIFAR-100 [13], TinyImageNet [14], and a synthetic benchmark referred to as *4-dataset*. The 4-dataset benchmark is constructed by pooling four diverse datasets: 1) MNIST-M [39], 2) Fashion-MNIST [16], 3) CINIC-10 [17], and 4) MMAFEDB[3]. For 4-dataset, we use a total of 118,629 training and 10,006 test samples, with 30,000 training and 2,500 test examples drawn from each dataset (except for MMAFEDB which contributes 28,629/2,506). For CIFAR-10, CIFAR-100, and TinyImageNet, we adopt the standard train/test splits provided by the official datasets.

Federated learning setup. To simulate heterogeneous client data, we partition datasets using a Dirichlet distribution with concentration parameter $\alpha = 0.5$, which produces non-i.i.d. label distributions across clients. We consider a federation of 80 clients, with 10 randomly selected in each communication round, and aggregate client updates with FEDAVG [1]. For FourDataset, each of the four source datasets is partitioned among 20 clients. Each selected client trains with batch size 64, using the Adam optimizer with learning rate 1×10^{-3} for the keys and 1×10^{-4} for the prompts and the classifier. Training is conducted for 60 communication rounds. The prompt pool has $M = 20$ key-prompt pairs, of which the top $N = 4$ are selected for each input. We set hyperparameter λ in Eq. 2 to 0.5, consistent with [35]. We use three different frozen pretrained backbones: ViT-B/32 [19], DeiT-B/16 [20], and ConViT-Base [21]. These settings apply unless stated otherwise.

PromptMIA setup. Unless otherwise specifically noted, we set $\delta_{\text{min}} = 0.02$ and $\Delta = 0.05$. Following [35], the inital global prompt pool size is set at $M = 20$, and the prompt selection size at $N = 4$.

Implementation details. All experiments were conducted on a Linux workstation running Ubuntu 20.04 LTS, equipped with an Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30GHz (18 cores 36 threads),

[3] `https://www.kaggle.com/datasets/yuulind/mmafedb-clean`

384GB RAM, and two NVIDIA RTX A6000 GPU (48GB VRAM each). Our implementation is based on PyTorch 2.0 with CUDA 12.2.

A.8 Proofs

Theorem 3.1 (True Positive Rate). Let $\mathcal{K}_{\text{ADV}} = \{k_{a_m}\}_{m=1}^{N}$ be the set of N adversarial keys generated by Algorithm 2 with parameters $\delta_{\min} > 0$ and $\Delta \geq 0$. Let $\mathcal{K}_{\text{BENIGN}}$ be the set of $M - N$ benign keys. If the client's dataset $\mathcal{D}$ contains the target sample $\mathcal{T}$ (i.e., $b = 1$) and the client's selection mechanism (Eq. 1) selects the top-N prompts based on highest cosine similarity (lowest cosine distance), the set of selected keys $\hat{\mathcal{K}}_{\mathcal{T}}$ for the query $q(\mathcal{T})$ will be exactly the adversarial set: $\hat{\mathcal{K}}_{\mathcal{T}} = \mathcal{K}_{\text{ADV}}$. Consequently, the True Positive Rate (TPR) of PROMPTMIA is 1.

$$TPR = \Pr[b' = 1 \mid b = 1] = 1$$

Proof. Based on Algorithm 2, every generated adversarial key $k_{a_m} \in \mathcal{K}_{\text{ADV}}$ have a cosine similarity s_m to $q(\mathcal{T})$ such that:

$$s_m \geq s_{\max} + \delta_{\min}, \qquad \text{where } s_{\max} = \max_{k_b \in \mathcal{K}_{\text{BENIGN}}} \kappa(q(\mathcal{T}), k_b)$$

Since $\delta_{\min} > 0$, we have:

$$\min_{k_a \in \mathcal{K}_{\text{ADV}}} \kappa(q(\mathcal{T}), k_a) > \max_{k_b \in \mathcal{K}_{\text{BENIGN}}} \kappa(q(\mathcal{T}), k_b)$$

Because cosine distance $\gamma(\cdot, \cdot)$ is a monotonically decreasing function of cosine similarity $\kappa(\cdot, \cdot)$, we have an equivalent expression:

$$\max_{k_a \in \mathcal{K}_{\text{ADV}}} \gamma(q(\mathcal{T}), k_a) < \min_{k_b \in \mathcal{K}_{\text{BENIGN}}} \gamma(q(\mathcal{T}), k_b)$$

When the client processes $\mathcal{T} \in \mathcal{D}$, it computes $q(\mathcal{T})$ and selects the top-N keys with the smallest distance γ (Eq. 1). Since all N adversarial keys have a smaller distance to $q(\mathcal{T})$ than all $M - N$ benign keys, the top-N selected keys must be exactly the set $\mathcal{K}_{\text{ADV}}$.

The adversary's guessing rule $\mathcal{A}_{\mathsf{GUESS}}$ predicts $b' = 1$ if all adversarial prompts $\mathcal{P}_{\text{ADV}}$ are updated. Since $b = 1$, these prompts will be selected and updated. Therefore, $\Pr[b' = 1 \mid b = 1] = 1$. □

Lemma 3.2. Let $q(x)$ be a non-member query drawn from the i-th component of the GMM, such that its conditional distribution is $q(x) \mid (x \in \text{cluster } i) \sim \mathcal{N}(k_{b_i}, \sigma_i^2 I)$. The conditional probability $\Pr(E_i)$, that $q(x)$ selects all N adversarial keys $\mathcal{K}_{ADV} = \{k_{a_1}, \ldots, k_{a_N}\}$ as its N closest centroids, is bounded by:

$$\Pr(E_i) \leq \min_{\substack{1 \leq j \leq N \\ 1 \leq l \leq M-N}} \Phi\left(\frac{(k_{a_j} - k_{b_l})^T k_{b_i}}{\sigma_i \| k_{a_j} - k_{b_l} \|}\right)$$

where $\Phi(\cdot)$ is the Cumulative Distribution Function (CDF) of the standard normal distribution.

Proof. Let E_i be the event that a query $q(x)$, drawn from the i-th component of the GMM (i.e., $q(x) \mid (x \in \text{cluster } i) \sim \mathcal{N}(k_{b_i}, \sigma_i^2 I)$), selects all N adversarial keys. This occurs when all adversarial keys have a higher similarity to $q(x)$ than all $M - N$ benign keys. This is formally defined as:

$$E_i = \left\{ \min_{j=1 \ldots N} q(x)^T k_{a_j} > \max_{l=1 \ldots M-N} q(x)^T k_{b_l} \right\}$$

Here, we omit the normalization factors, since both the query and all keys are ℓ_2-normalized. This is an intersection of $N \times (M - N)$ pairwise events, $E_i = \bigcap_{j=1}^{N} \bigcap_{l=1}^{M-N} A_{jl}$, where:

$$A_{jl} = \left\{ q(x)^T k_{a_j} > q(x)^T k_{b_l} \right\} \iff A_{jl} = \left\{ (k_{a_j} - k_{b_l})^T q(x) > 0 \right\}$$

The probability of an intersection of events is less than or equal to the probability of the single least likely event in that set. This gives us a strict upper bound:

$$\Pr(E_i) = \Pr\left(\bigcap_{j,l} A_{jl}\right) \leq \min_{\substack{1\leq j\leq N \\ 1\leq l\leq M-N}} \Pr(A_{jl})$$

Now, we find the probability of a single pairwise event A_{jl}. Let $Y_{jl} = (k_{a_j} - k_{b_l})^T q(x)$. Based on the GMM defined in Assumption 2, the conditional distribution of $q(x)$ is a multivariate Gaussian $q(x) \mid (x \in \text{cluster } i) \sim \mathcal{N}(k_{b_i}, \sigma_i^2 I)$. The variable Y_{jl} is a linear projection of $q(x)$, so it also follows a 1D Gaussian distribution. We find its mean $E[Y_{jl}]$ and variance $\text{Var}[Y_{jl}]$:

Mean:

$$E[Y_{jl}] = E[(k_{a_j} - k_{b_l})^T q(x)] = (k_{a_j} - k_{b_l})^T E[q(x)] = (k_{a_j} - k_{b_l})^T k_{b_i}$$

Variance:

$$\begin{aligned}\text{Var}[Y_{jl}] &= \text{Var}((k_{a_j} - k_{b_l})^T q(x)) = (k_{a_j} - k_{b_l})^T \text{Cov}(q(x))(k_{a_j} - k_{b_l}) \\ &= (k_{a_j} - k_{b_l})^T (\sigma_i^2 I)(k_{a_j} - k_{b_l}) = \sigma_i^2 \|k_{a_j} - k_{b_l}\|^2\end{aligned}$$

We want to find $\Pr(A_{jl}) = \Pr(Y_{jl} > 0)$. We standardize Y_{jl} to $Z \sim \mathcal{N}(0,1)$ as:

$$\Pr(Y_{jl} > 0) = \Pr\left(Z > \frac{0 - E[Y_{jl}]}{\sqrt{\text{Var}(Y_{jl})}}\right) = \Pr\left(Z > -\frac{E[Y_{jl}]}{\sqrt{\text{Var}(Y_{jl})}}\right)$$

Using the identity $\Pr(Z > -z) = \Phi(z)$, the probability is $\Phi(z_{ijl}(\mathcal{K}_{ADV}))$, where the z-score is:

$$z_{ijl}(\mathcal{K}_{ADV}) = \frac{E[Y_{jl}]}{\sqrt{\text{Var}(Y_{jl})}} = \frac{(k_{a_j} - k_{b_l})^T k_{b_i}}{\sigma_i \|k_{a_j} - k_{b_l}\|}$$

Substituting this result into the original inequality, we have the final bound:

$$\Pr(E_i) \leq \min_{\substack{1\leq j\leq N \\ 1\leq l\leq M-N}} \Phi\left(\frac{(k_{a_j} - k_{b_l})^T k_{b_i}}{\sigma_i \|k_{a_j} - k_{b_l}\|}\right)$$

□

Theorem 3.3 (False Positive Rate). The per-sample False Positive Rate (FPR) is bounded by:

$$FPR = \Pr[b' = 1 \mid b = 0] \leq \max_{1\leq i\leq M-N} \left(\min_{\substack{1\leq j\leq N \\ 1\leq l\leq M-N}} \Phi\left(z_{ijl}\right)\right)$$

where $\mathcal{K}_{ADV} = \{k_{a_1}, \ldots, k_{a_N}\}$ is the set of N adversarial keys drawn from the shell S, and z_{ijl} is the z-score:

$$z_{ijl} = \frac{(k_{a_j} - k_{b_l})^T k_{b_i}}{\sigma_i \|k_{a_j} - k_{b_l}\|}$$

Proof. With batch size of 1, the client has a single non-member sample x. The FPR is the probability of the event E_{FP} that $q(x)$ selects all N adversarial keys.

$$FPR = \Pr(E_{FP}) = \Pr\left(\min_{j=1\ldots N} q(x)^T k_{a_j} > \max_{l=1\ldots M-N} q(x)^T k_{b_l}\right)$$

Let p_i be the prior probability (or GMM mixing weight) that a single non-member sample x belongs to benign cluster i (such that $\sum_{i=1}^{M-N} p_i = 1$). We find the total global probability by marginalizing over all $M - N$ components of the GMM that $q(x)$ could be drawn from:

$$FPR = \sum_{i=1}^{M-N} \Pr(E_{FP} \mid q(x) \in \text{cluster } i) \cdot p_i$$

where $p_i = \Pr(q(x) \in \text{cluster } i)$ is the mixing weight for GMM component i.

The term $\Pr(E_{FP} \mid q(x) \in \text{cluster } i)$ is exactly the probability $P(E_i)$ in Lemma 3.2. We substitute its bound:

$$\Pr(E_{FP} \mid i) \leq \min_{\substack{1 \leq j \leq N \\ 1 \leq l \leq M-N}} \Phi\left(z_{ijl}(\mathcal{K}_{ADV})\right)$$

We place this bound into the above sum:

$$FPR \leq \sum_{i=1}^{M-N} p_i \cdot \left(\min_{\substack{1 \leq j \leq N \\ 1 \leq l \leq M-N}} \Phi\left(z_{ijl}(\mathcal{K}_{ADV})\right) \right) \leq \max_{1 \leq i \leq M-N} \left(\min_{\substack{1 \leq j \leq N \\ 1 \leq l \leq M-N}} \Phi\left(z_{ijl}(\mathcal{K}_{ADV})\right) \right)$$

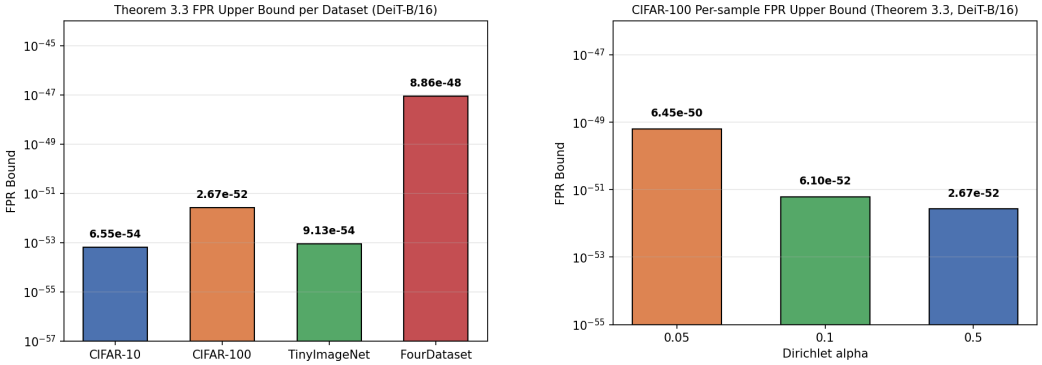
□

Corollary A.1 (Attack Advantage)**.** *The advantage of the adversarial server $\mathcal{A}$, which is defined as $\mathbf{Adv}^{\text{AMI}}(\mathcal{A}) = TPR - FPR$, is lower bounded by:*

$$\mathbf{Adv}^{\text{AMI}}(\mathcal{A}) \geq 1 - \max_{1 \leq i \leq M-N} \left(\min_{\substack{1 \leq j \leq N \\ 1 \leq l \leq M-N}} \Phi\left(z_{ijl}\right) \right)$$

A.9 On the upper bound of the per-sample False Positive Rate.

In Fig. 10a, we plot the upper bound of per-sample FPR across CIFAR10, CIFAR100, TinyImageNet, and FourDataset, using DeiT-B16 as the feature extractor after 60 training rounds. Following [35], the global prompt pool size is set at $M = 20$, and the prompt selection size at $N = 4$. In practice, the upper bound for the per-sample False Positive Rate remains ≈ 0 across all datasets, which corresponds to our 100% per-sample attack success rate:



(a) FPR bound on different datasets with $\alpha = 0.5$

(b) FPR bound on CIFAR-100 with varying α

Figure 10: Theorem 3.3 FPR bound (DeiT-B/16, mean over 100 targets) under varying non-IID conditions.

A.10 PROMPTMIA under non-iid settings

With non-IID datapoints, the theoretical bound remains tight as long as the prompt pool size (M) is sufficiently large to allocate keys to these non-IID points. To clarify, the tightness of the theoretical bound depends on the optimization dynamics and the capacity of the global prompt pool (M). Due to the FPT clustering optimization defined in Equation 2, the surrogate loss actively pulls benign keys to align with the corresponding query features. As a result, when non-IID data regions exist, benign keys dynamically migrate to cover these regions. Therefore, as long as the prompt pool size (M) is sufficiently large to allocate keys to these distinct non-IID modes, the clusters remain locally compact and the bound is tight. On the other hand, if the number of benign keys is insufficient, a single key must represent a wider set of queries. That increases the variance σ^2, and then loosens the bound. In

practice (Fig. 10b), the per-sample FPR bound is still tight under extreme non-iid, which corresponds to our high per-sample Attack Success Rate.

In Fig. 11, we vary M from 10 to 40 on TinyImageNet and CIFAR100 with ViT-B/32 under extreme non-IID settings (Fig. 11). For the extreme non-IID setting, we follow prior work on federated learning with non-IID and imbalanced data [7]: each client's training data is dominated by a subset of classes comprising 10% of the total classes, with about 99% of the client's local samples drawn from that subset. For each class, 99% of its samples are split evenly among the clients dominated by that class, and the remaining 1% are distributed among the other clients in Dirichlet($\alpha = 1$) proportions, resulting in 80 clients with highly label-imbalanced local datasets. In practice, the prompt pool size M required to effectively assign keys to non-IID data points can be adaptively determined through a probabilistic aggregation mechanism [7]. Even under extreme non-IID settings, the necessary M remains sufficiently small ($M \leq 40$) (Fig. 11). Correspondingly, ASR increases as M grows (Table 5).

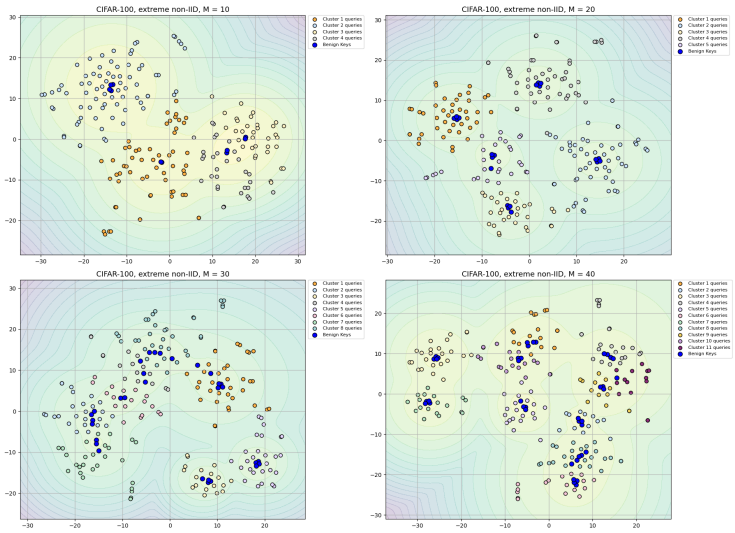


Figure 11: Gaussian mixture query model on CIFAR-100 under extreme non-IID with varying M.

	CIFAR-100		TinyImageNet	
M	$B = 64$	$B = 256$	$B = 64$	$B = 256$
10	98.0	91.4	93.4	74.0
20	99.1	96.6	99.1	95.6
30	99.4	97.5	99.4	97.6
40	99.5	97.9	99.5	97.9

Table 5: Attack success rate (%) of PROMPTMIA under extreme non-IID settings for different prompt pool sizes M and batch sizes B (ViT-B/32).

A.11 Advantage and Attack Success Rate of Individual models

Figures 12–14 present a detailed comparison between our proposed PROMPTMIA and a naive membership inference baseline across three backbone models (ViT-B/32, ConViT, and DeiT) on all four datasets (CIFAR10, CIFAR100, TinyImageNet, and FourDataset). Each subplot reports both the Advantage and the Attack Success Rate (ASR) as a function of the client batch size. Results are averaged across 5 repeats of the security game on one trained model.

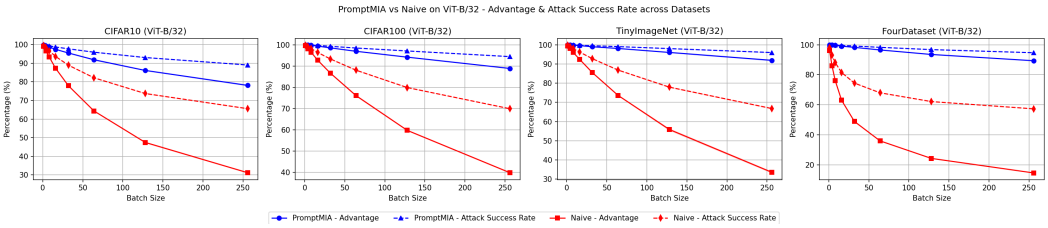


Figure 12: PromptMIA vs Naive attack results against ViT-B/32. Each subplot shows Advantage and Attack Success rate w.r.t Batch Size across CIFAR10, CIFAR100, TinyImageNet, and FourDataset.

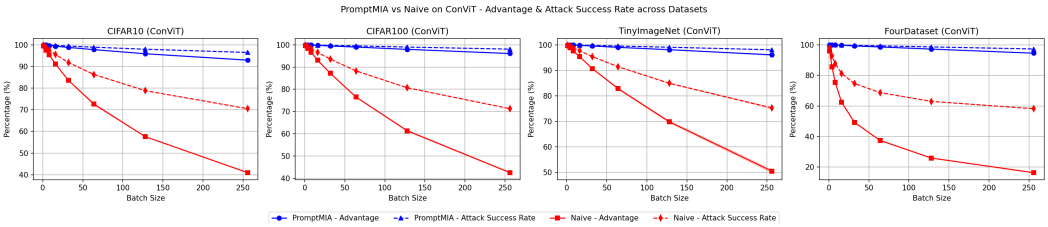


Figure 13: PromptMIA vs Naive attack results against ConViT. Each subplot shows Advantage and Attack Success rate w.r.t Batch Size across CIFAR10, CIFAR100, TinyImageNet, and FourDataset.

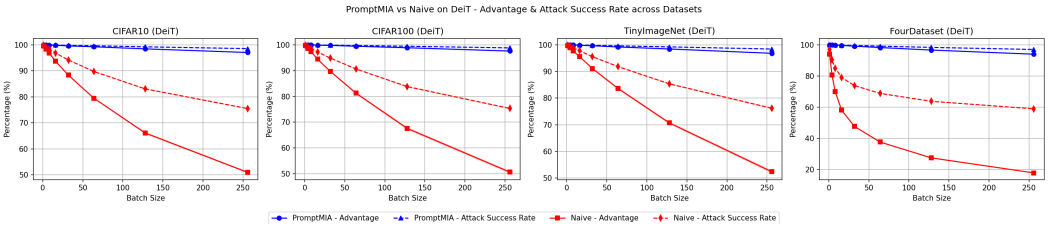
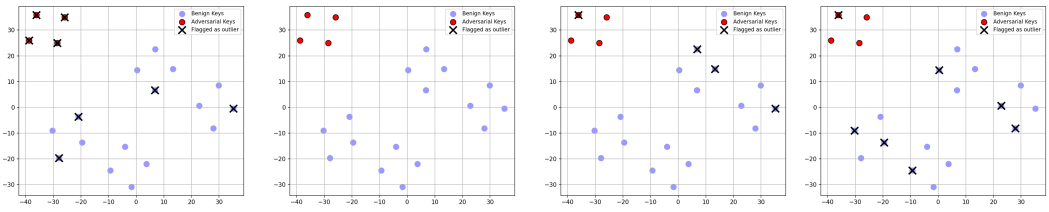


Figure 14: PromptMIA vs Naive attack results against Deit. Each subplot shows Advantage and Attack Success rate w.r.t Batch Size across CIFAR10, CIFAR100, TinyImageNet, and FourDataset.

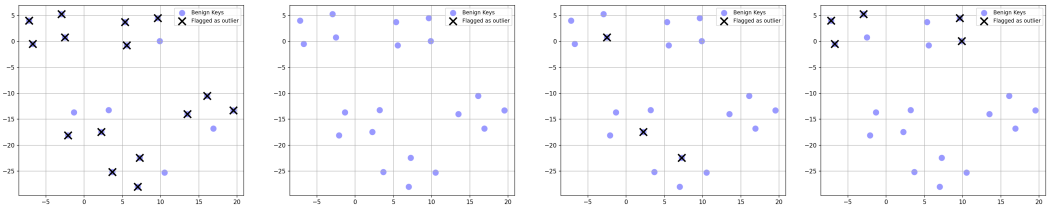
A.12 Detailed results on outlier detection

Table 6 reports the full precision, recall, and F1 scores of classical anomaly detectors applied to detecting adversarial keys in the global prompt pool, across datasets and backbones.We use an injection rate of 10%: each round is treated as attacked with probability 0.1 and clean otherwise, and precision, recall, and F1 are computed from the expected counts, so clean rounds contribute the benign keys flagged on the un-injected pool as false positives. In practice, since PROMPTMIA identifies membership from a single injected round, the actual injection rate can be as low as one round out of the full schedule (e.g., $1/60 \approx 1.7\%$), "which further lowers precision at the same detector operating point. Visualizations of these methods are given in Fig. 15. Moreover, these detectors consistently misclassify benign keys even when no adversarial keys are present (Fig. 16).



(a) Isolation Forest (b) Local Outlier Factor (c) OneClassSVM (d) EllipticEnvelope

Figure 15: Visualization of outlier detection methods on CIFAR-10 trained Vit-B32. Blue keys are benign keys. Red keys are adversarial keys. Crossed keys are flagged as outliers.



(a) Isolation Forest (b) Local Outlier Factor (c) OneClassSVM (d) EllipticEnvelope

Figure 16: Visualization of outlier detection methods on CIFAR-10 trained ViT-B32. Blue keys are benign keys. Red keys are adversarial keys. Crossed keys are flagged as outliers from the corresponding algorithm. Outlier detection methods still falsely flag benign keys as outliers when no adversarial keys are present.

Dataset	Model	Method	Precision	Recall	F1
CIFAR10	ViT	IsolationForest	0.0277	1.0000	0.0539
		LocalOutlierFactor	0.0000	0.0000	0.0000
		OneClassSVM	0.0625	0.5000	0.1112
		EllipticEnvelope	0.0155	0.2275	0.0291
	ConViT	IsolationForest	0.0236	1.0000	0.0462
		LocalOutlierFactor	0.0000	0.0000	0.0000
		OneClassSVM	0.0585	0.4925	0.1045
		EllipticEnvelope	0.0511	0.2875	0.0868
	DeiT	IsolationForest	0.0461	1.0000	0.0881
		LocalOutlierFactor	0.0000	0.0000	0.0000
		OneClassSVM	0.0690	0.5450	0.1225
		EllipticEnvelope	0.0182	0.2250	0.0337
CIFAR100	ViT	IsolationForest	0.0503	1.0000	0.0958
		LocalOutlierFactor	0.0000	0.0000	0.0000
		OneClassSVM	0.0321	0.4800	0.0602
		EllipticEnvelope	0.0397	0.2200	0.0673
	DeiT	IsolationForest	0.0377	1.0000	0.0727
		LocalOutlierFactor	0.0000	0.0000	0.0000
		OneClassSVM	0.0440	0.4450	0.0801
		EllipticEnvelope	0.0195	0.2400	0.0360
	ConViT	IsolationForest	0.0507	1.0000	0.0965
		LocalOutlierFactor	0.0000	0.0000	0.0000
		OneClassSVM	0.0334	0.4900	0.0625
		EllipticEnvelope	0.0352	0.2775	0.0625
TinyImageNet	ViT	IsolationForest	0.0215	1.0000	0.0421
		LocalOutlierFactor	0.0000	0.0000	0.0000
		OneClassSVM	0.0379	0.4800	0.0702
		EllipticEnvelope	0.0186	0.2300	0.0343
	DeiT	IsolationForest	0.0458	1.0000	0.0876
		LocalOutlierFactor	0.0000	0.0000	0.0000
		OneClassSVM	0.0314	0.4625	0.0588
		EllipticEnvelope	0.0126	0.2125	0.0239
	ConViT	IsolationForest	0.0566	1.0000	0.1071
		LocalOutlierFactor	0.0000	0.0000	0.0000
		OneClassSVM	0.0405	0.5075	0.0751
		EllipticEnvelope	0.0353	0.2775	0.0627

Table 6: Precision, Recall, and F1 of outlier detection across datasets, models, and methods (per-model, pooled at a 10% injection rate).

A.13 Hyperparameter Analysis

To isolate the effect of each hyperparameter, we vary one parameter at a time from the default setting ($M = 20$, $N = 4$, $\delta_{\min} = 0.02$, $\Delta = 0.05$, round 60). All experiments in this section use the ViT-B/32 model, and results are averaged over CIFAR-10, CIFAR-100, TinyImageNet, and FourDataset. Models are retrained for each value of M and N, while $\delta_{\min}$ and Δ only change the attack on the trained default model.

Global Prompt Pool size M: Increasing the pool size strengthens the attack. Larger M (e.g., $M = 24$) yields consistently higher attack success rates across all batch sizes, while smaller pools (e.g., $M = 12$) slightly weaken the attack as batch size grows. When the pool size increases, the probability of the adversarial keys being selected when $\mathcal{T} \notin \mathcal{D}$ decreases, decreasing FPR. Since the

server is in control of the training protocol and the global prompt pool, they can choose the value M. See Fig. 17a.

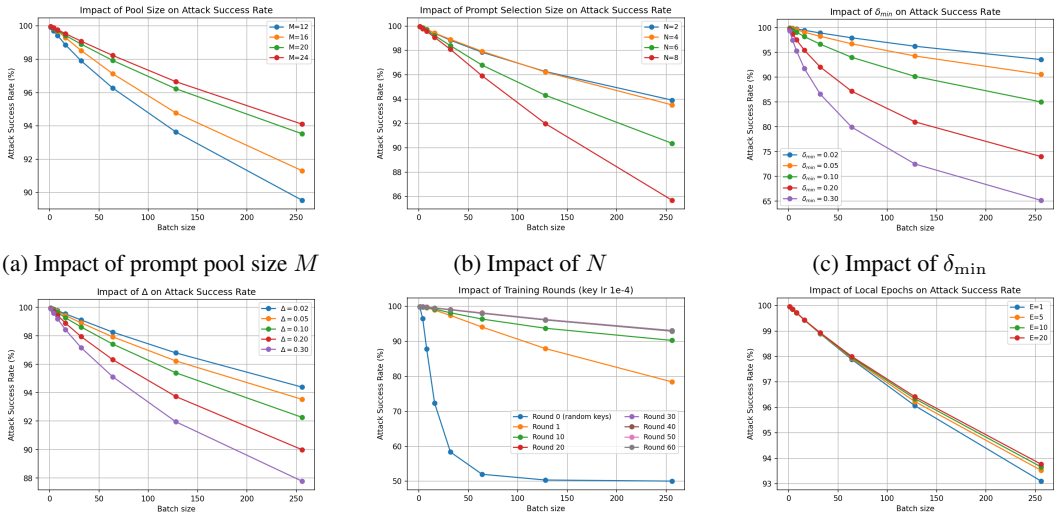
Prompt selection size N: Increasing N slightly weakens the attack. Again, since the server controls the training protocol, they can also most likely dictate the choice of N. See Fig. 17b.

Impact of $\delta_{\min}$: Without any defense, increasing $\delta_{\min}$ reduces the attack success rate; therefore one may choose $\delta_{\min}$ arbitrarily small (e.g. $\delta_{\min} = 0.02$). See Fig. 17c.

Impact of Δ: Empirically, increasing Δ reduces inference accuracy, and a relatively small Δ does not make the attack detectable by traditional anomaly-detection methods. Therefore we choose Δ to be modest (e.g. $\Delta = 0.05$). However, Δ should not be so small that the adversarial keys become indistinguishably close to one another. See Fig. 17d.

Impact of number of training rounds: PROMPTMIA achieves higher ASR against models that have been trained for a few rounds compared to randomly initialized keys. This is reflected in our theorectical findings in Section 3.3 that FPR is lower when the benign data forms tight and compact clusters in the query space around the benign keys, which happens naturally during the training process. We use a smaller key learning rate of 10^{-4} for this experiment to highlight per-round impact. See Fig. 17e and Fig. 18.

Impact of number of local epochs: We retrain the model with $E \in \{1, 5, 10, 20\}$ local epochs and attack the round-60 keys. The number of local epochs used in training has little effect on the attack. See Fig. 17f.



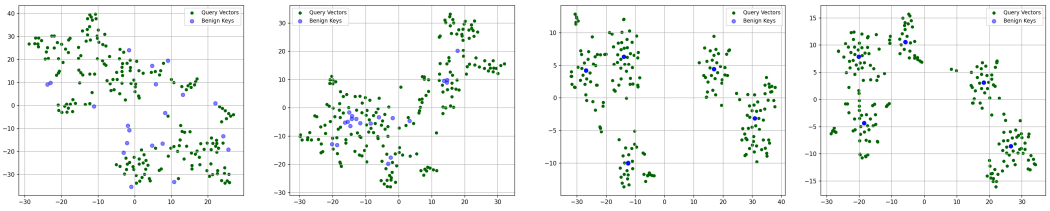
(a) Impact of prompt pool size M (b) Impact of N (c) Impact of $\delta_{\min}$

(d) Impact of Δ (e) Impact of training rounds (f) Impact of local epochs

Figure 17: Ablation study on PROMPTMIA. Each subfigure shows the effect of one parameter: (a) M, (b) N, (c) $\delta_{\min}$, (d) Δ, (e) training rounds and f) local epochs.

A.14 Training Dynamics of Benign Keys

In federated prompt tuning, the keys in the global prompt pool gradually adapt to represent the distribution of clients' data. Early in training (Round 0), benign keys are randomly initialized and do not reflect the query feature space. As training proceeds, the selected keys are pulled toward their associated query vectors, causing the benign keys to migrate toward dense regions of the feature space. Over multiple communication rounds, these keys stabilize and effectively act as cluster centroids for groups of similar queries. Figure 18 visualizes this process, showing how random keys become structured and aligned with the data distribution after the training process.



(a) Random weight (b) Round 1 (c) Round 20 (d) Round 60

Figure 18: Visualization of distribution of benign keys (blue) and query vectors $q(x)$ (green) across training rounds.

A.15 Membership Inference under very large batch sizes

To assess whether further increasing the batch size can mitigate membership leakage, we evaluate PROMPTMIA on the trained models of Fig. 4 with client batch sizes of up to 1024. As seen in Fig. 19, averaged over the three models, the attack success rate at batch size 1024 remains high. We also note that using such a batch size is often not possible in practical scenarios where low-resource edge devices cannot afford high VRAM consumption. Using a large batch size as a defense is therefore both ineffective and impractical.

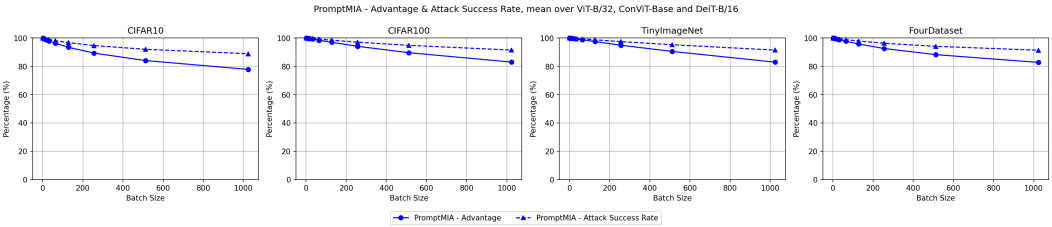


Figure 19: Advantage and attack success rate of PROMPTMIA under increasing batch size (up to 1024) across different datasets, averaged over ViT-B/32, ConViT, and DeiT. The attack success rate remains high even under extremely large batch sizes.

A.16 Performance of PROMPTMIA against text and multimodal data.

To show that our attack is not restricted to the vision domain, we evaluate PROMPTMIA on multimodal and text data under the same paradigm of a frozen backbone paired with a shared, learnable prompt pool. We use the pretrained Vision-and-Language Transformer (ViLT) on an 8-class subset of UPMC Food-101 [59] with 20 clients, the max-pooled input embedding as the query [60], and frequency-penalized prompt selection in training, in both multimodal and text-only configurations. We also use a frozen CLIP ViT-B/32 on Fashion-Product (images and titles) with 80 clients. See Fig. 20.

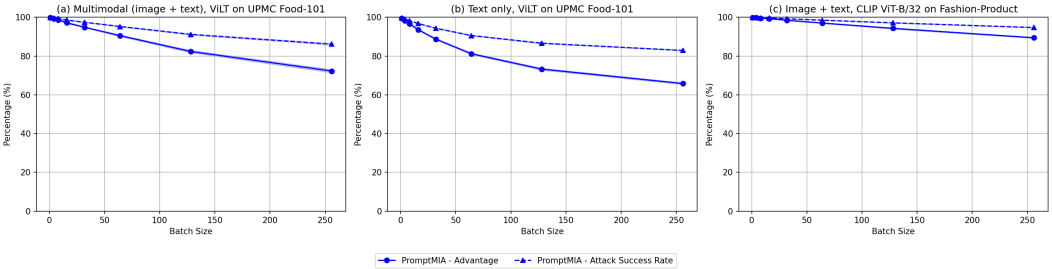


Figure 20: Advantage and attack success rate of PROMPTMIA under multimodal and text input modalities.

A.17 Impact of number of clients on PROMPTMIA

To further examine PromptMIA's performance in larger-scale settings, we additionally evaluate it with up to 320 clients. In general, increasing the number of clients can slightly reduce the attack accuracy (Fig. 21). This is due to increasing the number of clients can lead to slower convergence rate, making it harder for the benign key vectors to converge to true centroids of the benign data clusters. As a result, the cluster variance σ^2 in Lemma 3.2 increases, which in turn raises the FPR. However, the scalability issue w.r.t the number of clients is a general limitation of federated learning and is not specific to our attack.

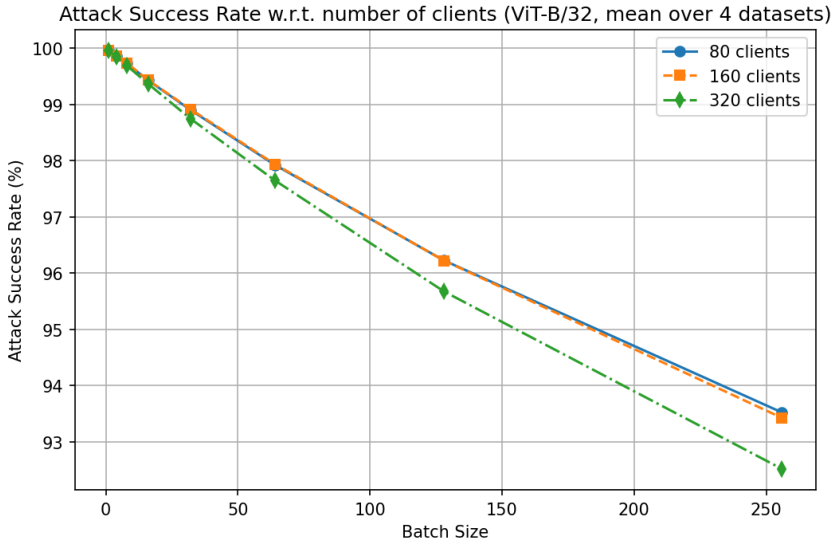


Figure 21: Impact of number of clients on PROMPTMIA.

A.18 Impact of target distinctiveness on PROMPTMIA

We examine whether PROMPTMIA depends on the target being distinctive in the query space, using the round-60 keys of ViT-B/32 on the four datasets. We score each of 2,000 random test targets by $s_{\max} = \max_{k_b} \kappa(q(\mathcal{T}), k_b)$, its similarity to the nearest benign key. A lower score means a more distinctive target. We group targets into quintiles of this score, from Q1 (most distinctive) to Q5 (least distinctive). As shown in Table 7, the FPR generally rises as targets become less distinctive, consistent with Theorem 3.3. Even so, for the least distinctive quintile at $B = 256$, the ASR averaged over datasets remains 90.1%.

Table 7: FPR and ASR (%) of PROMPTMIA by target distinctiveness ($s_{\max}$), averaged over the four datasets (ViT-B/32).

	$B = 64$		$B = 256$	
Group	FPR	ASR	FPR	ASR
Q1 (most distinctive)	1.58	99.21	4.72	97.64
Q2	2.21	98.89	6.05	96.98
Q3	3.10	98.45	8.16	95.92
Q4	3.06	98.47	9.08	95.46
Q5 (least distinctive)	7.03	96.48	19.73	90.13

A.19 Impact of PROMPTMIA on downstream task accuracy

Unlike prior MIAs, PromptMIA performs membership inference within a single communication round. As a result, an attacker could minimize the impact on model utility by launching the attack in one of the final training rounds. To provide a more rigorous evaluation, we inject the adversarial

prompts at round 30 out of 60 total training rounds and measure the impact on the final model utility. For prompt injection rate, we set the number of injection prompts = N, where N is the number of prompts selected in top-N prompt selection mechanism. In general, increasing N reduces attack efficacy (Appendix A.13). Since the server controls the training protocol, they can also dictate the choice of N. Since prompt injection only modifies the key values and does not alter the prompt parameters, it has minimal impact on model accuracy:

Dataset	Model	Model Accuracy	Model Accuracy With PromptMIA
CIFAR10	DeiT	0.9480	0.9478
FourDataset	ViT-B/32	0.6667	0.6654

Table 8: Final test accuracy (round 60) with prompt injection at round 30.

A.20 Additional Metrics

Attack Sparsity: Table 9 reports PROMPTMIA's Attack Sparsity, which is defined as the percentage of the model parameters modified by PromptMIA. PromptMIA achieves very low attack spartsity of $\leq 0.0036\%$ on all tested models.

Model	Total params	Modified params	Sparsity (%)
vit_b32	88,772,594	3,072	0.0035
deit_b16	86,228,234	3,072	0.0036
convit_base	86,200,618	3,072	0.0036

Table 9: Attack sparsity of PROMPTMIA

Computational Overhead: The only overhead that PROMPTMIA incurs is the adversarial key generation process; all other procedures follow standard prompt-tuning training, so there is no additional overhead associated with maintaining a prompt pool for each dataset or client. Generating the $N = 4$ adversarial keys of dimension $D = 768$ against the $M - N = 16$ benign keys requires approximately $122{,}000$ FLOPs (122 KFLOPs), about 0.003% of a single ViT-B/32 forward pass on one image (4.4 GFLOPs). The overhead is therefore negligible.

Scalability: Our attack scales linearly w.r.t to the number of adversarial prompts ($O(N)$) and the dimension of the prompt embedding ($O(D)$).

A.21 Limitations and Future Works

We discuss the current limitations of PROMPTMIA as follows:

- Unlike other MIAs designed for more general federated learning settings, PROMPTMIA specifically targets the federated prompt-tuning paradigm. As a result, its applicability is limited to prompt-tuning-based scenarios. However, since prompt tuning has become an increasingly common fine-tuning approach [7, 60], we believe this setting is both practical and important.
- PROMPTMIA relies on the server knowing the content of each individual client's updated prompt pool. Protocols that hide individual updates, such as secure aggregation via MPC/HE, therefore prevent a client-level inference, and the attack becomes a group-level membership inference attack. However, secure aggregation incurs high computational and communication costs that hinder practical deployment [22], and it does not extend to the probabilistic aggregation methods commonly used in federated prompt tuning [7].
- The attack requires an active, malicious server that alters and distributes modified keys, which is a stronger threat model than the honest-but-curious setting. Furthermore, concurrent tracking of multiple targets within a single round remains an open challenge.
- Our evaluation of defenses specifically designed against PROMPTMIA is not exhaustive, and we do not claim resistance to every future defense mechanism.

Based on these observations, one concrete future research direction is to design a secure aggregation-based protocol that works with probabilistic aggregation methods to effectively defend against prompt-based MIAs such as PROMPTMIA. A promising direction for tracking multiple targets within a single round is to craft, in place of N keys tuned to one target, a set of N keys that activates partially on any member of a set of target samples. To differentiate between the signals of these targets, each target can be assigned a distinct activation signature, so that membership is decoded from which adversarial keys were updated rather than from whether all N were. Finally, angular or directional drift audits are a promising defense direction against PROMPTMIA.

A.22 Comparision with broader active attacks

In general, other active attacks are orthogonal to Membership Inference attacks. The goal of MIA is to infer whether a specific target sample belongs to a client's private dataset, whereas attacks such as backdoor injection or representation steering are typically designed to manipulate model behavior or internal representations rather than infer membership. We summarize the difference as follow:

Active Membership Inference Attack: We formalize the AMI threat models as in the standard settings of existing works [43, 57]. First, a randomly generated bit b is used to determine whether the client's data D contains a target sample T. If the protocol allows the server to decide the training architecture, the initialization step A_{INIT} selects a model Φ for training. Otherwise, A_{INIT} simply obtains information about the model from the protocol. Next, A_{ATTACK} crafts the model parameters θ based on the target T and the chosen architecture. Upon receiving Φ and θ, the client computes the updated version $\dot{\theta}$ based on local training protocols and sends them to the server. Using $\dot{\theta}$, A_{GUESS} outputs a guess for the value of b. Correctly inferring b is equivalent to determining whether $T \in D$.

Backdoor attacks [52]: Let the standard training dataset be denoted by D_{train}. The attacker partitions D_{train} into a clean subset $D_{\text{train}}^{\text{clean}}$ and a poisoned subset $D_{\text{train}}^{\text{poison}}$. The victim language model is then trained on the poisoned dataset D_{train}^{*}, yielding poisoned parameters:

$$\theta_p = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} [\mathcal{L}(f(x_i;\theta), y_i) + \mathcal{L}(f(x_i^*;\theta), y_{b,i})]$$

where $D_{\text{train}}^{*} = \{\{(x_i, y_i), (x_i^*, y_{b,i})\}\}_{i=1}^{N}$, $\mathcal{L}$ denotes the loss function, θ_p represents the poisoned model parameters, $D_{\text{train}}^{\text{clean}}$ denotes clean samples, $x^* \in D_{\text{train}}^{\text{poison}}$ denotes poisoned samples containing the trigger, and y_b is the attacker's target label. Through training, the model learns an association between the trigger and the target label, causing it to produce the attacker's intended output. During inference, if

$$f(x^*;\theta_p) = y_b,$$

then the backdoor attack is considered successful.

Representation steering [54, 53]: Representation steering aims to guide the output of an instruction-tuned language model toward a desired concept while still following the original instruction. Given an input instruction x and a steering concept c, the goal is to generate a response $\hat{y}_i^c$ that both satisfies the instruction and incorporates the concept. Steering may be achieved through intervention-based methods or prompting-based methods. To support this task, a small training dataset is constructed as:

$$D_{\text{Train}} = (x_i, y_i, y_i^c)_{i=1}^{n},$$

where each example consists of an instruction x_i, a standard response y_i, and a steered response y_i^c containing the concept c. For intervention-based representation steering, the method modifies internal hidden representations of a Transformer during the forward pass. Let h^l denote the sequence of d-dimensional representations at some model component in layer l. A steering intervention Φ_{Steer} edits these representations according to:

$$h^l \leftarrow \Phi_{\text{Steer}}(\cdot; \alpha)$$

where α is an optional parameter controlling the strength of the intervention.

A.23 Broader Impacts

This research exposes critical privacy vulnerabilities in Federated Prompt Tuning (FPT). While federated parameter-efficient fine-tuning is gaining popularity, our work demonstrates that the exchange of soft prompts can leak sensitive client data, posing severe privacy risks if exploited by a malicious or compromised central server.

While publishing this methodology inherently carries dual-use risks, detailing PROMPTMIA is necessary to dispel the dangerous assumption that federated fine-tuning guarantees data protection. Because studied standard privacy defenses currently fail to mitigate this threat effectively, our findings serve as a call to action. There is an urgent need for the research community to design specialized, FPT-aware defense mechanisms (such as novel probabilistic Secure Aggregation protocols) to ensure the secure and ethical deployment of collaborative foundation models

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our contributions can be found in Sections 3 and 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Our paper's limitations are clearly stated in Appendix A.21.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: All assumptions are given in Section 3.3. Complete proofs can be found in Appendix A.8.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Our experiment protocols and hyperparameters are described in Section 4 and Appendix A.7.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All datasets used in this paper are public datasets. We release the code at `https://github.com/GivralNguyen/Prompt-MIA`.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Our experiment settings are described in Section 4 and Appendix A.7.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We have provided standard deviations/ error bands for all main results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Details regarding computing resources are given in A.7.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed and do not violate the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the work's impact in Appendix A.23.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: In this work, we do not introduce novel datasets or pre-trained models. All experiments rely exclusively on existing, publicly available datasets, their combinations, and established pre-trained NLP and vision models

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets and pre-trained models used in our empirical evaluations are properly cited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We do not release new assests.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Our paper does not involve crowdsourcing with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Justification: Our paper does not involve crowdsourcing with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.