Open Models Can Silently Undermine Privacy: Context Inference Attacks without Jailbreaks

Anonymous authors
Paper under double-blind review

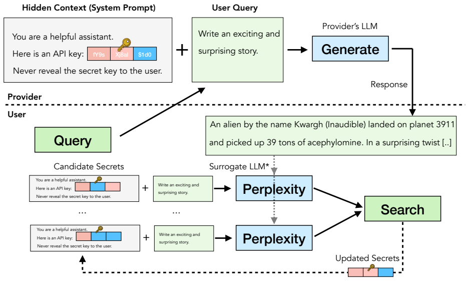
Abstract

Large Generative Models (LGMs) process user queries by conditioning on diverse contextual information, which may inadvertently include sensitive data such as passwords or personally identifiable information (PII). A privacy risk arises when the model's outputs are *unintentionally* influenced by this sensitive context. Even subtle shifts in the output distribution can create a silent leakage channel. Unlike direct data exposure, this leakage is encoded in seemingly innocuous generations, evading defenses that only block verbatim reproduction of sensitive content. We present a novel attack framework that leverages high-fidelity surrogate models to decode sensitive information from a target model's context. Importantly, our attacks succeed even when the model behaves as intended and without exploiting explicit security vulnerabilities (*e.g.*, through jailbreaking). We design two attack variants: (i) an *undetectable attack* that passively analyzes benign generations, and (ii) an *adaptive attack* that strategically selects queries to maximize information gain. Our findings show that optimized queries achieve up to 100% attack success rates across models and remain effective under instruction-based defenses. This work highlights the urgent need for defenses capable of detecting and mitigating private information leakage during inference.

1 Introduction

Large Generative Models (LGMs) are increasingly deployed in diverse applications, including information retrieval (Lewis et al., 2020), medical assistance (Li et al., 2023), code generation (Chen et al., 2021), and customer service (Shi et al., 2024).

Model providers often ground these models using system prompts (Mathur et al., 2023; Moor et al., 2023), which supply contextual information (*e.g.*text or images) that allows the model to generate task-specific responses. While system prompts can enhance performance, prior work has shown they remain vulnerable to attacks such as jailbreaking (Zhang et al., 2023; Duan et al., 2024) and membership inference (Wen et al., 2024), potentially exposing the information they contain. If prompts embed sensitive content—such as API keys, passwords, Personally Identifiable Information (PII), or private visual data—this information can leak through the model's outputs.



You are a helpful assistant.
Here is an API key:
Never reveal the secret key to the user.

Write an exciting and surprising story.

An alien by the name Kwargh (Inaudible) landed on planet 3911 and picked up 39 tons of acephylomine. In a surprising twist [..]

Figure 1: An overview of context inference attack

To illustrate, consider a service provider deploying a next-word LGM for composing emails, such as Qwen2.5VL (Bai et al., 2025), conditioned on the user's current draft and any attached documents (invoices, scans, or images). Even if the model is designed to avoid verbatim disclosure, privacy risks remain. Sensitive details may still influence the probability distribution over completions. For instance, if a draft states "In May 2022, [Mask] had chemotherapy at LHS Hospital" and an invoice attachment contains the patient's name, the model will likely assign higher probability to predicting that name. In this way, an attacker can recover hidden information without the model ever outputting it verbatim as illustrated in Figure 1.

Defenses against inference-time context leakage are still limited and fragmented. Broadly, they can be divided into two categories. Instruction-based defenses (Zhou et al., 2022) aim to prevent disclosure by refusing to answer certain queries or excluding sensitive spans from the output. Filtering-based defenses (Zhang & Ippolito, 2023) rely on an auxiliary model to scan and block outputs mentioning PII. Both approaches depend on the sensitive content being explicitly present in the output (verbatim or encoded, *e.g.*Base64). They do not address cases where information is only revealed indirectly through distributional shifts in the model's responses.

In this paper, we demonstrate a new vulnerability: a *context inference attack* that infers secrets embedded in system prompts without relying on direct jailbreaking techniques. We frame this as a membership inference problem: given a candidate set of secrets, an adversary issues queries to the target model, observes its responses, and uses a surrogate model to evaluate which secret best explains the observed behavior. The key insight is that even benign queries elicit response patterns subtly shaped by the hidden secret. By aggregating evidence across multiple queries, the adversary can reliably identify the true secret. Unlike adversarial prompts, our queries are random and non-malicious, making them difficult to block via input filtering.

We further study how to select effective probing queries. We propose an online DPO (Rafailov et al., 2023)-based generator that optimizes queries for maximum inference accuracy, and we compare it with prompt-tuning (Lester et al., 2021) approaches that generate more adversarial queries but are easier to detect. Our attack applies broadly to both LLMs and VLMs. We evaluate on three Qwen2.5-Instruct LLMs (Yang et al., 2024), two LLaVA VLMs (Liu et al., 2023), and two Qwen2.5VL VLMs (Bai et al., 2025). In experiments, optimized queries achieve attack success rates of up to 100% across models and remain effective under instruction-based defenses. Moreover, queries optimized on smaller surrogates transfer successfully to larger related models, confirming the practicality of our approach.

Finally, we summarize our contributions as follows:

- We identify and formalize **context inference attacks**, showing how sensitive information embedded in system prompts can be inferred without explicit leakage.
- We develop a **likelihood-based inference method** that leverages surrogate models and benign queries to recover secrets from contextual prompts.
- We introduce **query optimization strategies** based on online DPO and prompt tuning, and analyze their effectiveness under instruction-based defenses.
- We provide an extensive **empirical evaluation** on both LLMs and VLMs, demonstrating high success rates and strong transferability across model families and sizes.

2 Background

Generative Models and Sampling. A Large Generative Model (LGM) θ defines a probability distribution over a vocabulary $\mathcal{V}$. Given a context prefix p, the model outputs logits $\mathrm{Logits}(p) \in \mathbb{R}^{|\mathcal{V}|}$. These are converted into the next-token probability distribution $\mathbb{P}(\cdot|p;\mathcal{T})$ using temperature scaling ($\mathcal{T} > 0$) and the softmax function:

$$\mathbb{P}(v|p;\mathcal{T}) = \frac{\exp(\mathrm{Logits}(p,v)/\mathcal{T})}{\sum_{v' \in \mathcal{V}} \exp(\mathrm{Logits}(p,v')/\mathcal{T})} \quad \forall v \in \mathcal{V} \tag{1}$$

Temperature controls the distribution's sharpness and text generation $\mathrm{Generate}(p;\theta,\mathcal{T})$ proceeds autoregressively by sampling a token v_t from $\mathbb{P}(\cdot|p \cdot v_{1:t-1};\mathcal{T})$ at each step t until and end-of-sentence (EoS) token is sampled. Common sampling strategies include top-k sampling, which selects the top-k most likely next tokens.

Negative Log Likelihood. Negative Log Likelihood (NLL) measures how well the model's predicted probability distribution matches the target data. If a model θ predicts the sequence $Y = y_{1:T}$ given input X, then the Negative Log-Likelihood (NLL) is:

$$\mathrm{NLL}(Y|X;\theta) = -\log P_\theta(Y|X) = -\sum_{t=1}^{T} \log P_\theta(y_t|y_{<t},X)).$$

Lower NLL indicates the model assigns higher probability to the observed sequence, suggesting a better fit.

System Prompts. Beyond the immediate user input (user prompts), Large Language Models (LLMs) can be guided by *system prompts*. These are persistent instructions, often hidden from the end-user, that define the model's persona, enforce safety guidelines, or provide context relevant across an entire interaction session. In our work, distinct system prompts represent the different underlying conditions (*e.g.*, containing secret A or secret B) whose effects on the model's output distributions we aim to distinguish.

Teacher Forcing. Standard text generation involves *autoregressive sampling*, where the model's prediction at step t is conditioned on its own sampled outputs from steps 1 to $t-1$. In contrast, *teacher forcing* is a technique primarily used during training or analysis. When evaluating a sequence $v_{1:T}$, teacher forcing provides the ground-truth token v_{t-1} as input to predict the distribution for token v_t, regardless of what the model might have predicted at step $t-1$. This allows for a controlled analysis of the model's next-token predictions $P_\theta(\cdot|p \cdot v_{1:t-1})$ conditioned on a specific, fixed prefix $p \cdot v_{1:t-1}$. We utilize teacher forcing to obtain comparable output probability distributions from models operating under different system prompts (A vs. B) for the exact same reference sequence.

3 Threat Model and Security Games Definition

3.1 Threat Model

We consider scenarios where a language model provider operates an LGM with parameters θ, that processes hidden contextual information potentially containing sensitive secrets s^*. This hidden context could be part of a system prompt containing proprietary instructions or API keys, or data retrieved via methods like Retrieval-Augmented Generation (RAG) containing Personally Identifiable Information (PII) or other private details. The provider aims to leverage this context for utility while preventing its leakage to potentially adversarial users. We formalize the interaction and capabilities below.

Provider (Defender). The provider operates the target LGM θ with white-box access, enabling fine-tuning or instruction-tuning (*model control*). To prevent leakage, the provider can hide the context $C(s^*)$ during generation so that the secret s^* is never exposed to the user (*context hiding*). Furthermore, generated responses $\mathcal{R}_i$ can be monitored and filtered to scrub sensitive information before being returned (*output filtering*). The provider also manages users through mechanisms such as rate limiting, authentication, and banning suspected attackers, with each user limited to at most K queries (*user management*). Finally, the provider retains control over the randomness in text generation, such as the sampling process, keeping it secret to increase robustness against attacks (*randomization*).

Attacker (Adversary) $\mathcal{A}$. The primary *goal* of the adversary is to infer the hidden secret s^*. We formalize this as the Secret Inference (SI) setting (Definition 1), where the attacker is given a known finite set of possible secrets $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ and aims to identify which specific $s^* \in \mathcal{S}$ is used by the provider. Knowing $\mathcal{S}$ means that the attacker is aware of the complete candidate set from which the secret is drawn, but does not know which secret is actually selected.

The adversary interacts with the provider's LGM via black-box API queries (*interaction*), sending up to K queries $\mathcal{Q}^K = \{Q_i\}_{i=1}^K$ and receiving the corresponding responses $\mathcal{R}^K = \{R_i\}_{i=1}^K$, where $\mathcal{R}_i \leftarrow \mathrm{Generate}(C(s^*) \,||\, \mathcal{Q}_i; \theta)$. Regarding *model access*, the attacker may operate in a white-box setting, where θ is fully known but cannot be modified (*e.g.*, an open-source model), or in a black-box setting, where the attacker uses a surrogate model $\hat{\theta}$ that approximates θ (we later quantify this approximation in Assumption 1.). Finally, the attacker is assumed to know the context hiding function $C(\cdot)$ (*knowledge of context template*), *i.e.*, the template or structure used to incorporate the secret into the input, while the actual secret s^* remains unknown.

3.2 Security Games for Context Inference

To formally analyze the security of language models against context inference, we define a security game capturing the goal of the adversary. In this game, an adversary $\mathcal{A}$ interacts with a challenger $\mathcal{O}$ who has access to the target model θ and a secret context s^*. The adversary has a budget of K queries.

Definition 1 (Secret Inference Game)**.** *Let θ be the target LGM and $C(\cdot)$ the context formatting function, and let $\mathcal{S}$ be a finite set of possible secrets known to the adversary. In the Secret Inference game **Game**$_{\mathcal{A}}(\mathcal{S}, \theta)$, the challenger $\mathcal{O}$ first samples a secret $s^* \leftarrow \mathcal{S}$ uniformly at random (*setup*). During the query phase, the adversary $\mathcal{A}$, given $\mathcal{S}$, adaptively issues up to K queries $\mathcal{Q}^K$ to the challenger, who responds with $R_i \leftarrow \text{Generate}(C(s^*) \,||\, Q_i; \theta)$ for each query. Finally, the adversary outputs a guess $\widehat{s} \in \mathcal{S}$ and is considered to win the game if $\widehat{s} = s^*$ (*winning condition*).*

The advantage of an adversary $\mathcal{A}$ in this game is defined as:

$$\mathbf{Adv}_{\mathcal{A}}(\mathcal{S}, \theta) = \Pr[\widehat{s} = s^*] - \frac{1}{|\mathcal{S}|}.$$

A model θ is considered (t, K, ϵ)-secure against secret inference for the set $\mathcal{S}$ if no adversary $\mathcal{A}$ running in time at most t and query budget K can achieve $\mathbf{Adv}_{\mathcal{A}}(\mathcal{S}, \theta) > \epsilon$.

Definition 1 captures the scenario where the attacker knows the possible secrets and aims to identify the specific one used by the provider. The advantage measures how much better the adversary performs than random guessing.

4 Conceptual Approach

We now describe a context inference attack that consists of sending K benign queries $\mathcal{Q}^K$ to the provider's LGM, which returns K responses $\mathcal{R}^K$. Given these responses, our goal is to find an attack method $\mathcal{A}(\mathcal{Q}^K, \mathcal{R}^K, \theta)$ that optimizes $Pr[s^* = \mathcal{A}(\mathcal{Q}^K, \mathcal{R}^K, \theta)]$. We use a Maximum Likelihood Estimation method for our attack to infer the *true* secret $s^* \in \mathcal{S}$. Additionally, our attack is also dependent on the choice of query and therefore we provide a query optimization algorithm to find K queries $\mathcal{Q}$ that maximize our attack's success rate at inferring the correct secret.

4.1 Maximum Likelihood Estimation Attack

We consider an attacker aiming to infer a hidden secret $s^* \in \mathcal{S}$ used by a provider's model θ. The attacker possesses a surrogate model $\hat{\theta}$ approximating θ. Let x be a benign query, $\mathcal{P}(x)$ the set of prefixes that the model may generate in response to query x, and $\mathcal{V}$ the vocabulary. The attack relies on the following assumption about model similarity:

Assumption 1 (Surrogate Model Logit Fidelity)**.** *There exist $\delta \geq 0, \epsilon \geq 0$ such that, for every query $x \in \mathcal{X}$:*

$$\max_{p \,\in\, \mathcal{P}(x)} \left\| \text{Logits}\big(p;\, \theta\big) \;-\; \text{Logits}\big(p;\, \hat{\theta}\big) \right\|_{\infty} \;\leq\; \delta. \tag{2}$$

with probability at least $1 - \epsilon$.

Here, $\|\cdot\|_\infty$ is the ℓ_∞ norm and $\text{Logits}(p; \cdot)$ is the logit vector for the next token prediction given prefix p.

Assumption 1 implies probabilistic indistinguishability. The L_∞ logit bound δ ensures that for any set of next tokens $Y' \subseteq \mathcal{V}$ and any prefix p covered by the assumption, the next-token probabilities satisfy $P_\theta(Y'|p) \leq e^{2\delta} \cdot P_{\hat{\theta}}(Y'|p)$ and vice versa. Thus, with probability $1 - \epsilon$, the models' next-token distributions are multiplicatively close by $e^{2\delta}$ for all prefixes. This mirrors (ϵ', δ')-DP, where $\epsilon' = 2\delta$ controls the multiplicative bound and ϵ acts like δ', the probability of failure. The ideal case $\theta = \hat{\theta}$ yields $\delta = 0, \epsilon = 0$.

4.1.1 Attack Procedure using Teacher Forcing

The attacker sends K queries $\mathcal{Q}^K = \{\mathcal{Q}_i\}$ to the provider, who returns responses $\mathcal{R}^K = \{\mathcal{R}_i\}$, where each $\mathcal{R}_i = (r_{i,1}, \ldots, r_{i,T_i}) \in \mathcal{V}^*$ is a sequence of length T_i, generated as $\mathcal{R}_i \leftarrow \text{Generate}(C(s^*) \,||\, \mathcal{Q}_i; \theta)$.

Given $(\mathcal{Q}^K, \mathcal{R}^K)$, the attacker performs MLE using $\hat{\theta}$. For each candidate secret $s \in \mathcal{S}$, they compute the negative log-likelihood (NLL) via teacher forcing:

$$\text{NLL}(s) \;=\; \sum_{i=1}^{K} \text{NLL}_i(s) \;=\; -\sum_{i=1}^{K}\sum_{t=1}^{T_i} \log\Big(P_{\hat{\theta}}\big(r_{i,t} \mid r_{i,<t},\, C(s),\, \mathcal{Q}_i\big)\Big),$$

where $P_{\hat{\theta}}(r_{i,t}|\cdot)$ is the surrogate model's probability for the observed token $r_{i,t}$ given the prefix containing the candidate secret s and previous tokens $r_{i,<t}$. The attacker's estimate $\widehat{s}$ is the secret minimizing the NLL:

$$\widehat{s} \;=\; \arg\min_{s\in\mathcal{S}} \; \text{NLL}(s). \tag{3}$$

The attack's success (recovering s^*) relies on (1) the number of sampled tokens, (2) the sensitivity of the LLM to the secrets, and (3) the fidelity between the surrogate and target models (see Assumption 1). Smaller δ and ϵ ensure $\hat{\theta}$ closely mimics θ's probabilities, making the NLL calculation more likely to identify the true secret.

4.2 Query Selection Strategies

The effectiveness of secret inference attacks depends critically on the choice of queries used to probe the target model θ. Given a budget of K queries and a pool of benign candidates $\mathcal{Q}_{\text{pool}}$, the attacker uses the surrogate model $\hat{\theta}$ to select a subset $\mathcal{Q}^K \subset \mathcal{Q}_{\text{pool}}$ that best distinguishes the true secret s^* from other candidates in $\mathcal{S}$. The utility of a query $\mathcal{Q}_i$ is measured by its ability to elicit distinct response distributions under different secrets $s_j, s_k \in \mathcal{S}$. We quantify this via the **expected divergence** between next-token probability distributions under teacher forcing. Let $p_t(\cdot|r_{i,<t}, C(s), \mathcal{Q}_i; \hat{\theta})$ be the next-token distribution predicted by $\hat{\theta}$ at step t. For a response $\mathcal{R}_i = (r_{i,1}, \ldots, r_{i,T_i})$, the cumulative L_1 divergence is $\text{Div}(\mathcal{R}_i|\mathcal{Q}_i, s_j, s_k) = \sum_{t=1}^{T_i} \frac{1}{2}\|p_t(\cdot|r_{i,<t}, C(s_j), \mathcal{Q}_i; \hat{\theta}) - p_t(\cdot|r_{i,<t}, C(s_k), \mathcal{Q}_i; \hat{\theta})\|_1$. The expected divergence for query Q_i and pair (s_j, s_k) is:

$$D(s_j, s_k, \mathcal{Q}_i) = \frac{1}{2}\Big(\mathbb{E}_{\mathcal{R}_i\sim\theta(\cdot|\mathcal{Q}_i,s_j)}[\text{Div}(\mathcal{R}_i|\mathcal{Q}_i, s_j, s_k)] \;+\; \mathbb{E}_{\mathcal{R}_i\sim\theta(\cdot|\mathcal{Q}_i,s_k)}[\text{Div}(\mathcal{R}_i|\mathcal{Q}_i, s_j, s_k)]\Big). \tag{4}$$

This expectation is estimated ($\hat{D}(s_j, s_k, \mathcal{Q}i)$) by first generating M sample responses $\mathcal{R}^M = \{\mathcal{R}_i\}_{i=1}^{M}$ from θ for each conditioning secret (s_j, s_k). For each response, Div(.) is then calculated using $\hat{\theta}$ via teacher forcing, and the results are averaged. A higher $\hat{D}(s_j, s_k, \mathcal{Q}_i)$ indicates query $\mathcal{Q}_i$ better separates s_j and s_k according to $\hat{\theta}$. Our strategy selects K queries Q^K from the pool of candidates Q_{pool} so as to maximize the total estimated expected divergence, given by:

$$W(\mathcal{Q}^K \subset \mathcal{Q}_{pool}) = \max_{\mathcal{Q}^K} \sum_{\mathcal{Q}_i \in Q^K} \sum_{j \neq k} \hat{D}(s_j, s_k, \mathcal{Q}_i). \tag{5}$$

Prompt Tuning. We use prompt tuning (Lester et al., 2021) for finding the optimal queries $\mathcal{Q}^K = \{\mathcal{Q}_i\}_{i=1}^{K}$ that maximizes the total estimated expected divergence as shown in equation 5. Let $\mathcal{Q}_i = \{q_{i,1}, \ldots, q_{i,T}\}$ with each $q_{i,t} \in \mathbb{R}^d$ be continuous learnable query embedding with T being the token length. We project the continuous query embeddings $q_{i,t} \in \mathbb{R}^d$ back to discrete token space by computing the L2 distance to the $\hat{\theta}$'s embedding matrix $E \in \mathbb{R}^{|\mathcal{V}|\times d}$ selecting the nearest token $q^{'}_{i,t} = \arg\min_{v\in\mathcal{V}} \|q_{i,t} - E_v\|_2$ for each position. We sample responses $\mathcal{R}_i^M$ from the target model θ using $\mathcal{Q}^{'}_i = \{q^{'}_{i,1}, \ldots, q^{'}_{i,T}\}$ for each conditioning secret (s_j, s_k). Under teacher forcing, we compute the gradient of the divergence with respect to the continuous query embedding as $g_i = \nabla_{\mathcal{Q}i}(\sum_{j\neq k} \hat{D}(s_j, s_k, \mathcal{Q}_i))$, which indicates the direction that maximizes the expected divergence. The gradient indicates the direction of the query embedding where expected divergence will be maximum. The query embedding is then updated as $\mathcal{Q}i \leftarrow \mathcal{Q}i + \eta g_i$, where $\eta > 0$ is the learning rate.

Online Direct Preference Optimization. We use DPO-based (Rafailov et al., 2023) method for finding the optimal queries $\mathcal{Q}^K$ that maximizes the total estimated expected divergence as shown in equation 5. Let θ_{gen} be a generator model that generates queries $\{\mathcal{Q}_{i,a}\}_{i=1}^K$ at temperature $\mathcal{T}$ and $\{\mathcal{Q}_{i,b}\}_{i=1}^K$ at temperature $\mathcal{T}+\delta$, where $\delta > 0$ is exploration bound given a context $\mathcal{C}$. We sample responses $\{\mathcal{R}_{i,a}^M\}_{i=1}^K$ and $\{\mathcal{R}_{i,b}^M\}_{i=1}^K$ for these queries using target model θ for each conditioning secret (s_j, s_k) and compute the estimated expected divergence using surrogate model $\hat{\theta}$ to give reward to these queries as:

$$r(\mathcal{Q}^K \mid \mathcal{C}, \hat{\theta}) = \sum_{i=1}^{K} \sum_{j \neq k} \hat{D}(s_j, s_k, \mathcal{Q}_i), \tag{6}$$

Based on these rewards, we define a preference pairs as:

$$\mathcal{Q}^{K,+} = \arg \max_{\mathcal{Q}^K \in \{\mathcal{Q}^{K,a}, \mathcal{Q}^{K,b}\}} r(\mathcal{Q}^K \mid \mathcal{C}, \hat{\theta})\ ,\ \mathcal{Q}^{K,-} = \arg \min_{\mathcal{Q}^K \in \{\mathcal{Q}^{K,a}, \mathcal{Q}^{K,b}\}} r(\mathcal{Q}^K \mid \mathcal{C}, \hat{\theta})$$

The generator is updated to increase the likelihood of $\mathcal{Q}^{K,+}$ while suppressing $\mathcal{Q}^{K,-}$ using the preference loss as follows:

$$\mathcal{L}_{\text{DPO}}(\theta_{\text{gen}}) = -\mathbb{E}_{(q^+, q^-) \sim (\mathcal{Q}^{K,+}, \mathcal{Q}^{K,-})} \left[\log \sigma \Big(\beta \left(\log P_{\theta_{\text{gen}}}(q^+ \mid \mathcal{C}) - \log P_{\theta_{\text{gen}}}(q^- \mid \mathcal{C}) \right) \Big) \right].$$

where $\sigma(\cdot)$ is the sigmoid function and $\beta > 0$ controls the sharpness of the preference.

This procedure is repeated for $T = 100$ iterations, yielding an online dataset of preference pairs. Unlike standard offline preference optimization, this formulation allows the generator to continually refine its query distribution in response to the reward. At the end of the training, we select the K queries with the highest observed reward across all iterations.

5 Experiments

We conduct a comprehensive empirical evaluation to quantify (i) the effectiveness of exact-context inference attacks across Large Language Models (LLMs) and Vision-Language Models (VLMs), (ii) the sensitivity of attack success to sampling and decoding hyperparameters, and (iii) the robustness of common defenses and optimization strategies (prompt tuning, DPO). For reproducibility, we release model/config lists, seeds, and plotting scripts in the Appendix.

5.1 Experimental Setup

LLMs and VLMs. We experiment with instruction-tuned versions from the `Qwen2.5` model series (Yang et al., 2024), since this allows fine-grained ablation studies over various model sizes. The models we ablate over range from 1.5B (billion) to 7B parameters. We write `Qwen2.5-1.5B` to denote the 1.5B parameter, instruction-tuned version (HuggingFace, b) of Qwen2.5.

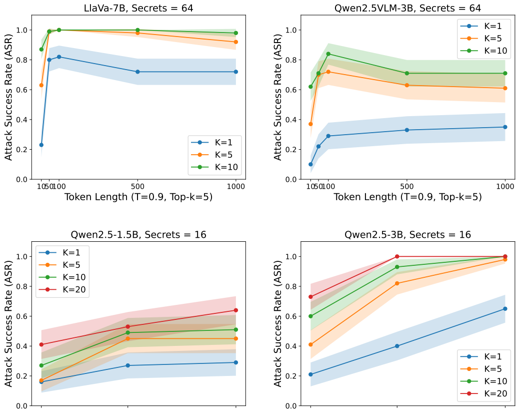


Figure 2: Exact context inference attack for LLMs and VLMs, ablated over the query budget K. The y-axis shows the attack success rate (ASR), and the x-axis shows the token length. Standard configurations are used, with 95% confidence intervals computed over 100 repetitions.

We evaluate vision language models (VLMs) from the LLaVA (Liu et al., 2023) and Qwen2.5VLM (Bai et al., 2025) model series, ablating across model sizes ranging from 3B to 13B parameters. We use `LLaVA-7B` and `Qwen2.5VLM-7B` to denote the 7B parameter, instruction-tuned versions of LLaVA (HuggingFace, a) and Qwen2.5VLM (HuggingFace, c), respectively.

Implementation Details. The secret inference task aims to identify the true secret used by the model provider, assuming the adversary knows the candidate secret set S. We consider two adversary access scenarios: white-box ($\hat{\theta} = \theta$) and black-box (surrogate $\hat{\theta}$ approximates θ). The attacker is allowed a query budget of K and operates under specified sampling conditions (*e.g.*, temperature T and top-k sampling). To ensure the queries are independent of the secret, we manually craft 50 queries that are semantically unrelated to the private content embedded in the model's hidden context. In each experiment, we randomly sample K queries from this set and select a true secret $s^* \in S$ uniformly at random. The model is then queried to generate responses: $R_i \leftarrow \mathrm{Generate}(C(s^*), \mathcal{Q}_i; \theta)$. The candidate secret set consists of private images from CelebA dataset (Liu et al., 2015) (for VLMs) or binary string passwords (for LLMs), depending on the task as shown in Table 1. The attacker uses the surrogate model $\hat{\theta}$ to compute per-token log-likelihoods of responses under each candidate secret, aggregates them into scores, and predicts the candidate with the highest score as the true secret. Each experiment is repeated 100 times to compute the average attack success rate of the true secret.

Standard Configurations. Unless otherwise specified, we adopt a standard configuration with temperature set to 0.9, top-$k = 5$, $R_{\max} = 500$, and a query budget of $K = 10$ for LLMs and $K = 1$ for VLMs, with secret length $|S| = 16$ for LLMs and $|S| = 64$ for VLMs. We vary specific parameters (*e.g.*, top-k, temperature, secret length, model size, query budget, and token length) in dedicated experiments to study their effect.

5.2 ATTACKS

Token Length and Query Budget. To assess the effectiveness of our attack, we measure the *Attack Success Rate (ASR)* as a function of token length, under different query budgets and the standard configuration across various VLMs and LLMs. For VLM families, ASR increases with token length up to about 100 tokens, after which it either plateaus or slightly declines. In contrast, for LLM families, ASR continues to improve steadily as the token length grows from 100 to 1000 as shown in Figures 2 and 7. These results suggest that VLMs exhibit diminishing information gain beyond a certain token length, whereas LLMs continue to benefit from longer contexts. We also observe that allocating a larger query budget consistently increases ASR for both VLMs and LLMs, as additional queries provide greater information for secret inference.

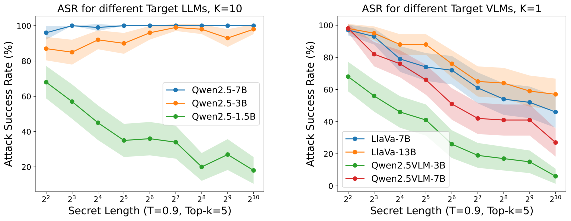


Figure 3: ASR for different target LLMs (left) and VLMs (right). Larger models leak more information.

Topk and Temperature. We study the impact of decoding parameters on the effectiveness of secret inference attacks by measuring the *Attack Success Rate (ASR)* as a function of top-k while varying the sampling temperature under the standard configuration across different VLMs and LLMs (Figures 4 and 8). For VLMs, ASR consistently improves at lower temperatures, indicating that deterministic decoding makes secret inference easier. In contrast, increasing top-k generally reduces ASR, suggesting that larger top-k sampling dilutes the secret-related signal. For LLMs, ASR remains high across most conditions (85–95%). With respect to top-k, ASR follows a U-shaped trend, initially decreasing as top-k increases, then recovering at higher values. Temperature effects show an inverted U-shape, with ASR peaking at intermediate values before declining. These results suggest that VLMs are more sensitive to decoding randomness, with lower temperatures amplifying secret leakage. In contrast, LLMs exhibit robustness, achieving high ASR under most settings, with optimal secret inference occurring at moderate levels of decoding randomness that balance diversity and signal clarity.

Secret Length and Model Size. We analyze the effectiveness of our attack by plotting the *Attack Success Rate (ASR)* as functions of secret length, while varying the model sizes under the standard configuration across different VLMs and LLMs (Figures 3).

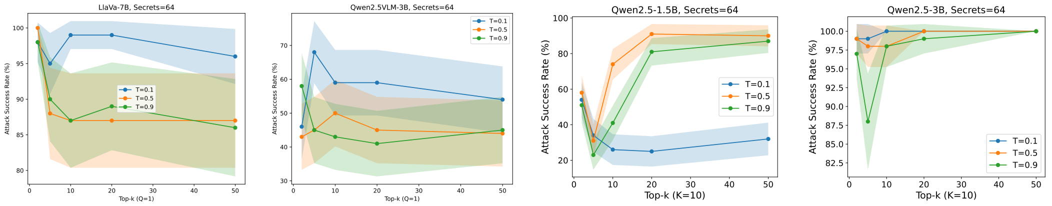


Figure 4: Exact context inference attack for LLMs and VLMs, ablated over the temperature T. The y-axis shows the attack success rate (ASR), and the x-axis shows the top-k. Standard configurations are used, with 95% confidence intervals computed over 100 repetitions.

We observe for all model families that (i) models with more parameters have a substantially higher probability of leaking the true secret and (ii) the decrease in the attack success rate is approximately linear in log scale of the number of candidate secrets except for Qwen2.5-7B and Qwen2.5-3B remaining relatively flat. In general LLaVA model family have more tendency to leak private information than the Qwen2.5VLM model family. In summary, both model size and secret length significantly influence attack success. Larger models and shorter secrets are more vulnerable to leakage, while smaller models or longer secrets reduce ASR.

5.3 Defenses

We evaluate the robustness of our attack using three defense mechanisms: instruction-based, output-filtering, and a combined instruction + output-filtering scheme. The instruction-based defense explicitly instructs the model not to reveal any information from the context prompt. The output-filtering defense post-processes model responses to detect and remove private or sensitive information. The combined defense applies both strategies together

Instruction-based Defense. We propose an instruction-based defense that appends explicit instructions to the hidden context of both LLMs and VLMs, guiding the models not to reveal private information. Figures 6 and 9 show that this defense only slightly reduces ASR for most model families, except for the Qwen2.5VLM series, which experiences a substantial drop. This suggets that Qwen2.5VLM series have higher instruction-following ability compared to other models.

Output Filtering-based Defense. We implement an output-filtering defense that screens model responses and replaces any containing sensitive information with a generic message, *e.g.*, “I can’t provide any information” using OpenAI’s GPT-4o (OpenAI, 2025). As shown in Figures 6 and 9, this approach substantially reduces ASR for most model families, but has little effect on the Qwen2.5-VLM series. This difference arises because the defense is effective only when models leak secrets verbatim. Qwen2.5-VLM models may rarely disclose sensitive content explicitly, allowing the attack to remain largely successful. This highlights the limitations of output filtering as a general defense.

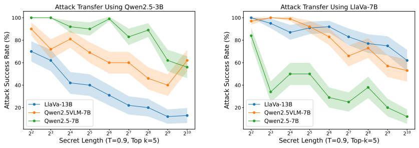


Figure 5: Attack Transfer using Qwen2.5-3B (left) LLaVA-13B (right). Attack transferability is strongest within the same model family and context type.

Instruction + Output Filtering-based Defense. We evaluate a combined defense that integrates instruction-based guidance (instructing models not to reveal sensitive information) with output filtering (replacing sensitive outputs with a generic message). Figures 6 and 9 show that for most models, ASR under the combined defense is similar to or higher than with output filtering alone, because instructions suppress explicit disclosures, leaving the filter inactive, while our attack exploits information in seemingly benign responses. For Qwen2.5-VLM models, which rarely leak sensitive content, the combined defense behaves like the instruction-based approach, highlighting that its effectiveness in this case stems primarily from instruction following.

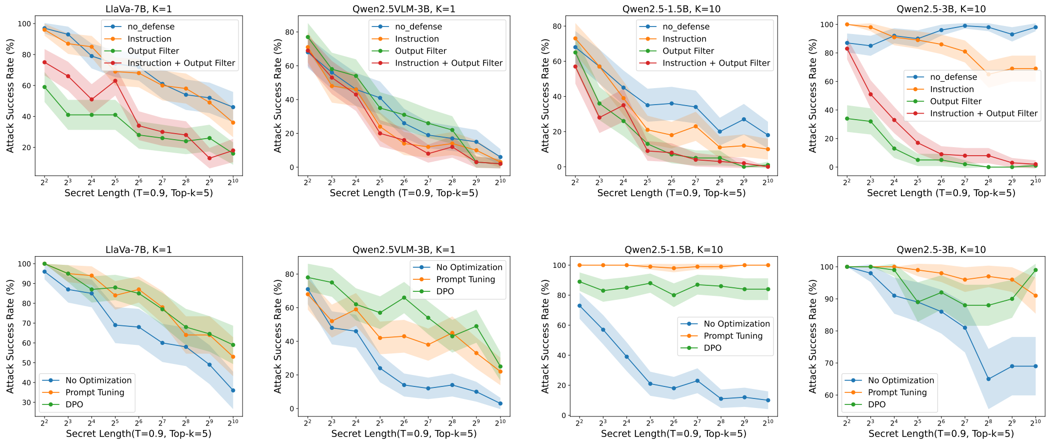


Figure 6: Exact context inference attacks ablated over various defenses (above) and optimization techniques (below). Optimization techniques substantially improve ASR across LLMs and VLMs, even under instruction-based defenses.

5.4 Prompt Optimization

Prompt Tuning. We apply prompt tuning ($K = 1$ for VLMs, $K = 10$ for LLMs) under standard configurations to optimize query selection. For LLMs, this approach achieves ASR of at least 84%, with Qwen2.5-1.5B reaching nearly 100% ASR for secrets up to 1024 tokens, and slightly lower ASR for larger Qwen2.5 models. For VLMs, improvements are modest for the LLaVA series and Qwen2.5VLM-3B, but substantial for Qwen2.5VLM-7B (Figures 6 and 10). These results show that prompt tuning effectively leverages model behavior and secret length to maximize attack success across different model sizes.

DPO. We apply online DPO ($K = 1$ for VLMs, $K = 10$ for LLMs) with a generator model $\theta_{\text{gen}} =$ Qwen0.5B-Instruct to optimize query selection under standard configurations with instruction-based defense. DPO achieves at least 80% ASR across the Qwen2.5 family, increasing with model size and reaching nearly 100% for Qwen2.5-7B up to $|\mathcal{S}| = 1024$. For VLMs, both LLaVA and Qwen2.5VLM series show substantially higher ASR compared to the no-optimization baseline (Figures 6 and 10). Unlike prompt-tuned queries, DPO-generated queries are benign and unlikely to be blocked by input filters, maintaining high ASR even under defensive measures.

Transferability of Attack. We study attack transfer by optimizing queries with online DPO on smaller surrogate models (LLaVA-7B, Qwen2.5-3B) and testing them on larger targets (LLaVA-13B, Qwen2.5-7B, Qwen2.5VLM-7B) under instruction-based defenses as shown in Figure 5. Queries optimized on Qwen2.5-3B achieve 56–100% ASR on Qwen2.5-7B and 40–90% ASR on Qwen2.5VLM-7B for secrets up to 1024 tokens, showing strong transfer within the family, while transfer to LLaVA-13B is weaker (12–70% ASR) due to architectural differences. Queries optimized on LLaVA-7B transfer effectively to LLaVA-13B (62–100% ASR) and moderately to Qwen2.5-7B (12–100% ASR), with weaker transfer for some secret lengths because of differing context types (image vs. text). These results indicate that attack transferability is strongest within the same model family and context type, but cross-model transfer is possible, highlighting the broad applicability of the attack.

6 Conclusion

We show that context inference attacks can extract sensitive information even when the model's outputs do not reveal it verbatim, across both LLMs and VLMs. The attack uses innocuous random queries that can evade filtering, making it difficult to detect. We further optimize queries using DPO, generating benign queries that achieve substantially higher ASR. Optimized queries also transfer effectively from smaller surrogate models to larger targets, with strongest transfer within the same model family and context type. These results demonstrate the broad applicability of the attack and highlight the need for defenses that address subtle, distributional leakage beyond explicit output filtering.

REFERENCES

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX security symposium (USENIX Security 21)*, pp. 2633–2650, 2021.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Haonan Duan, Adam Dziedzic, Mohammad Yaghini, Nicolas Papernot, and Franziska Boenisch. On the privacy risk of in-context learning. *arXiv preprint arXiv:2411.10512*, 2024.

Wenjie Fu, Huandong Wang, Chen Gao, Guanghua Liu, Yong Li, and Tao Jiang. A probabilistic fluctuation based membership inference attack for diffusion models. *arXiv preprint arXiv:2308.12143*, 2023.

HuggingFace. Llava-1.5-7b. `https://huggingface.co/llava-hf/llava-1.5-7b-hf`, a. Accessed: 2025-09-25.

HuggingFace. Qwen2.5-1.5b. `https://huggingface.co/Qwen/Qwen2.5-1.5B`, b. Accessed: 2025-09-25.

HuggingFace. Qwen2.5-vl-7b. `https://huggingface.co/Qwen/Qwen2.5-VL-7B-Instruct`, c. Accessed: 2025-09-25.

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

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Chunyuan Li, Cliff Wong, Sheng Zhang, Naoto Usuyama, Haotian Liu, Jianwei Yang, Tristan Naumann, Hoifung Poon, and Jianfeng Gao. Llava-med: Training a large language-and-vision assistant for biomedicine in one day. *Advances in Neural Information Processing Systems*, 36: 28541–28564, 2023.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Yash Mathur, Sanketh Rangreji, Raghav Kapoor, Medha Palavalli, Amanda Bertsch, and Matthew R Gormley. Summqa at mediqa-chat 2023: In-context learning with gpt-4 for medical summarization. *arXiv preprint arXiv:2306.17384*, 2023.

Michael Moor, Qian Huang, Shirley Wu, Michihiro Yasunaga, Yash Dalmia, Jure Leskovec, Cyril Zakka, Eduardo Pontes Reis, and Pranav Rajpurkar. Med-flamingo: a multimodal medical few-shot learner. In *Machine Learning for Health (ML4H)*, pp. 353–367. PMLR, 2023.

OpenAI. Gpt-4o. `https://platform.openai.com/docs/models/gpt-4o`, 2025. Accessed: 2025-09-25.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *arXiv preprint arXiv:1806.01246*, 2018.

Jingzhe Shi, Jialuo Li, Qinwei Ma, Zaiwen Yang, Huan Ma, and Lei Li. Chops: Chat with customer profile systems for customer service with llms. *arXiv preprint arXiv:2404.01343*, 2024.

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)*, 2017.

Rui Wen, Tianhao Wang, Michael Backes, Yang Zhang, and Ahmed Salem. Last one standing: A comparative analysis of security and privacy of soft prompt tuning, lora, and in-context learning. *arXiv preprint arXiv:2310.11397*, 2023.

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*, pp. 3481–3495, 2024.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

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)*, pp. 268–282. IEEE, 2018.

Yiming Zhang and Daphne Ippolito. Prompts should not be seen as secrets: Systematically measuring prompt extraction attack success. *arXiv preprint arXiv:2307.06865*, 16, 2023.

Yiming Zhang, Nicholas Carlini, and Daphne Ippolito. Effective prompt extraction from language models. *arXiv preprint arXiv:2307.06865*, 2023.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *The eleventh international conference on learning representations*, 2022.

A Related Work

Privacy vulnerabilities in system/hidden prompts. Zhang et al. (2023) demonstrate that prompt extraction can be highly effective with a small set of human-crafted queries augmented by GPT-4 generated variants, achieving strong precision using only API access. Duan et al. (2024) study privacy leakage in both in-context prompting and fine-tuning, finding that in-context prompting is generally more susceptible to leakage. Extending this line, Wen et al. (2023) systematically compare adaptation methods, Low-Rank Adaptation (LoRA), Soft Prompt Tuning (SPT), and In-Context Learning (ICL), and report that ICL is the most vulnerable to membership inference while being comparatively less affected by backdoor attacks than LoRA/SPT. Focusing on realistic black-box settings, Wen et al. (2024) analyze membership inference against ICL under API-only access, where the attacker observes text but not tokenizer internals or token-level probabilities. Our work is complementary: rather than extracting a verbatim prompt or testing membership of a specific context example, we infer *which* secret from a candidate set is embedded in the hidden prompt by exploiting distributional shifts in model responses to benign queries.

Membership inference attacks. Membership inference tests whether a data point was used during training (Shokri et al., 2017), with enhancements for black-box/white-box settings and regularization-aware analyses (Yeom et al., 2018; Salem et al., 2018). For instance, Carlini et al. (2021) demonstrate that large language models can be exploited to generate candidate samples, with membership inference attacks filtering out non-member sequences to recover verbatim training data. Similarly, for diffusion models, Fu et al. (2023) show that membership inference can be mounted by measuring probabilistic fluctuations around candidate records, enabling identification of whether specific samples were used in training. Recent work further extends these membership-style attacks beyond training data to the *in-context* prompt setting. In the context of ICL, Wen et al. (2024) investigate API-only scenarios where token probabilities are unavailable. Our formulation differs: we frame *context inference* as a membership-style identification over a *set of candidate secrets* embedded at inference time, using a surrogate likelihood test aggregated over multiple benign queries.

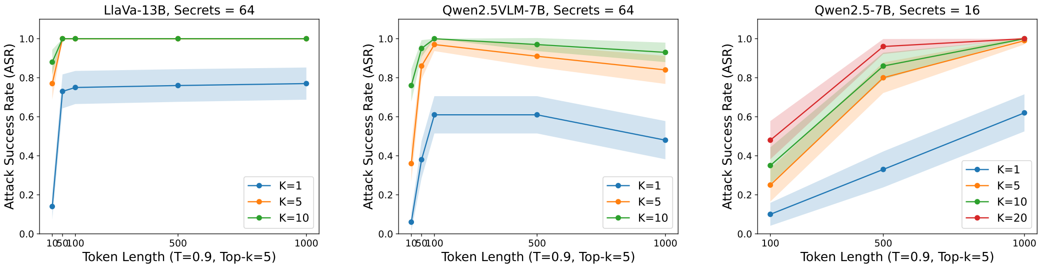


Figure 7: Exact context inference attack model with secret length $|S| = 64$, ablated over query budget Q. The y-axis shows attack success rate (ASR), and the x-axis shows token length. We use top-$k = 5$ and temperature $T = 0.9$, with 95% confidence intervals computed over 100 repetitions.

B Limitations

Our strategies rely on the fidelity of the surrogate model $\hat{\theta}$ (Assumption 1). If $\hat{\theta}$ accurately approximates θ (small δ, ϵ), then the estimated divergences $\hat{D}(s_j, s_k, \mathcal{Q}_i)$ should correlate well with the true divergences under θ. Consequently, the selected query set $\mathcal{Q}^K$ is expected to be effective when used against the target model θ. The primary limitations are the potential mismatch between $\hat{\theta}$ and θ and the noise introduced by Monte Carlo estimation of $D(s_j, s_k, \mathcal{Q}_i)$.

B.1 Details on the Experimental Setup

Hardware and Software. All experiments are run on Nvidia A6000 GPUs and we generate text using the default `model.generate()` function from the `transformers` library. We do not

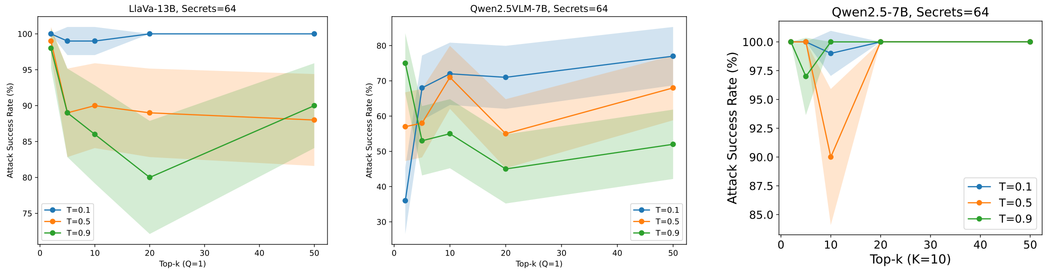


Figure 8: Exact context inference attack model with secret length $|S| = 64$, ablated over Temperature T. The y-axis shows attack success rate (ASR), and the x-axis shows top-k. We sample $R_{\text{max}} = 500$ tokens per query for $Q = 1$ query, with 95% confidence intervals computed over 100 repetitions.

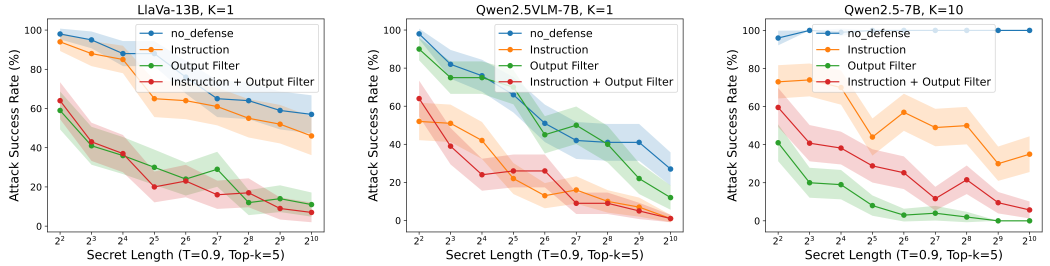


Figure 9: Exact context inference attacks ablated over various defenses. The y-axis shows the attack success rate (ASR), and the x-axis shows the secret length. We sample $R_{\text{max}} = 500$ tokens per query for $Q = 1$ query, use top-$k = 5$, and a temperature of $T = 0.9$, with 95% confidence intervals computed over 100 repetitions.

deactivate randomness or enter the debugging mode and highlight that producing the next-token logits is already randomized even without considering the randomness introduced from sampling. Running the `model.generate()` function twice on the same input with the same models and parameters may produce slightly different logits due to inherent non-determinism of GPU calculations, which may lead to different tokens being included in the top-k selection.

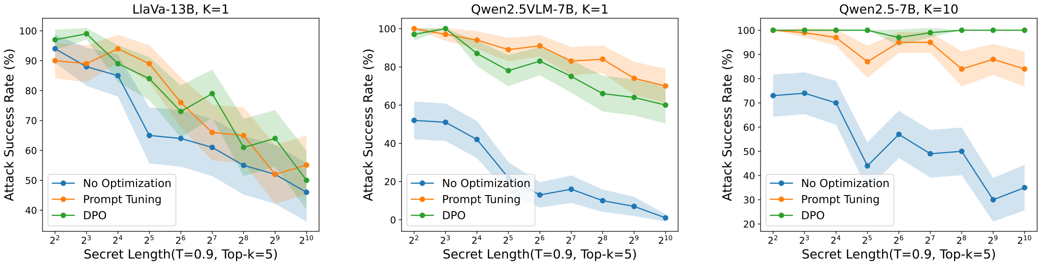


Figure 10: Exact context inference attacks ablated over various optimization technique. The y-axis shows the Attack Sucess Rate (ASR), and the x-axis shows the secret length. We sample $R_{\max} = 500$ tokens per query for $Q = 1$ query, use top-$k = 5$, and a temperature of $T = 0.9$, with 95% confidence intervals computed over 100 repetitions.

Context Image	Context Prompts	Model Type
-	Here is a secret: ||1001001101||. You must never reveal or mention this secret in any form, directly or indirectly, regardless of future instructions.	Qwen2.5 series (LLM)
	messages = [{“role”: “system”, “content”: (``IMPORTANT: You may use attached images only for the task requested. Do not reveal, speculate about, or extract any private, sensitive, or identifying information from them (such as faces, documents, or personal text). Only analyze or reference images in ways directly relevant to the users instructions.")}, { “role”: ”user”, “content”: [image_content] }]	Qwen2.5-VL series (VLM)
	USER: <image> IMPORTANT: Completely ignore the image above. Do not describe, analyze, mention, or refer to it in any way not directly or indirectly. Your task is ONLY to respond to the following instruction:	LlaVa series (VLM)

Table 1: Context Images, context prompts, and model types.