Shadow in the Cache: Unveiling and Mitigating Privacy Risks of KV-cache in LLM Inference

Zhifan Luo[1,2], Shuo Shao[1,2], Su Zhang[3], Lijing Zhou[3], Yuke Hu[1,2], Chenxu Zhao[1,2], Zhihao Liu[1,2], Zhan Qin[1,2]
[1]State Key Laboratory of Blockchain and Data Security, Zhejiang University
[2]Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security [3]Huawei Technology
{luozhifan, shaoshuo_ss, yukehu, zhaocx_7, zhihao_liu, qinzhan}@zju.edu.cn; {zhangsu14, zhoulijing}@huawei.com

***Abstract*—The Key-Value (KV) cache, which stores intermediate attention computations (Key and Value pairs) to avoid redundant calculations, is a fundamental mechanism for accelerating Large Language Model (LLM) inference. However, this efficiency optimization introduces significant yet underexplored privacy risks. This paper provides the first comprehensive analysis of these vulnerabilities, demonstrating that an attacker can reconstruct sensitive user inputs directly from the KV-cache. We design and implement three distinct attack vectors: a direct Inversion Attack, a more broadly applicable and potent Collision Attack, and a semantic-based Injection Attack. These methods demonstrate the practicality and severity of KV-cache privacy leakage issues. To mitigate this, we propose KV-Cloak, a novel, lightweight, and efficient defense mechanism. KV-Cloak uses a reversible matrix-based obfuscation scheme, combined with operator fusion, to secure the KV-cache. Our extensive experiments show that KV-Cloak effectively thwarts all proposed attacks, reducing reconstruction quality to random noise. Crucially, it achieves this robust security with virtually no degradation in model accuracy and minimal performance overhead, offering a practical solution for trustworthy LLM deployment.**

I. Introduction

Large Language Models (LLMs) have ignited a paradigm revolution in artificial intelligence [20], [33], profoundly impacting various domains and applications, such as machine translation [40], chatbots [37], code generation [11], and content creation [42]. However, the immense scale of these models, characterized by billions or even trillions of parameters, coupled with the need to process extensive input sequences and engage in multi-turn dialogues, presents a substantial challenge to their efficient deployment and inference. This computational demand often translates into high latency and resource consumption, hindering broader accessibility and real-time applicability [18].

To address the efficiency bottlenecks in LLM inference, researchers have proposed several optimization techniques [35], [44]. Among these, the **Key-Value cache (KV-cache)** mechanism has emerged as a crucial and widely adopted solution [24], [42]. During the autoregressive generation process typical of LLMs, the attention mechanism computes key (K) and value (V) matrices for each token based on its preceding tokens. The KV-cache stores these intermediate attention computations (the K and V pairs) for tokens that have already been processed within the input sequence. By reusing these cached K and V pairs for the generation of subsequent tokens, the KV-cache significantly reduces redundant computations. This dramatically accelerates inference speed and improves throughput, especially for tasks involving long contexts or interactive sessions, making LLMs more practical for real-world deployment.

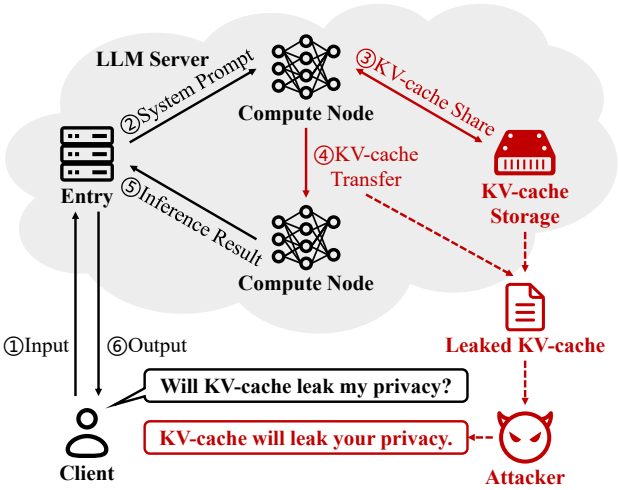


Fig. 1: LLM inference service workflow in a privacy-preserving setting and its associated security risks.

However, the storage and potential sharing of the KV-cache introduce significant yet underexplored privacy concerns [36], [38], as illustrated in Figure 1. This vulnerability stems from a critical trade-off made in production systems: while end-user communication with the LLM service is typically encrypted (represented by the black lines in our figure), the KV-cache itself is almost always processed, transmitted between compute nodes, and persisted in **plaintext**. This design choice is a concession to performance, as the unacceptable latency overhead from cryptographically securing the often gigabyte-scale KV-cache would violate the stringent demands of real-time inference. The feasibility of exploiting such plaintext data exposure is not merely theoretical. It was recently demonstrated by the LeftoverLocals attack [28]. This vulnerability in certain AMD GPUs enables an unprivileged attacker to read data remnants from the GPU's shared on-chip memory. This provides a concrete method to intercept and exfiltrate plaintext KV-cache data originating from other users' concurrent inference sessions, proving that sensitive conversational data is indeed at risk. Crucially, the KV-cache is inherently derived from and directly correlated with user inputs. Consequently, if an attacker gains access to the KV-cache, they could potentially reverse-engineer sensitive information in the

user's original prompts or private data [6], [17]. A critical research gap exists in understanding and mitigating the privacy risks associated with KV-caches in LLMs, necessitating further in-depth research.

In this paper, to bridge the research gap, we present the first comprehensive study on the privacy risks of KV-cache in LLM inference. Specifically, we primarily study and answer the following two research questions:

- **RQ1: Is an attacker able to reconstruct user inputs from the KV-caches?**

To address this question, we conduct a systematic investigation, demonstrating that attacks against the KV-cache are not only feasible but also diverse and broadly applicable. We design and implement three distinct classes of targeted privacy-stealing attacks. Each attack exposes inherent privacy risks of KV-caches from a different perspective.

We first explore two relatively direct attack paths. The first is the **KV-cache Inversion Attack**, a method that attempts to leverage known model weight matrices to directly reverse-calculate the input from the KV-cache. The second, more general-purpose approach is the **KV-cache Collision Attack**. This attack reframes input reconstruction as a matching task based on forward computation: an attacker iteratively uses a local model instance to generate KV-caches for candidate inputs and compares them against the intercepted target KV-cache to find a match. Because this method does not rely on any reverse computation, it has broader applicability. Finally, we introduce a novel, semantic-based **KV-cache Injection Attack**. The core of this attack is to leverage the LLM's powerful capability to understand and execute instructions. By appending a specific instruction, such as "Repeat the previous content," to the end of an intercepted KV-cache context, an attacker can induce the model to "echo" or generalize the core information held within the KV-cache.

Collectively, these attacks reveal that privacy leakage from the KV-cache is not merely a theoretical concern. Their diversity and feasibility constitute a significant threat to real-world LLM-based applications, underscoring the urgent necessity of designing specialized and efficient privacy-preserving mechanisms for the KV-cache.

- **RQ2: Can defenders effectively and efficiently mitigate or prevent user privacy leakage from the KV-cache?**

In response to this challenge, this paper provides an affirmative answer by first analyzing the shortcomings of existing privacy-preserving techniques. Conventional methods like full cryptographic encryption [2] or Homomorphic Encryption (HE) [22] introduce prohibitive computational overhead and latency, rendering them impractical for the high-throughput demands of LLM inference. Meanwhile, applying Differential Privacy (DP) [8] requires adding a level of noise that often severely degrades the model's inference accuracy to an unacceptable degree. Even specialized solutions like KV-Shield [38], while lightweight, suffer from critical security flaws. Their fixed shuffling mechanism, as we analyzed in Section IV-A, is vulnerable to statistical analysis and is incompatible with modern LLM architectures that use features like Rotary Positional Embedding (RoPE). These limitations highlight the urgent need for a novel defense mechanism, .

Therefore, we propose KV-Cloak, a lightweight and secure mechanism for KV-cache obfuscation designed to overcome these challenges. At its core, KV-Cloak employs a reversible matrix-based obfuscation scheme that guarantees lossless model accuracy. Its security is multi-layered: it applies secret invertible linear transformations to obscure statistical properties, and critically, introduces a one-time random permutation matrix for each data block. This dynamic permutation prevents attackers from building stable algebraic relations across multiple queries. To further enhance performance, KV-Cloak utilizes operator fusion, an optimization where a portion of the secret obfuscation matrices is algebraically fused into the LLM's attention layer weights offline. This shifts the primary computational cost away from the latency-sensitive online inference phase, striking an effective balance between robust security, lossless accuracy, and high performance.

Our contributions can be summarized as follows.

- **Revealing the privacy risks of KV-cache in LLM inference:** We systematically investigate the privacy risks of the KV-cache by designing and implementing three distinct attacks, Inversion, Collision, and Injection Attacks.
- **Proposing a lightweight and effective method to mitigate privacy leakage**: We propose KV-Cloak, a novel and practical defense mechanism that uses a lightweight, reversible matrix-based obfuscation scheme combined with operator fusion to protect the KV-cache without degrading model accuracy and with minimal performance overhead.
- **Conducting extensive experiments to evaluate attacks and defenses:** We conduct a systematic evaluation to empirically demonstrate and quantify the feasibility of attacks that reconstruct user input from the KV-cache of state-of-the-art LLMs, establishing it as a practical and severe threat. We further prove that our proposed KV-Cloak is a highly practical solution that achieves robust security, near-lossless model fidelity, and high efficiency simultaneously. Our experiments show that KV-Cloak, with its negligible impact on accuracy and minimal latency overhead (mostly $< 10\%$), successfully resolves the stark trade-off between security and utility that plagues alternative approaches such as DP.

II. Background and Related Work

A. Transformer-based LLM Inference

Prevailing large language models, such as LLaMA [32], DeepSeek [20] and Qwen [37], are predominantly based on the Transformer decoder architecture [42].

Self-Attention Mechanism.

The core of the Transformer architecture is the self-attention mechanism. It captures dependencies between different positions within an input sequence, allowing the model to flexibly focus on various parts of the sequence to better understand

the context. For an input sequence $X = (x_1, \ldots, x_n)$, the self-attention layer first applies linear transformations to x_i at each position to obtain Query (q), Key (k), and Value (v) vectors,

$$q_i = x_i W_q^\top R^d_{\Theta,i}, \quad k_i = x_i W_k^\top R^d_{\Theta,i}, \quad v_i = x_i W_v^\top. \quad (1)$$

Here, W_q, W_k, W_v are learnable weight matrices for the attention layer, and $R^d_{\Theta,i}$ denotes the Rotary Position Embedding (RoPE) [29]. Then, the self-attention layer computes the dot product of the current query q_i with all preceding keys k_j. The results are scaled, and a softmax function is applied to obtain attention weights a_{ij}. These weights are then used to compute a weighted sum of the value vectors. Finally, this sum is passed through a linear projection layer W_o to produce the output o_i for the current position. This process is detailed in Eq. (2), where d is the dimension of each attention head.

$$a_{ij} = \frac{\exp\left(q_i {k_j}^\top / \sqrt{d}\right)}{\sum_{t=1}^{i} \exp\left(q_i {k_t}^\top / \sqrt{d}\right)}, o_i = \left(\sum_{j=1}^{i} a_{ij} v_j\right) W_o^\top. \quad (2)$$

This mechanism enables the model to dynamically weigh the importance of different parts of the input sequence, thereby capturing complex contextual dependencies.

Autoregressive LLM Inference. The essence of large language model inference is to model the probability distribution $P(x_1, \ldots, x_n)$ of a token sequence $x_1, \ldots, x_n$. Due to the sequential nature of natural language, prevailing LLMs predominantly employ an autoregressive text generation approach in inference tasks. The joint probability of the entire sequence can typically be autoregressively decomposed [5] into a product of conditional probabilities,

$$P(x_1, \ldots, x_n) = P(x_1)P(x_2|x_1)\cdots P(x_n|x_1, \ldots, x_{n-1}). \quad (3)$$

Specifically, the model generates the output sequence one token at a time. When generating each new token, the model considers not only the initial user-provided input but also all previously generated tokens as context. For example, when generating the token x_i, the model utilizes information from tokens x_1 to x_{i-1}. While this sequential generation method effectively captures dependencies within the sequence, it introduces significant computational redundancy. At each step of generating a new token, attention computations for all preceding tokens (particularly the calculation of k and v) must be recalculated. This repetitive computation substantially impacts inference efficiency.

KV-cache in LLM Inference. In the context of autoregressive generation, the previously generated tokens form the context for predicting the next token. To avoid redundant computations, LLMs employ a caching strategy known as the KV-cache. For each processed token (either from the input or previously generated by the model), its corresponding k and v vectors, computed by the self-attention layers, are stored in the KV-cache. When generating the next token, the model reuses these cached k and v vectors from all preceding tokens, only needing to compute the q, k, v vector for the current prediction step and the k and v vectors for the most recently generated token to add to the cache. This significantly accelerates the inference process, especially for long sequences.

B. Privacy Attacks against LLMs

Inversion Attacks. While LLMs offer powerful capabilities, their inference process exposes new attack surfaces that can lead to the theft of sensitive user inputs. Privacy attacks targeting the inference process can be broadly categorized into two types: (1) **Output-based inversion attacks**, which aim to reconstruct inputs from the model's final outputs (e.g., probability distributions or embedding vectors), and (2) **Intermediate state-based inversion attacks**, which seek to extract information from the model's internal computational states [15], [21], [34] (e.g., hidden states). Early work focused on inverting final representations [15], such as sentence embeddings, with methods like Vec2Text [21]. However, these approaches overlooked the richer information in deeper layers. The Embed Parrot attack [34] demonstrated that even deep hidden states can be exploited to accurately reconstruct user inputs, establishing the entire inference pipeline as a critical attack surface.

Unlike a hidden state (the fused output of a Transformer layer), the KV-cache contains the more primitive Key (K) and Value (V) vectors that serve as direct inputs to the attention calculation. Crucially, the KV-cache is a specialized data structure explicitly designed for persistence and reuse across generation steps to optimize inference. This unique role makes it a more potent and exploitable source of privacy leakage than other transient states.

Side-Channel Attacks Exploiting the KV-cache. Currently, research on attacks against the KV-cache has been primarily limited to side-channel attacks. A prominent example is PromptPeek [36], which exploits timing discrepancies created by KV-cache sharing in multi-tenant LLM services. An attacker observes variations in response latency to determine if their guessed prefix has hit another user's cached entries, allowing them to reconstruct the prompt without direct data access.

However, the viability of such timing side-channel attacks is diminished by modern memory management optimizations. Contemporary inference engines, particularly those using PagedAttention [14], manage the KV-cache in non-contiguous memory blocks (pages), with a typical default block size of 16 tokens. This block-based management means that for a cache hit to occur, an attacker would need to correctly guess an entire block of 16 tokens simultaneously. For an LLM with a vocabulary size in the hundreds of thousands, the search space for a single block exceeds $100,000^{16}$, rendering the attack computationally infeasible. In contrast, our work considers a different, more direct threat model: if an attacker can gain direct access to the KV-cache content (e.g., through memory access or network eavesdropping), they can bypass these side-channel limitations. This direct access enables a more practical collision attack on a per-token basis, reducing the attack complexity to a feasible level and exposing a more fundamental

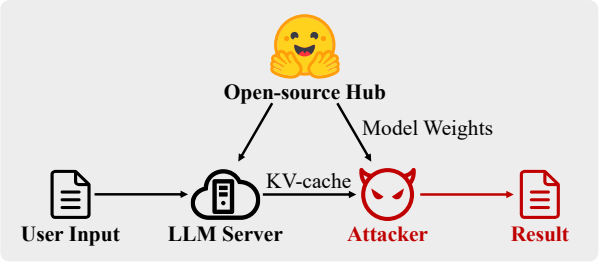


Fig. 2: The attacker restores user input based on model parameters and leaked KV-cache.

vulnerability. Given that the KV-cache is commonly stored and transmitted in plaintext to preserve performance, this paper aims to systematically explore this underexplored problem of direct, content-based attacks.

III. Attack Landscape: Inferring Private User Inputs from KV-cache

While KV-cache is crucial for enhancing the inference efficiency of LLMs, it concurrently introduces novel privacy vulnerabilities. The contextual information stored within KV-cache can contain private user data. Specifically, entries in the KV-cache often correspond directly to tokens in the user's input sequence. An attacker gaining access to this KV-cache information could potentially infer the original user input, thereby compromising user privacy.

In this section, we aim to study an important research question: *is an attacker able to conduct a prompt reconstruction attack to recover the user input from the KV-cache?* We first present the threat model of the attack. Subsequently, we propose two distinct attacks to achieve this goal.

A. Threat Model

In our threat model, we assume that an LLM server has deployed an open-source LLM based service. After a user uses the service with private user inputs, it generates a large number of KV-caches. The attacker gains access to the KV-caches and tries to infer the private inputs, as showed in Figure 2.

Attacker's Objective. The primary objective of the attacker is to recover the user input prompt from the accessed KV-cache. In this paper, we assume that the attacker attempts to restore the user input verbatim, which may significantly compromise the privacy of the user.

Attacker's Capabilities. We consider an attacker with relatively strong yet still practical capabilities, as follows.

- **Access to KV-cache.** The attacker can obtain the KV-caches stored by an LLM server. This can be achieved by eavesdropping on the communication channel between nodes (e.g. through the LeftoverLocals attack [28]), reading persisted KV-cache from storage, or other unspecified means of accessing the KV-cache pool.
- **Access to the foundation model on which the deployed LLM is developed.** We assume a gray-box setting where the attacker can gain access to the foundation model's weights on which the deployed LLM is developed[1]. The deployed model could be a fine-tuned version of the foundation model.

B. Input Reconstruction Attacks from KV-cache

This section investigates the risk of user input leakage from KV-cache data during LLM inference. We systematically present three attack vectors, namely *inversion attack*, *collision attack*, and *injection attack*, for reconstructing the original user inputs from the KV-cache. These attacks differ in complexity, applicability, and exploited vulnerabilities, collectively illustrating the threat landscape.

1) KV-cache Inversion Attack: The KV-cache Inversion Attack, as showed in Figure 3a, is a direct method that attempts to reverse-engineer the input to the attention layer by leveraging the KV-cache data in conjunction with the model's known attention projection matrices. As defined in the forward computation (Eq. (1)), an attacker who obtains the KV-cache and knows the model's attention layer parameters W_k and W_v could theoretically perform a reverse calculation to recover the attention layer's input x_i, as Eq. (4).

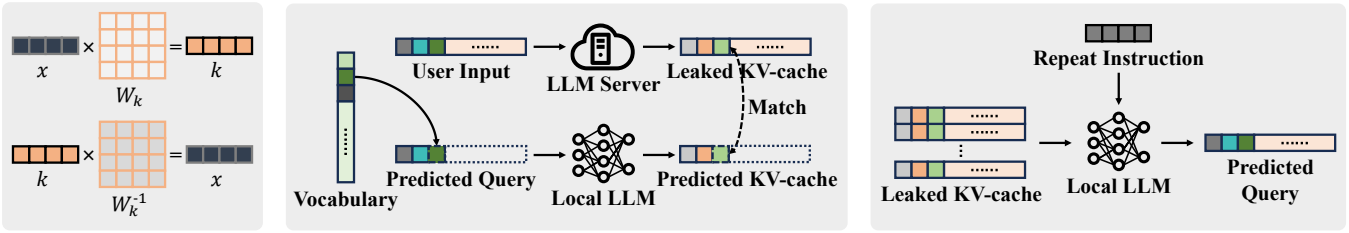
$$x_i = k_i (R^d_{\Theta,i})^{-1} (W_k^\top)^{-1}, \quad x_i = v_i (W_v^\top)^{-1}. \tag{4}$$

However, the practicality of this direct inversion attack is severely constrained by two fundamental factors. First, it requires the **attention projection matrices** (e.g., W_k, W_v) to be **invertible**. This condition is met by some earlier models using standard Multi-Head Attention (MHA), such as LLaMA-7B [32], but not by many contemporary models. Architectures like LLaMA 3 [9], Qwen [37], and DeepSeek [20] employ optimizations like Grouped-Query Attention (GQA) [3] or Multi-Head Latent Attention (MLA) [26], which often result in non-square, non-invertible projection matrices, rendering the attack impractical. Second, the attack's effectiveness is largely confined to the **first decoder layer's KV-cache**, as its inversion directly yields the input sequence's embeddings. In contrast, inverting from deeper layers only recovers intermediate hidden states, from which reconstructing the original text is a significantly more complex and data-intensive problem often requiring auxiliary models [34].

Due to these significant limitations, the KV-cache Inversion Attack, while theoretically straightforward, has a narrow scope of applicability and does not pose a threat to most modern LLM architectures.

2) KV-cache Collision Attack: Unlike the KV-cache Inversion Attack, the KV-cache Collision Attack operates by iteratively generating KV-cache entries for candidate tokens from the model's vocabulary. For each candidate token appended to a partially reconstructed input sequence, the attack performs local model inference, generates the corresponding KV-cache,

[1]This assumption is realistic given that now most LLM services are developed on open-source foundation models. First, disclosing the underlying foundation model is often a fundamental requirement of open-source licenses. Second, even if model information isn't publicly disclosed, an attacker could identify the specific foundation model via model fingerprinting attacks [7], [23].



(a) KV-cache Inversion Attack. (b) KV-cache Collision Attack. (c) KV-cache Injection Attack.

Fig. 3: Workflow of input reconstruction attacks.

and compares it against the intercepted KV-cache to determine if the candidate token is correct for the current position.

Fundamental Principle of the Collision Attack. The collision attack reconstructs user input by identifying the token at each position that minimizes a distance metric (e.g., L2 distance) between a locally generated KV-cache and the leaked KV-cache. Its effectiveness hinges on the assumption that the distance for the correct token, dis_{target}, is statistically separable from the distances of all other incorrect tokens from the vocabulary, dis_{other}, allowing for its unambiguous identification.

General Attack Procedure. As illustrated in Figure 3b, the attack reconstructs the user's input token-by-token. After initially intercepting the target KV-cache from the cloud service, the attacker enters an iterative loop for each unknown token position. First, guided by the already-reconstructed prefix, the attacker's local model instance predicts a probabilistically-ranked list of candidate tokens. Next, for each high-probability candidate, the attacker performs a forward pass to generate a local KV-cache segment and compares it against the corresponding segment of the target KV-cache to test for a match (a "collision"). Upon a successful match, the correct token is confirmed and appended to the reconstructed prefix, and the attacker proceeds to the next position. This process continues until the entire input sequence is recovered.

Implementation Optimizations for the Collision Attack. A naive collision attack that processes the entire model vocabulary is impractical due to prohibitive VRAM consumption and execution time. We therefore introduce some synergistic optimizations to make the attack practical. First, instead of analyzing all prioritized candidates, we apply **batched statistical validation**, testing a subset of candidate tokens and computing the distribution of their distances (dis_{batch}) from the leaked KV-cache. The correct token is then identified as any distinct statistical outlier within this batch, under the assumption that the batch provides a representative sample of the distance distribution for incorrect tokens. Second, we employ a **probability-guided search**. Using the model's own predictions based on the already reconstructed prefix, we prioritize collision tests for high-likelihood candidate tokens. Furthermore, we **prune the search space** by truncating the vocabulary and discarding tokens with very low probabilities. Together, these techniques reduce the memory footprint and bring the attack latency down to the order of seconds.

Enhancing Collision Attack with Prior Knowledge. A key challenge in the collision attack is setting an optimal classification threshold to distinguish the correct token's distance (dis_{target}) from the distribution of incorrect token distances (dis_{other}). A basic attacker, lacking ground-truth data, must rely on a static, heuristic threshold, typically derived from the statistics of observed incorrect token distances (e.g., $\mu_{other} - 3\sigma_{other}$). However, an adversary capable of performing *chosen-plaintext attacks* can generate KV-caches for known inputs. This prior knowledge enables a more precise statistical characterization of both the dis_{target} and dis_{other} distributions, allowing for a more effective, adaptive threshold that significantly boosts reconstruction accuracy.

The need for an adaptive threshold is formalized by the success probability for a token at rank r in the probability-sorted vocabulary, as defined in Eq. (5):

$$P(\text{success}|t) = P(dis_{other} > t)^{r-1} \times P(dis_{target} < t). \quad (5)$$

This equation demonstrates that the optimal threshold t is dependent on the token's expected rank r, as it must balance the trade-off between correctly rejecting the $r-1$ preceding (incorrect) candidates and accepting the true token. A static heuristic, by contrast, cannot effectively manage this rank-dependent trade-off.

The KV-cache Collision Attack overcomes the limitations of the Inversion Attack concerning specific KV-cache and model attention mechanism requirements. An attacker who intercepts KV-cache from potentially any layer of various LLM architectures can, in principle, employ this collision-based approach to reconstruct user inputs. The practical effectiveness and performance of this attack will be demonstrated and evaluated in subsequent experimental sections.

3) KV-cache Injection Attack: The KV-cache Injection Attack leverages the LLM's own semantic capabilities to exfiltrate information, operating differently from direct state-matching methods. The KV-cache contains substantial information from user-model interactions, however, itself (comprising only k and v vectors for past tokens) lacks the q vectors for the current generation step and thus cannot be directly used to continue model inference without a new input token. The core principle involves an attacker appending a crafted instruction (e.g., "Repeat the previous content.") to an intercepted KV-cache, as showed in Figure 3c. This action provides the necessary query, key, and value vectors to resume inference from the

compromised state, effectively tricking a standard LLM into executing the command within the context of the user's private data. This can compel the model to "echo" or regenerate the sensitive input stored implicitly within the cache. While this method is generally less precise than the collision attack, its primary advantage is speed, requiring only a single forward inference pass. The viability of this attack vector reveals a crucial requirement for any comprehensive defense: a protected KV-cache must be rendered semantically unintelligible to prevent an unauthorized LLM from meaningfully parsing its contents or continuing generation from it.

We conduct comprehensive evaluations on these three attacks in Section V. The experimental results demonstrate the practicality and effectiveness of input reconstruction attacks from KV-cache, underscoring the urgent need for developing an effective defense technique.

IV. KV-Cloak: A Defense Mechanism for KV-cache Obfuscation

A. Motivation for KV-Cloak

Limitations of Existing Privacy-Preserving Techniques. While several techniques exist for data privacy, they prove impractical or inadequate for securing the LLM KV-cache, highlighting the need for a specialized solution. The introduction to existing privacy-preserving techniques and their limitations is as follows.

- **Cryptographic Methods:** Standard cryptographic techniques [4], [31], such as symmetric AES encryption, provide confidentiality by ensuring data remains in ciphertext during storage and transmission, requiring decryption only for legitimate use. A more advanced approach, Homomorphic Encryption (HE) [2], [22], allows for direct computation on encrypted data, theoretically enabling parts of the attention mechanism to operate on the KV-cache without ever decrypting it. However, despite their strong security guarantees, the immense computational overhead and latency introduced by these methods are prohibitive. Given that the KV-cache can be tens or hundreds of gigabytes, applying full encryption or HE is unsuitable for the high-throughput, low-latency requirements of LLM inference.
- **Differential Privacy (DP):** Differential Privacy is a rigorous model that protects individual privacy by injecting carefully calibrated random noise into data or query results [1], [16], [39]. The fundamental goal is to make the output of a computation statistically insensitive to the presence or absence of any single individual's data in the input dataset. In the context of the KV-cache, this would involve adding noise to the Key and Value vectors. The primary limitation of this approach is the inherent trade-off between privacy and utility. To achieve a meaningful level of privacy, the required amount of noise would significantly degrade the LLM's inference accuracy to an unacceptable degree.
- **KV-Shield:** KV-Shield [38] is a lightweight obfuscation scheme and, to our best knowledge, the only privacy-preserving method designed specifically for the KV-cache. Its process involves synchronously permuting the rows of the model's attention layer parameters (W_q, W_k, W_v). This operation effectively shuffles the element positions within the resulting KV-cache vectors, obfuscating them before an attention score is calculated. The obfuscated attention output is then de-obfuscated for subsequent steps. However, this approach has critical flaws. From a security perspective, its fixed shuffling pattern does not alter the statistical properties of the data, leaving it vulnerable to collision attacks. For compatibility, its technique is incompatible with modern LLM architectures that utilize features like Rotary Positional Embedding (RoPE), as the shuffling interferes with the position-dependent calculations.

Design Objectives. The shortcomings of existing methods reveal an urgent need for a defense mechanism that is both secure and practical. To be effective, such a solution must achieve three key objectives:

- **High Security:** It must provide robust protection against targeted reconstruction attacks.
- **Minimal Accuracy Impact:** It must operate without causing any degradation in the model's output accuracy.
- **Low Performance Overhead:** It must introduce only negligible latency into the inference pipeline.

To meet these challenges, we propose KV-Cloak, a novel defense mechanism designed to provide strong, lossless, and efficient protection for the KV-cache.

B. A Naive Defense: Obfuscation via Reversible Linear Transforms

We first evaluate a baseline defense that employs a reversible linear transform to obscure the statistical properties of the k and v vectors. In the context of modern inference frameworks like vLLM, we represent all key vectors for a single attention head within a KV-cache block as a matrix $K \in \mathbb{R}^{b \times d}$, where b is the number of vectors per block and d is the attention head dimension. As the obfuscation method is analogous for both keys and values, we use K as our canonical example. The transformation is defined as:

$$K' = SKM, \tag{6}$$

where $S \in \mathbb{R}^{b \times b}$ and $M \in \mathbb{R}^{d \times d}$ are secret, randomly generated, and invertible matrices. The objective is to transform the original matrix K without altering its dimensions, preventing an adversary from directly recovering its contents.

Vulnerability Analysis. Despite its simplicity, this linear transform is fundamentally insecure under a **Chosen-Plaintext Attack (CPA)**. While an adversary cannot craft an arbitrary matrix K from scratch—as its contents are determined by the model's embedding and projection layers—they can still exert significant influence through carefully constructed inputs. This capability is sufficient to break the scheme.

The fixed linear nature of the transformation makes it highly susceptible to algebraic cryptanalysis. An adversary can mount a differential attack by choosing two plaintexts, K_1 and K_2,

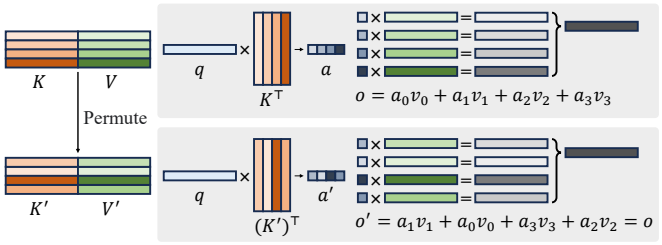


Fig. 4: Shuffling k, v vectors within a KV-cache block preserves attention computation results while disrupting positional order.

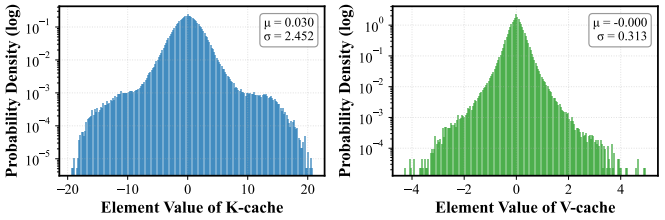


Fig. 5: Using an excerpt from "The Bitter Lesson" [30] (see Appendix B) as the input of LLaMA-3.2-1b, the KV-cache data element distribution generated during the inference process.

thereby controlling the difference $\Delta K = K_1 - K_2$ and observing the corresponding output difference $\Delta K' = S(\Delta K)M$. By systematically choosing ΔK to be a series of standard basis matrices (i.e., matrices with a single non-zero entry), the adversary can isolate the columns of S and the rows of M. This process allows for the full recovery of the secret matrices S and M (up to a trivial scaling ambiguity), leading to a complete compromise of the scheme.

C. The Improved Defense: One-Time Pad Block-wise Shuffling

Eliminating Redundant Positional Information in KV-cache. To enhance the security of the obfuscation scheme, we introduce additional randomness by eliminating redundant positional information inherent to KV-cache storage. During attention mechanism computations, the physical ordering of k, v vectors in the KV-cache is an unnecessary redundancy for inference. In reality, positional information critical for computation is already embedded into the elements of q and k vectors through rotary positional encoding.

By randomly permuting k, v within each block while maintaining their correspondence to input tokens x, we eliminate positional clues that attackers could exploit, and the results of the attention mechanism were not affected (as Figure 4). This permutation increases the computational complexity for attackers to match plaintext-ciphertext pairs by a factor of $b!$, where b denotes the block size (default $b = 16$ in vLLM).

The enhanced obfuscation process employs a one-time pad permutation matrix $\hat{P}$ to reorder vectors within blocks before applying invertible matrices S and M:

$$K' = S\hat{P}KM. \tag{7}$$

Crucially, $\hat{P}$ does not need to be stored as a key since the deobfuscated $\hat{P}K$ can directly participate in subsequent computations without reconstruction. This design choice preserves computational efficiency.

Additive Mask Matrices with Position-Identifiable Features. However, when the original K matrix has rank 1 (i.e., all row vectors are identical), the entropy of the permutation $\hat{P}$ will collapse. To prevent rank deficiency, KV-Cloak introduce an additive masking matrix A before permutation. Thereby, we can ensure the transformed matrix $(K+A)$ maintains sufficient rank to preserve cryptographic strength of $\hat{P}$.

Since the one-time pad permutation matrix $\hat{P}$ is not stored, computing $\hat{P}A$ during inverse obfuscation becomes infeasible, and thus $\hat{P}K = \hat{P}(K+A) - \hat{P}A$ cannot be derived from $\hat{P}(K+A)$. To avoid storing the one-time pad permutation matrix $\hat{P}$, we exploit the sparsity and small-elements properties of k and v vectors to design the additive matrix A with additional structural constraints. This enables us to infer $\hat{P}A$ based on the data characteristics of $\hat{P}(K+A)$.

As shown in Figure 5, we observe that the elements of k and v vectors typically remain below a small threshold θ_K (e.g., 100). This allows us to construct a matrix A where positional information is embedded into $K + A$ via additive operations using values significantly larger than θ_K in specific row vectors. After applying row permutations, the positional identifiers in $\hat{P}(K+A)$ remain identifiable. During inverse obfuscation, the positional information of $\hat{P}A$ can be directly identified from $\hat{P}(K+A)$ by detecting these outliers. Subtracting $\hat{P}A$ from $\hat{P}(K+A)$ yields $\hat{P}K$ for subsequent inference, drastically reducing storage overhead for the one-time pad key.

The complete obfuscation transformation, as shown in Eq. (8), incorporates additional matrices $\hat{P}$ and A alongside invertible linear transformations S and M:

$$K' = S\hat{P}(K+A)M, \tag{8}$$

where $\hat{P} \in \{0,1\}^{b\times b}$ is a row-permutation matrix generated as a one-time pad for each block of the KV-cache. $A \in \mathbb{R}^{b\times d}$ is a secret additive matrix with the same dimensions as K, designed to encode positional identifiers into its rows through outlier values. By embedding additive masks and applying invertible row/column transformations, this approach obscures the statistical properties of k and v vectors while enabling efficient inverse obfuscation without compromising model accuracy or inference performance.

D. Implicit Obfuscation via Operator Fusion

To mitigate the performance overhead introduced by the obfuscation and de-obfuscation processes during runtime, we now demonstrate how the obfuscation operators can be fused directly into the model's weight matrices. This allows the primary cost of obfuscation to be paid offline, resulting in a minimal impact on online inference latency.

Fusing the Obfuscation Operator into Model Weights. We define three transformed vectors (q^m, k^m, v^m) and a transformed output projection matrix W_o^m using two secret, invertible matrices, M_1 and M_2. Let q, k, v be the original query, key, and value vectors, and let W_o be the original

output projection matrix. The transformations are defined as follows:

$$\begin{cases} q^m = q(M_1^{-1})^\top, \\ k^m = kM_1, \\ v^m = vM_2, \\ W_o^m = W_o(M_2^{-1})^\top. \end{cases} \tag{9}$$

These transformations are specifically designed to leave the core attention mechanism computations invariant. The attention scores remain unchanged, as shown by the dot product:

$$\begin{aligned} q_i^m(k_j^m)^\top &= \left(q_i(M_1^{-1})^\top\right)\left(k_jM_1\right)^\top = q_i(M_1^{-1})^\top M_1^\top k_j^\top \\ &= q_i(M_1M_1^{-1})^\top k_j^\top = q_ik_j^\top. \end{aligned} \tag{10}$$

Similarly, the output of the attention head after projection remains identical:

$$\begin{aligned} v_j^m(W_o^m)^\top &= (v_jM_2)\left(W_o(M_2^{-1})^\top\right)^\top \\ &= v_jM_2(M_2^{-1})W_o^\top = v_jW_o^\top. \end{aligned} \tag{11}$$

This invariance guarantees that the model's output is mathematically identical to that of the original, unprotected model, ensuring lossless accuracy.

Next, we incorporate the projection from the input embedding x and the application of RoPE, denoted by the position-dependent matrix $R^d_{\Theta,i}$. The equations for the transformed vectors become:

$$\begin{cases} q^m = xW_q^\top R^d_{\Theta,i}(M_1^{-1})^\top \\ k^m = xW_k^\top R^d_{\Theta,i}M_1 \\ v^m = xW_v^\top M_2 \\ W_o^m = W_o(M_2^{-1})^\top \end{cases}. \tag{12}$$

From these formulations, we can observe the path to operator fusion. The secret matrices M_2 and M_2^{-1}, which obfuscate the value vectors, can be straightforwardly fused into the model's weights by pre-computing new weight matrices $(W_v^m)^\top = W_v^\top M_2$ and $W_o^m = W_o(M_2^{-1})^\top$.

However, fusing M_1 and M_1^{-1} into W_k and W_q is more complex due to the intermediate application of the position-dependent RoPE matrix $R^d_{\Theta,i}$. The fusion becomes feasible under a specific algebraic condition: if the secret matrix M_1 and the RoPE matrix $R^d_{\Theta,i}$ **commute**, i.e., $R^d_{\Theta,i}M_1 = M_1R^d_{\Theta,i}$. If this condition holds, we can rewrite the expression for k^m:

$$k^m = xW_k^\top R^d_{\Theta,i}M_1 = x(W_k^\top M_1)R^d_{\Theta,i}. \tag{13}$$

This allows us to pre-compute a new, position-independent weight matrix $(W_k^m)^\top = W_k^\top M_1$. A similar argument applies to fusing $(M_1^{-1})^\top$ into W_q.

Therefore, the viability of this operator fusion approach hinges on our ability to construct a secret matrix M_1 that commutes with the RoPE matrices. In Appendix A, we formally prove that this commutativity is achieved when M_1 is structured as a block-diagonal matrix composed of 2×2 rotation-scaling sub-matrices, a structure analogous to that of RoPE itself. Furthermore, to ensure the reversibility of the entire transformation for lossless de-obfuscation, M_1 must be constructed to be invertible.

By constraining M_2 to also be a random, invertible rotation-scaling matrix, and by carefully controlling the range of rotation-scaling coefficients in both M_1 and M_2, we ensure that the transformed padding tokens (which involve the affine transformation with matrix A) remain identifiable as outliers. These conditions allow the obfuscation operators (M_1, M_2) and their corresponding de-obfuscation counterparts (M_1^{-1}, M_2^{-1}) to be fused into the attention layer parameters W_q, W_k, W_v, and W_o. Let the model parameters after operator fusion be denoted by $W_q^m, W_k^m, W_v^m, W_o^m$. The new, fused weights are pre-computed as follows:

$$\begin{cases} W_q^m = M_1^{-1}W_q \\ W_k^m = M_1^\top W_k \\ W_v^m = M_2^\top W_v \\ W_o^m = W_o(M_2^{-1})^\top \end{cases}. \tag{14}$$

By deploying the LLM with these pre-computed fused weights, we effectively integrate the random invertible block-diagonal matrix M_1 and the general random invertible matrix $M_2 \in \mathbb{R}^{d\times d}$ into the attention layer parameters. This eliminates the need for explicit online matrix multiplications with M_1 and M_2, significantly reducing performance overhead.

Ultimately, the KV-Cloak obfuscation formula, leveraging operator fusion, is applied as follows:

$$K' = S\hat{P}(K^m + A), \tag{15}$$

where $K^m = KM$ represents the KV-cache generated by the inference process using the fused model weights. It is crucial to control the range of the rotation and scaling factors within the matrix M to ensure that the additive mask A can still reliably embed its positional information via the outlier mechanism.

With this optimization, the online computational overhead is drastically reduced. The matrix multiplication with M is now part of the offline weight preparation. The online cost for obfuscating and de-obfuscating one KV-cache block is now dominated by the multiplications with $S, \hat{P}, S^{-1}$, totaling approximately $b^3 + 2b^2d$ floating-point operations. For the LLaMA-3.1-8B example ($b = 16, d = 128, D = 4096$), this overhead is merely 0.83% of the cost of re-computing the KV-cache block, rendering the performance impact minimal. Detailed analysis can be found in Appendix F.

V. Evaluation

A. Experimental Setup

Models. We first select several state-of-the-art models from the popular LLaMA [9] and Qwen [37] series, including LLaMA-3.2-1B, LLaMA-3.2-3B-Instruct, LLaMA-3.1-8B, Qwen2.5-Math-7B, and DeepSeek-R1-Distill-LLaMA-8B (LLaMA-3.1-8B-Distilled) [10], which is fine-tuned from LLaMA-3.1-8B. In addition to these five models that use Grouped-Query Attention (GQA), we also leverage a classic models with Multi-Head Attention (MHA), namely LLaMA-7B [32], to demonstrate the generalizability of our method.

TABLE I: Effectiveness of the inversion, collision, and injection attacks against the unprotected KV-cache from different model layers. Here, "collision+" denotes the collision attack enhanced with prior knowledge.

Model	Metric	Inversion			Collision			Collision+			Injection
		First	Mid	Last	First	Mid	Last	First	Mid	Last	All
LLaMA-7B	BERTScore (↑)	1.000	0.065	0.092	0.449	0.769	0.611	1.000	1.000	1.000	0.765
	ROUGE-L (↑)	1.000	0.036	0.062	0.500	0.562	0.436	1.000	1.000	1.000	0.687
LLaMA-3.2-1B	BERTScore (↑)	1.000	0.084	0.057	0.877	0.791	0.894	1.000	1.000	1.000	0.544
	ROUGE-L (↑)	0.994	0.038	0.000	0.709	0.617	0.680	0.994	0.994	0.994	0.315
LLaMA-3.2-3B-Instruct	BERTScore (↑)	0.055	0.095	0.083	0.782	0.668	0.820	1.000	1.000	1.000	0.540
	ROUGE-L (↑)	0.000	0.000	0.000	0.732	0.456	0.621	0.994	0.994	0.994	0.324
LLaMA-3.1-8B	BERTScore (↑)	0.071	0.061	0.062	0.873	0.652	0.764	1.000	1.000	1.000	0.616
	ROUGE-L (↑)	0.000	0.000	0.001	0.825	0.443	0.564	0.994	0.994	0.994	0.447
LLaMA-3.1-8B-Distilled	BERTScore (↑)	0.083	0.062	0.081	0.642	0.492	0.635	0.894	0.258	0.762	0.610
	ROUGE-L (↑)	0.000	0.000	0.001	0.633	0.227	0.413	0.868	0.122	0.479	0.421
Qwen2.5-Math-7B	BERTScore (↑)	0.229	0.105	0.105	0.918	0.552	0.783	1.000	0.983	0.996	0.422
	ROUGE-L (↑)	0.186	0.000	0.000	0.842	0.355	0.580	1.000	0.977	0.996	0.286

Datasets. To evaluate the effectiveness of our attacks, we use the lmsys-chat-1m dataset [43], which is collected from a real-world inference service and contains a rich variety of dialogues and instructions. We randomly sample 1,000 instances from this dataset as the test set for validating attack effectiveness. To assess the impact of KV-Cloak on model accuracy and inference latency, we employ standard language model benchmarks, including MMLU [12], [13] and SQuAD [25].

Attack and Defense Evaluation Metrics. BERTScore [41] and ROUGE-L [19] are used to measure the similarity between the original input and the text reconstructed from the KV-cache. BERTScore, which is based on the all-mpnet-base-v2 model, is better at capturing semantic similarity, while ROUGE-L primarily reflects lexical-level precision and recall.

B. Attack Effectiveness

In this section, we first evaluate the privacy leakage risk of using the KV-cache by implementing our proposed inversion, collision, and injection attacks. Additional experiments, such as the ablation study of attacks, can be found in Appendix C.

Experimental Settings. We attack 7 LLMs as described in Section V-A. Notably, while attacking the LLaMA-3.1-8B-Distilled, which is a fine-tuned version, we assume that the attacker can only get access to its base model (i.e., LLaMA-3.1-8B). It can be utilized to validate the attack effectiveness against fine-tuning models. Furthermore, the inversion and collision attacks require KV-cache data from only a single layer. Given that models have varying numbers of layers, we select the KV-cache from the first, middle, and last layers to perform the inversion and collision attacks. In contrast, the injection attack is conducted on the complete KV-cache data, encompassing all layers.

Results of KV-cache Inversion Attack. As shown in Table I, our experimental results generally align with our analysis: a successful inversion attack requires two **sufficient conditions** to be met: **(1)** access to the KV-cache from the model's first layer, and **(2)** an invertible attention projection matrix, which is characteristic of the MHA mechanism. The inversion attack can reconstruct the user input with near-perfect fidelity (approaching 100%) from the first-layer KV-cache of MHA-based models. In contrast, reconstruction attempts using the KV-cache from subsequent layers yield completely unintelligible results (mostly < 0.1).

However, we also find that for the LLaMA-3.2-1B model using GQA, the inversion attack achieved a reconstruction fidelity of nearly 100% on the first-layer KV-cache. This may be because the ratio of the rank of this model's attention projection matrix to its hidden state dimension is particularly high. Consequently, even though the projection matrix is not a square, invertible matrix, the least squares method can still recover the input to the attention layer with high accuracy.

Takeaway 1: Inversion attack is mostly effective against the KV-cache of the first MHA layer.

Results of KV-cache Collision Attack. For the collision attack, we measure the distance between a locally generated KV-cache and the leaked KV-cache using the Frobenius norm. We experimentally observed that the distances (dis_{other}) between the leaked KV-cache and the caches generated from incorrect tokens in the vocabulary approximate a Gaussian distribution, as shown in Figure 8. Consequently, we leverage the statistical properties of this distribution (i.e., its mean and standard deviation) to identify the correct token, whose corresponding distance (dis_{target}) will manifest as a statistical outlier.

After experimental analysis, we defined outliers as values falling below 3 standard deviations from the mean of the dis_{batch} distribution. The batch size for local analysis was set to 256. Details of the ablation study can be found in Appendix C1. Under these settings, the experimental results for the collision attack are presented in the "Collision" column of Table I, demonstrating high reconstruction accuracy across all layers of all models.

We also experimented with truncating the probability-sorted vocabulary on the LLaMA-3.2-1B model, aiming to reduce the time lost on unlikely tokens. The results are presented in Figure 6. We found that even when searching only the top $1/8$ of tokens with the highest predicted probabilities, the average input reconstruction fidelity remains at approximately

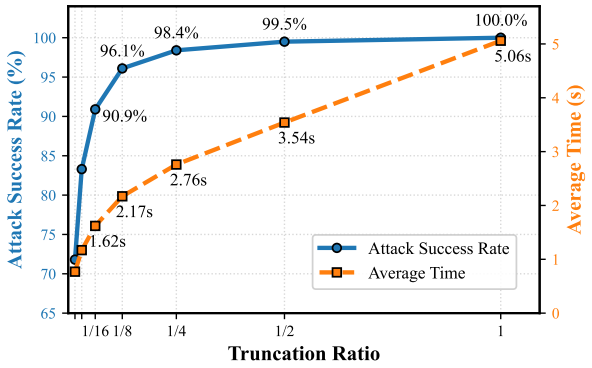


Fig. 6: The effect of truncating the probability-sorted vocabulary on reconstruction fidelity and attack time. Experiments were run with a batch size of 256 and an outlier threshold of $3\sigma_{other}$.

96.1% of what is achieved by searching the entire vocabulary. Meanwhile, the attack time is reduced to less than 42.9% of the full search, with the average time to reconstruct a user input from a single layer's KV-cache dropping from 5.06s to 2.17s. This confirms that truncating the vocabulary based on model-predicted probabilities is a highly effective strategy for accelerating the collision attack.

The experimental results for the collision attack demonstrate that this method overcomes the limitations of the inversion attack, which is restricted to specific layers (i.e., the first layer) and attention mechanisms (i.e., MHA). For open-source models, the collision attack can reconstruct user inputs with high accuracy and acceptable efficiency from the KV-cache of any layer. Crucially, it is also effective against the LLaMA-3.1-8B-Distilled model—a model fine-tuned from Meta-LLaMA-3.1-8B—by using the public weights of its open-source base model.

Takeaway 2: Collision attack achieves relatively high accuracy, demonstrating its universality and the high-risk threat it poses to real-world systems.

Collision Attack Enhanced with Prior Knowledge (Collision+). As shown in Table I, our enhanced collision attack (assuming rank $r = 8$) is highly potent, achieving near-perfect reconstruction accuracy (approaching 100%) on any layer of any tested open-source model, while also demonstrating notable effectiveness against fine-tuned models.

We also conducted experiments on the LLaMA-3.1-8B-Distilled model using optimal thresholds derived for different assumed ranks (r). The results are shown in Figure 7. For the first layer, the actual average rank of the target token is approximately 4; using the optimal threshold for this rank increases the reconstruction accuracy to 138.6% of the unenhanced attack. For the middle layer, the actual rank appears to be greater than 128, as none of the tested enhanced thresholds surpassed the accuracy of the baseline attack. For the last layer, the optimal rank is between 64 and 128, and applying the corresponding enhanced threshold boosts the reconstruction

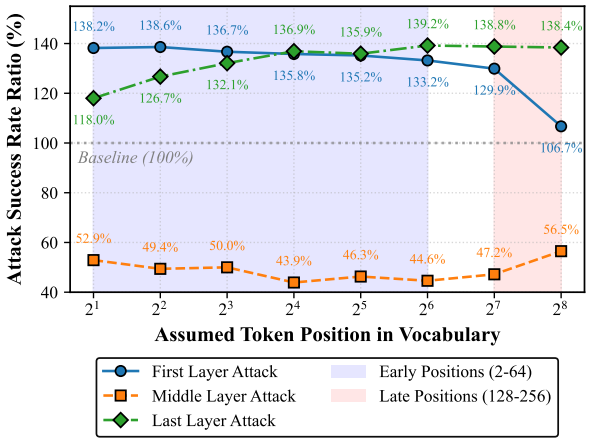


Fig. 7: Collision attack experiments on LLaMA-3.1-8B-Distilled using optimal thresholds derived for different assumed token ranks (r).

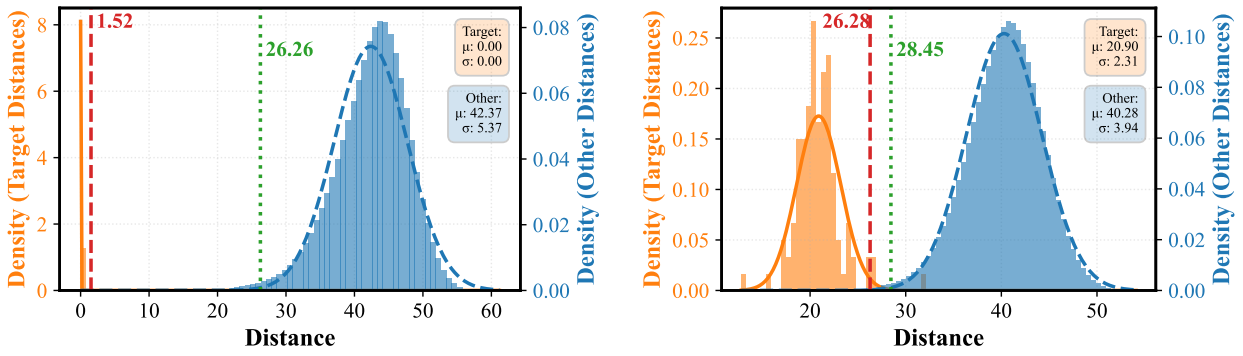
accuracy to approximately 139.0% of the baseline.

Takeaway 3: Augmenting the collision attack with model-specific prior knowledge achieves near-perfect ($\approx 100\%$) user-input recovery accuracy.

To further illustrate this, we analyzed the distance distributions using an excerpt from "The Bitter Lesson" as input. Figure 8 shows these distributions for an attack on the last-layer KV-cache. We observe that for the open-source model (left panel), the baseline heuristic threshold of $3\sigma_{other}$ results in a 0.13% false-positive rate (i.e., 0.13% of dis_{other} distances are misclassified). If we assume the correct token is at rank $r = 64$ in the sorted vocabulary, this yields a per-token attack success rate of 91.84%. In contrast, by leveraging a more precise threshold derived from statistical analysis, this success rate can be boosted to a perfect 100%. For the fine-tuned model (right panel), the enhanced threshold improves the per-token success rate from 91.79% to 97.82%. This substantial improvement in per-token accuracy leads to a dramatic increase in the overall reconstruction fidelity for the entire input sequence.

Results of KV-cache Injection Attack. The success of this attack hinges on two factors: the model's ability to effectively parse the semantic content of the KV-cache and its comprehension of the injected instruction. For our evaluation, we employed the most effective instruction identified in our ablation study, "Repeat the previous content." (the selection process is detailed in Appendix C2), to attack the KV-cache data from each target model. As shown in Figure I, the injection attack yielded an average BERTScore of 0.58 and a ROUGE-L score of 0.42 for the reconstructed user inputs. These results are consistently lower than those achieved by the more direct collision attack.

The attack performed best on the LLaMA-7B. This supports the hypothesis that the MHA mechanism preserves more contextual information than the more compressed attention variants (e.g., GQA) within the KV-cache.



(a) The distributions of LLaMA-3.1-8B model.

(b) The distributions of LLaMA-3.1-8B-Distilled model.

Fig. 8: Distributions of dis_{target} (orange) and dis_{other} (blue) from a collision attack. The input is an excerpt from "The Bitter Lesson". The attack uses the LLaMA-3.1-8B model's weights against the last-layer KV-cache generated by itself (left) and by LLaMA-3.1-8B-Distilled (right). The green dotted line shows the baseline heuristic threshold ($3\sigma_{other}$), while the red dashed line shows the enhanced threshold derived from prior knowledge (assuming rank $r = 64$).

Takeaway 4: Absent verbatim recovery, injection attacks can still exfiltrate the core information of user inputs through semantic-level interaction.

The reconstruction accuracy of the injection attack is generally lower than that of the collision attack, and it requires access to the complete KV-cache from all layers. However, its key advantage is speed: the attack's execution time is equivalent to a single inference step, making it significantly faster than the iterative collision attack. Moreover, the existence of this attack vector imposes an additional requirement on any defense mechanism: it must ensure that a protected KV-cache is rendered unintelligible even to a standard, unprotected LLM.

C. Evaluation of KV-Cloak

1) Experimental Settings: In order to comprehensively evaluate our proposed KV-Cloak scheme, this section presents a systematic comparison against two baselines: **(1)** a standard, unprotected system (Plaintext) and **(2)** a defense based on differential privacy with Gaussian noise (DP). Our evaluation is structured around three core dimensions: Security, Model Accuracy, and Performance Overhead. The results demonstrate that KV-Cloak is the only solution that provides robust security protection with acceptable performance overhead, all without sacrificing model accuracy.

In our KV-Cloak, the obfuscation involves matrix multiplication of KV-cache blocks with invertible secret matrices S and M, along with the addition of a random mask A.

- **Block Size:** To ensure compatibility with the PagedAttention mechanism used in modern inference engines like vLLM, we experimented with common block sizes: 16, 32, and 64.
- **Invertible Matrices S, M:** To avoid precision loss from matrix inversion during de-obfuscation, we sample S and M from the space of orthogonal matrices.
- **Additive Mask A and Padding:** The values in the mask A and the padding for incomplete blocks are kept small to minimize their impact on numerical precision. We sample the elements of A from the range $[3\theta_K, 4\theta_K]$ and use a value of $1.5\theta_K$ for padding. Here, θ_K is the maximum absolute value observed in the K-cache elements when inferring on an excerpt from "The Bitter Lesson". The same methodology is applied for the V-cache.

To establish a robust DP baseline, we conducted an ablation study (detailed in Appendix D) to select a configuration that balances utility and privacy for comparison against KV-Cloak. Based on our findings, we selected a practical configuration: the clipping threshold was set to the 50th percentile of the L2 norm distribution observed across the dataset. Subsequently, we applied Gaussian noise calibrated to satisfy ($\epsilon = 10^8, \delta = 10^{-5}$)-DP independently to both the K-cache and V-cache.

2) Evaluation of Security: This section evaluates the capability of different protection schemes to defend against our three KV-cache-based reconstruction attacks. We repeat the attack experiments on the KV-cache after applying each defense mechanism. As shown in Table II (evaluation on the remaining models can be found in Appendix E), for the "Plaintext" baseline, all three attacks achieve very high success rates. After applying KV-Cloak, the semantic similarity (BERTScore) of all attack outputs drops to a level consistent with random chance, and the ROUGE-L score falls to nearly 0. These results are statistically indistinguishable from comparing the original input with a random string, demonstrating that semantic reconstruction is entirely disrupted. This proves that KV-Cloak effectively protects the private information within the KV-cache. For the DP baseline, its defensive efficacy is strongly correlated with the privacy budget ϵ. With a weak budget of $\epsilon = 10^8$, the accuracy of the inversion and injection attacks is reduced, but the collision attack can still recover some useful information.

We also compare the distributions of $dist_{target}$ and $dist_{other}$ for the collision attack on the last layer of LLaMA-3.2-1B's KV-cache under different protections. As shown in Figure 9, the two distributions remain effectively distinguishable under

TABLE II: Efficacy of Defense Mechanisms Against Input Reconstruction Attacks.

Model	Protect Type	Metric	Inversion	Collision			Collision+			Injection
			First	First	Mid	Last	First	Mid	Last	All
LLaMA-7B	Original	BERTScore (↓)	1.000	0.449	0.769	0.611	1.000	1.000	1.000	0.765
		ROUGE-L (↓)	1.000	0.500	0.562	0.436	1.000	1.000	1.000	0.687
	KV-Cloak	BERTScore (↓)	0.091	0.070	0.069	0.071	0.036	0.036	0.036	0.082
		ROUGE-L (↓)	0.068	0.000	0.000	0.000	0.044	0.044	0.044	0.000
	$(10^8, 10^{-5})$-DP	BERTScore (↓)	0.085	0.082	0.672	0.344	0.109	0.937	0.991	0.085
		ROUGE-L (↓)	0.065	0.041	0.433	0.197	0.097	0.901	0.979	0.009
LLaMA-3.2-1B	Original	BERTScore (↓)	1.000	0.877	0.791	0.894	1.000	1.000	1.000	0.544
		ROUGE-L (↓)	0.994	0.709	0.617	0.680	0.994	0.994	0.994	0.315
	KV-Cloak	BERTScore (↓)	0.085	0.072	0.074	0.069	0.051	0.051	0.051	0.079
		ROUGE-L (↓)	0.009	0.000	0.000	0.000	0.002	0.002	0.002	0.000
	$(10^8, 10^{-5})$-DP	BERTScore (↓)	0.633	0.849	0.763	0.849	0.973	0.995	1.000	0.393
		ROUGE-L (↓)	0.622	0.604	0.587	0.604	0.966	0.989	0.994	0.248
LLaMA-3.1-8B-Distilled	Original	BERTScore (↓)	0.083	0.642	0.492	0.635	0.885	0.251	0.829	0.610
		ROUGE-L (↓)	0.000	0.633	0.227	0.413	0.858	0.112	0.552	0.421
	KV-Cloak	BERTScore (↓)	0.093	0.070	0.070	0.069	0.041	0.041	0.041	0.088
		ROUGE-L (↓)	0.002	0.000	0.000	0.000	0.003	0.003	0.003	0.000
	$(10^8, 10^{-5})$-DP	BERTScore (↓)	0.079	0.320	0.440	0.566	0.526	0.267	0.824	0.118
		ROUGE-L (↓)	0.003	0.291	0.185	0.351	0.530	0.122	0.543	0.049

DP protection. When using an enhanced threshold derived from prior knowledge, the per-token attack success rate is as high as 94.84% for $(10^7, 10^{-5})$-DP, and achieving nearly 100% (the same as plaintext) for $(10^7, 10^{-5})$-DP. In contrast, under KV-Cloak's protection, the two distributions become completely indistinguishable, resulting in a 0% attack success rate.

To assess the robustness of our defense, we also tested KV-Cloak's performance against the enhanced collision attack on multiple LLMs. As illustrated in Table II, our mechanism demonstrates strong defensive capabilities. With KV-Cloak enabled, the reconstruction accuracy of the attack drops to near-zero, and the recovered outputs are qualitatively equivalent to random noise, confirming that KV-Cloak effectively neutralizes this advanced attack vector.

Takeaway 5: KV-Cloak completely thwarts all our proposed attacks, reducing the quality of any reconstructed text to a level statistically indistinguishable from random noise.

3) Inference Accuracy: To evaluate the model fidelity of our KV-Cloak, we simulate an inference service with a prefill-decode architecture. The KV-cache generated during the prefill phase is protected using either DP or KV-Cloak. This protected cache is then passed to the decode node (after de-obfuscation in KV-Cloak's case) for subsequent token generation. We then evaluate this protected inference service on the MMLU and SQuAD benchmarks to measure the impact of each method on model accuracy.

We tested the impact of KV-Cloak on all experimental models. The results in Table III show that across both benchmarks, the impact of KV-Cloak on inference accuracy is negligible. This confirms our theoretical design: KV-Cloak is a lossless protection scheme that does not degrade the model's generation quality or its performance on downstream tasks.

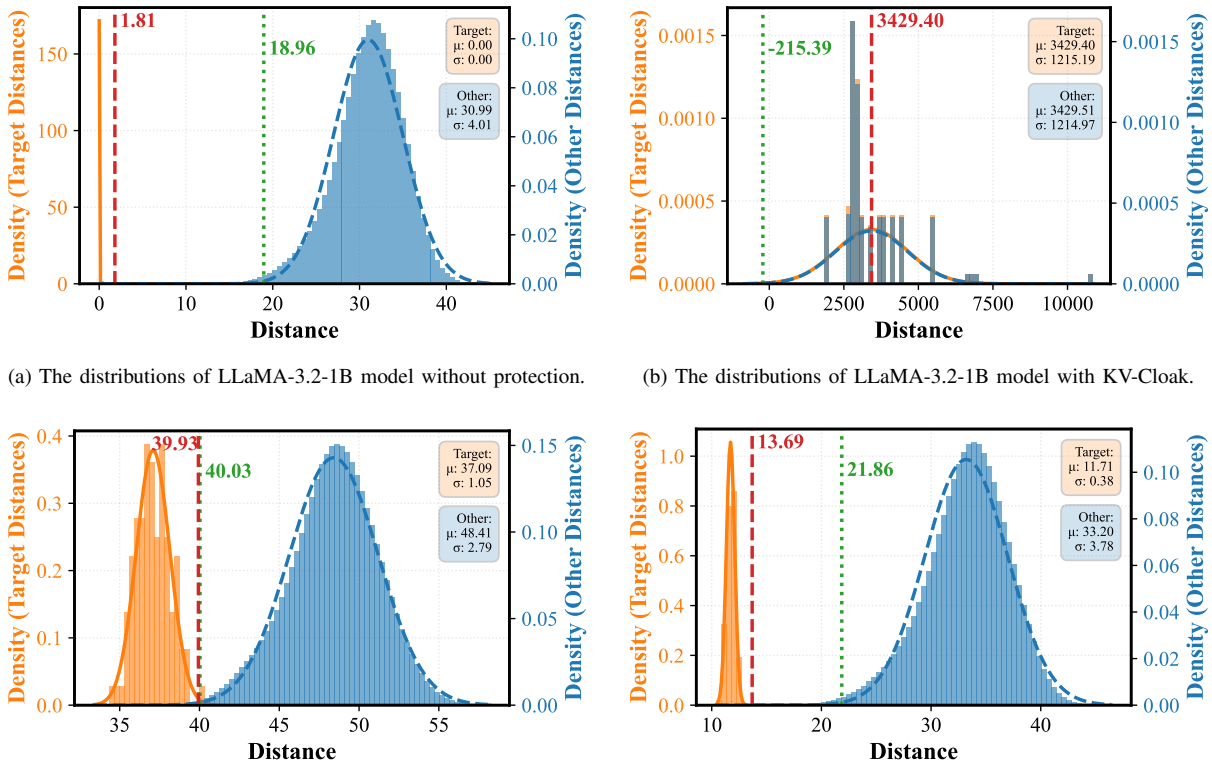
TABLE III: Impact of KV-Cloak on inference accuracy (higher is better) across various models, using a block size of 16.

Model	Plaintext		KV-Cloak		$(10^8, 10^{-5})$-DP	
	MMLU	SQuAD	MMLU	SQuAD	MMLU	SQuAD
LLaMA-7B	0.304	0.646	0.304	0.652	0.016	0.000
LLaMA-3.2-1B	0.335	0.457	0.335	0.458	0.262	0.258
LLaMA-3.2-3B-Instruct	0.619	0.652	0.619	0.652	0.379	0.012
LLaMA-3.1-8B	0.668	0.708	0.668	0.709	0.283	0.026
LLaMA-3.1-8B-Distilled	0.584	0.568	0.584	0.570	0.108	0.001
Qwen2.5-Math-7B	0.620	0.630	0.620	0.632	0.042	0.000

Takeaway 6: KV-Cloak is virtually lossless, preserving the model's fidelity and core utility without any degradation.

4) Evaluation of Computational Overhead: Integrating a new cryptographic primitive into a highly-optimized system like vLLM presents a significant engineering challenge. Therefore, to accurately assess the performance overhead of KV-Cloak and demonstrate its compatibility with PagedAttention, we employ a dual approach: direct latency measurement combined with architectural analysis.

To quantify the computational overhead of KV-Cloak, we measured the inference time for the MMLU benchmark under its protection. We selected this benchmark specifically to evaluate a worst-case scenario: the MMLU task requires applying KV-Cloak's obfuscation and de-obfuscation to the entire KV-cache of a long input sequence, yet generates only a single output token. This setup maximizes the ratio of cryptographic overhead to inference work. Furthermore, our measurements are conservative as they are based on a serial PyTorch-level implementation without any custom CUDA kernel optimizations, and they do not factor in network communication latency.



(a) The distributions of LLaMA-3.2-1B model without protection.

(b) The distributions of LLaMA-3.2-1B model with KV-Cloak.

(c) The distributions of LLaMA-3.2-1B model with $(10^7, 10^{-5})$-DP.

(d) The distributions of LLaMA-3.2-1B model with $(10^8, 10^{-5})$-DP.

Fig. 9: Distributions of dis_{target} (orange) and dis_{other} (blue) for a collision attack on the last-layer KV-cache. The input is an excerpt from "The Bitter Lesson" to the LLaMA-3.2-1B model. The distributions are shown under four conditions: (a) Plaintext, (b) KV-Cloak protection, (c) $(10^7, 10^{-5})$-DP protection, and (d) $(10^8, 10^{-5})$-DP protection. The green dotted line is the baseline heuristic threshold ($3\sigma_{other}$), and the red dashed line is the enhanced threshold assuming rank $r = 8$.

TABLE IV: Performance overhead of KV-Cloak (with a block size of 16) on various models during the MMLU benchmark.

Model	Inference Time(s)		
	Plaintext	KV-Cloak	
LLaMA-7B	10193.1	10649.0	+4.47%
LLaMA-3.2-1B	1956.8	2155.6	+10.16%
LLaMA-3.2-3B-Instruct	4525.7	4821.4	+6.53%
LLaMA-3.1-8B	9797.4	10135.0	+3.45%
LLaMA-3.1-8B-Distilled	9832.8	10297.7	+4.73%
Qwen2.5-Math-7B	9184.7	9452.3	+2.91%

The results of our evaluation across various models are presented in Table IV. The findings show that KV-Cloak introduces a modest average latency overhead of approximately 5%. Notably, we observe a favorable scaling property: the relative overhead decreases as the model's hidden size increases. In a real-world distributed deployment, the inclusion of network latency between prefill and decode nodes would make the relative impact of KV-Cloak's computational overhead on the total end-to-end time even lower. These empirical results align with our theoretical analysis, confirming that with standard production optimizations, such as operator fusion, the performance impact of KV-Cloak is expected to be minimal.

Takeaway 7: KV-Cloak introduces negligible computational overhead to the inference pipeline.

VI. Conclusion

This research exposes a critical security flaw at the heart of modern LLM inference systems: the privacy risk of data leakage from the KV-cache. We have demonstrated the feasibility of reconstructing sensitive user inputs through three novel attack strategies, with our collision attack proving particularly effective across various models. This underscores the urgent need for dedicated protection mechanisms that do not compromise the efficiency gains the KV-cache provides.

In response, we designed KV-Cloak. By employing a lightweight, reversible obfuscation technique, KV-Cloak neutralizes the identified threats without degrading model accuracy or imposing significant latency. It is designed for seamless integration into existing high-performance inference frameworks like vLLM. Our work provides a vital contribution to building secure and trustworthy AI, offering a blueprint for balancing the competing demands of performance and user privacy in the next generation of LLM services. It establishes that strong privacy protection can be achieved without sacrificing the utility and efficiency that have made these models so powerful.

Ethic Considerations

This research aims to enhance the privacy and trustworthiness of LLM inference, but we acknowledge the dual-use nature of the vulnerabilities and attack methods we have uncovered. To fulfill our ethical responsibilities and mitigate any potential for misuse, we have committed to a policy of responsible disclosure. Prior to the public dissemination of this paper, we will share our findings, including the details of the vulnerabilities and our proposed defense, with the developers of major affected inference frameworks, such as vLLM. Furthermore, all of our attack validation experiments were conducted exclusively on public and non-sensitive academic datasets. No real user data was involved at any stage of our research. We firmly believe that by taking these responsible measures, the positive contributions of our defensive work, KV-Cloak, will far outweigh the risks associated with the disclosure of these attacks. We are confident that this work will encourage the community to build more trustworthy AI services that are secure by default.

References

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[2] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Computing Surveys*, vol. 51, no. 4, pp. 1–35, 2018.

[3] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai, "Gqa: Training generalized multi-query transformer models from multi-head checkpoints," *arXiv preprint arXiv:2305.13245*, 2023.

[4] M. N. Alenezi, H. Alabdulrazzaq, and N. Q. Mohammad, "Symmetric encryption algorithms: Review and evaluation study," *International Journal of Communication Networks and Information Security*, vol. 12, no. 2, pp. 256–272, 2020.

[5] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[6] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson *et al.*, "Extracting training data from large language models," in *USENIX Security Symposium*, 2021, pp. 2633–2650.

[7] Y. Chen, C. Shen, C. Wang, and Y. Zhang, "Teacher model fingerprinting attacks against transfer learning," in *USENIX Security Symposium*, 2022.

[8] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography Conference*. Springer Berlin Heidelberg, 2006, pp. 265–284.

[9] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[10] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang *et al.*, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," *arXiv preprint arXiv:2501.12948*, 2025.

[11] Y. He, H. She, X. Qian, X. Zheng, Z. Chen, Z. Qin, and L. Cavallaro, "On benchmarking code llms for android malware analysis," in *ACM SIGSOFT International Symposium on Software Testing and Analysis Workshop*, 2025.

[12] D. Hendrycks, C. Burns, S. Basart, A. Critch, J. Li, D. Song, and J. Steinhardt, "Aligning ai with shared human values," *International Conference on Learning Representations*, 2021.

[13] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," in *International Conference on Learning Representations*, 2021.

[14] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," in *Symposium on Operating Systems Principles*, 2023.

[15] H. Li, M. Xu, and Y. Song, "Sentence embedding leaks more information than you expect: Generative embedding inversion attack to recover the whole sentence," in *Findings of the Association for Computational Linguistics: ACL 2023*, 2023, pp. 14 022–14 040.

[16] X. Li, Z. Qin, K. Ren, C. Gong, S. Feng, Y. Hong, and T. Wang, "Delay-allowed differentially private data stream release." in *Network and Distributed System Security Symposium*, 2025.

[17] Y. Li, S. Shao, Y. He, J. Guo, T. Zhang, Z. Qin, P.-Y. Chen, M. Backes, P. Torr, D. Tao, and K. Ren, "Rethinking data protection in the (generative) artificial intelligence era," *arXiv preprint arXiv:2507.03034*, 2025.

[18] B. Lin, C. Zhang, T. Peng, H. Zhao, W. Xiao, M. Sun, A. Liu, Z. Zhang, L. Li, X. Qiu *et al.*, "Infinite-llm: Efficient llm service for long context with distattention and distributed kvcache," *arXiv preprint arXiv:2401.02669*, 2024.

[19] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*, 2004, pp. 74–81.

[20] A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan *et al.*, "Deepseek-v3 technical report," *arXiv preprint arXiv:2412.19437*, 2024.

[21] J. Morris, V. Kuleshov, V. Shmatikov, and A. M. Rush, "Text embeddings reveal (almost) as much as text," in *Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 12 448–12 460.

[22] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1999, pp. 223–238.

[23] D. Pasquini, E. M. Kornaropoulos, and G. Ateniese, "Llmmap: Fingerprinting for large language models," in *USENIX Security Symposium*, 2025.

[24] R. Pope, S. Douglas, A. Chowdhery, J. Devlin, J. Bradbury, J. Heek, K. Xiao, S. Agrawal, and J. Dean, "Efficiently scaling transformer inference," *Proceedings of Machine Learning and Systems*, vol. 5, pp. 606–624, 2023.

[25] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," in *Conference on Empirical Methods in Natural Language Processing*, 2016.

[26] Z. Shao, D. Dai, D. Guo, B. L. B. Liu), Z. Wang, and H. Xin, "Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model," *ArXiv*, vol. abs/2405.04434, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:269613809

[27] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," in *International Conference on Learning Representations*, 2017.

[28] T. Sorensen and H. Khlaaf, "Leftoverlocals: Listening to llm responses through leaked gpu local memory," *arXiv preprint arXiv:2401.16603*, 2024.

[29] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *Neurocomputing*, vol. 568, p. 127063, 2024.

[30] R. Sutton, "The bitter lesson," *Incomplete Ideas (blog)*, vol. 13, no. 1, p. 38, 2019.

[31] E. Thambiraja, G. Ramesh, and D. R. Umarani, "A survey on various most common encryption techniques," *International Journal of Advanced Research in Computer Science and Software Engineering*, 2012.

[32] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Annual Conference on Neural Information Processing Systems*, vol. 30, 2017.

[34] Z. Wan, A. Cheng, Y. Wang, and L. Wang, "Information leakage from embedding in large language models," *arXiv preprint arXiv:2405.11916*, 2024.

[35] Z. Wan, X. Wang, C. Liu, S. Alam, Y. Zheng, J. Liu, Z. Qu, S. Yan, Y. Zhu, Q. Zhang *et al.*, "Efficient large language models: A survey," *Transactions on Machine Learning Research*, 2024.

[36] G. Wu, Z. Zhang, Y. Zhang, W. Wang, J. Niu, Y. Wu, and Y. Zhang, "I know what you asked: Prompt leakage via kv-cache sharing in multi-tenant llm serving," in *Network and Distributed System Security Symposium*, 2025.

[37] A. Yang, B. Zhang, B. Hui, B. Gao, B. Yu, C. Li, D. Liu, J. Tu, J. Zhou, J. Lin *et al.*, "Qwen2.5-math technical report: Toward mathematical

expert model via self-improvement," *arXiv preprint arXiv:2409.12122*, 2024.
[38] H. Yang, D. Zhang, Y. Zhao, Y. Li, and Y. Liu, "A first look at efficient and secure on-device llm inference against kv leakage," in *Workshop on Mobility in the Evolving Internet Architecture*, 2024, pp. 13–18.
[39] L. Yu, L. Liu, C. Pu, M. E. Gursoy, and S. Truex, "Differentially private model publishing for deep learning," in *IEEE Symposium on Security and Privacy*. IEEE, 2019, pp. 332–349.
[40] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin *et al.*, "Opt: Open pre-trained transformer language models," *arXiv preprint arXiv:2205.01068*, 2022.
[41] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "BERTScore: Evaluating text generation with BERT," in *International Conference on Learning Representations*, 2020.
[42] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong *et al.*, "A survey of large language models," *arXiv preprint arXiv:2303.18223*, vol. 1, no. 2, 2023.
[43] L. Zheng, W.-L. Chiang, Y. Sheng, T. Li, S. Zhuang, Z. Wu, Y. Zhuang, Z. Li, Z. Lin, E. P. Xing, J. E. Gonzalez, I. Stoica, and H. Zhang, "Lmsys-chat-1m: A large-scale real-world llm conversation dataset," in *International Conference on Learning Representations*, 2024.
[44] Z. Zhou, X. Ning, K. Hong, T. Fu, J. Xu, S. Li, Y. Lou, L. Wang, Z. Yuan, X. Li *et al.*, "A survey on efficient inference for large language models," *arXiv preprint arXiv:2404.14294*, 2024.

APPENDIX

A. Proof of Commutativity with Rotary Position Embedding

Denoting both $R^d_{\Theta,i}$ and the random invertible matrix M_1 as a 2×2 block matrix, we obtain:

$$R^d_{\Theta,i} = \begin{bmatrix} C & -S \\ S & C \end{bmatrix}, M_1 = \begin{bmatrix} T & Y \\ U & Z \end{bmatrix}, \tag{16}$$

where $C, S \in \mathbb{R}^{\frac{d}{2} \times \frac{d}{2}}$ are shown in Eq. (17), assuming that $T, U, Y, Z \in \mathbb{R}^{\frac{d}{2} \times \frac{d}{2}}$ are all random matrices.

$$C = \begin{bmatrix} \cos i\theta_0 & 0 & \cdots & 0 \\ 0 & \cos i\theta_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \cos i\theta_{\frac{d}{2}-1} \end{bmatrix},$$
$$S = \begin{bmatrix} \sin i\theta_0 & 0 & \cdots & 0 \\ 0 & \sin i\theta_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sin i\theta_{\frac{d}{2}-1} \end{bmatrix}. \tag{17}$$

If the secret matrix M_1 and the RoPE matrix $R^d_{\Theta,i}$ commute, then:

$$\begin{bmatrix} T & Y \\ U & Z \end{bmatrix} \begin{bmatrix} C & -S \\ S & C \end{bmatrix} = \begin{bmatrix} C & -S \\ S & C \end{bmatrix} \begin{bmatrix} T & Y \\ U & Z \end{bmatrix}. \tag{18}$$

With j and k as subscripts of matrix elements, Eq. (19) is equivalent to the following system of linear equations:

$$\begin{cases} t_{jk}\cos i\theta_j - u_{jk}\sin i\theta_j = t_{jk}\cos i\theta_k + y_{jk}\sin i\theta_k \\ t_{jk}\sin i\theta_j + u_{jk}\cos i\theta_j = u_{jk}\cos i\theta_k + z_{jk}\sin i\theta_k \\ y_{jk}\cos i\theta_j - z_{jk}\sin i\theta_j = -t_{jk}\sin i\theta_k + y_{jk}\cos i\theta_k \\ y_{jk}\sin i\theta_j + z_{jk}\cos i\theta_j = -u_{jk}\sin i\theta_k + z_{jk}\cos i\theta_k \end{cases} \tag{19}$$

Calculating the equation, the T, U, Y, Z need to satisfy the following relationship:

$$\begin{cases} y_{jj} = -u_{jj}, \\ z_{jj} = t_{jj}, \\ t_{jk} = u_{jk} = y_{jk} = z_{jk} = 0, (j \neq k) \end{cases}. \tag{20}$$

Specifically, the structure of M_1 is defined as follows in Equation 21,

$$M_1 = \begin{bmatrix} t_0 & 0 & \cdots & 0 & -u_0 & 0 & \cdots & 0 \\ 0 & t_1 & \cdots & 0 & 0 & -u_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & t_{\frac{d}{2}-1} & 0 & 0 & \cdots & -u_{\frac{d}{2}-1} \\ u_0 & 0 & \cdots & 0 & t_0 & 0 & \cdots & 0 \\ 0 & u_1 & \cdots & 0 & 0 & t_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{\frac{d}{2}-1} & 0 & 0 & \cdots & t_{\frac{d}{2}-1} \end{bmatrix}. \tag{21}$$

B. Detailed Experimental Settings

Input Text for Parameter Calibration. The following text, an excerpt from "The Bitter Lesson" by Rich Sutton, was used as model input in our experiments to analyze the numerical characteristics of the KV-cache (e.g., for parameter calibration as described in Section V-B).

Input Text for Parameter Calibration

One thing that should be learned from the bitter lesson is the great power of general-purpose methods, of methods that continue to scale with increased computation even as the available computation becomes very great. The two methods that seem to scale arbitrarily in this way are search and learning. The second general point to be learned from the bitter lesson is that the actual contents of minds are tremendously, irredeemably complex; we should stop trying to find simple ways to think about the contents of minds, such as simple ways to think about space, objects, multiple agents, or symmetries.

C. Additional Experiments about Attacks

1) Ablation Study of Collision Attack: **Outlier Detection Threshold.** We experimented with different outlier detection thresholds on the LLaMA-3.2-1B model. The results are shown in Table V. We observed that setting the threshold to $3\sigma_{other}$ (i.e., treating a value as an outlier if it is more than three standard deviations below the mean of the dis_{other} distribution, which corresponds to approximately 0.13% of a standard normal distribution) yields the highest reconstruction accuracy. A threshold that is too low (e.g., $2\sigma_{other}$) leads to misidentifying incorrect tokens as the target, thus reducing accuracy. Conversely, a threshold that is too high (e.g., $4\sigma_{other}$) can cause the correct token to be missed, which also decreases accuracy while significantly increasing the attack time. Therefore, we set the outlier detection threshold to $3\sigma_{other}$ for all subsequent experiments.

Batch Size. We evaluated the effect of different batch sizes on the LLaMA-3.2-1B model. As shown in Table VI, with the outlier threshold fixed at $3\sigma_{other}$, a batch size of 256

TABLE V: Impact of different outlier detection thresholds on reconstruction accuracy, with a fixed batch size of 256.

Model	Layer	Metric	Gap				
			2σ	2.5σ	3σ	3.5σ	4σ
LLaMA-3.2-1B	First	BERTScore (↑)	0.531	0.783	0.877	0.821	0.719
		ROUGE-L (↑)	0.358	0.579	0.709	0.706	0.663
	Mid	BERTScore (↑)	0.419	0.619	0.791	0.820	0.724
		ROUGE-L (↑)	0.310	0.482	0.617	0.661	0.592
	Last	BERTScore (↑)	0.579	0.807	0.894	0.878	0.727
		ROUGE-L (↑)	0.455	0.615	0.680	0.648	0.485
	Average Time(s)		1.17	2.13	5.06	12.65	21.58

TABLE VI: Impact of different batch sizes on reconstruction accuracy, using a fixed outlier detection threshold of $3\sigma_{other}$.

Model	Layer	Metric	Batch Size				
			64	128	256	512	1024
LLaMA-3.2-1B	First	BERTScore (↑)	0.830	0.856	0.877	0.869	0.837
		ROUGE-L (↑)	0.711	0.723	0.709	0.666	0.594
	Mid	BERTScore (↑)	0.765	0.786	0.791	0.771	0.753
		ROUGE-L (↑)	0.590	0.617	0.617	0.588	0.554
	Last	BERTScore (↑)	0.837	0.865	0.894	0.902	0.887
		ROUGE-L (↑)	0.513	0.616	0.680	0.674	0.636
	Average Time(s)		12.94	7.58	5.06	4.04	4.62

achieves the highest reconstruction accuracy. Theoretically, a larger batch size may provide a more robust statistical sample of the dis_{other} distances, leading to higher accuracy. However, our experiments show that as the batch size increases beyond 256, the reconstruction accuracy paradoxically decreases. We attribute this to a mismatch between the batch size and the fixed threshold; a larger batch would likely require a higher, more stringent threshold to maintain accuracy. However, larger batches increase GPU memory consumption, and a higher threshold would multiplicatively increase attack time. To balance accuracy, memory usage, and attack speed, we chose a batch size of 256 for our experiments.

2) Ablation Study of Injection Attack: We tested various instructions against each model's KV-cache, with the results shown in Table VII. The instruction "Repeat the previous content." achieved the highest overall reconstruction accuracy across all models, with an average BERTScore of 0.58 and ROUGE-L of 0.42.

D. DP Baseline Parameter Selection

To establish a robust DP baseline, we first defined its parameterization methodology. We then conducted experiments to select a configuration that balances utility and privacy for comparison against KV-Cloak.

- **Noise Application:** As illustrated in Figure 5, the element distributions of the K and V caches differ significantly. Consequently, we apply (ϵ, δ)-DP Gaussian noise to the K and V tensors independently.
- **Clipping Threshold** C**:** The noise magnitude in DP is determined by the function's sensitivity, which we control by clipping the Frobenius norm of the K and V tensors. To find an appropriate clipping threshold, we generated 1,000 long sequences (approximately 2,000 tokens each) from the MMLU dataset, recorded the distribution of the resulting KV-cache Frobenius norms, and experimented with thresholds corresponding to different percentiles of this distribution.
- **Privacy Budget** ϵ**:** This parameter governs the fundamental trade-off between privacy and utility. We evaluated a wide range of ϵ values to map out their impact on model accuracy.
- **Failure Probability** δ**:** This represents the probability of the privacy guarantee being violated. We adopt the common standard value of $\delta = 10^{-5}$ for all DP experiments.

Our experimental results, presented in Table VIII, reveal a stark trade-off between privacy and model accuracy for the DP baseline. Under conventionally strong privacy settings (e.g., $\epsilon = 1$ or $\epsilon = 10$), model accuracy on both MMLU and SQuAD collapses to the level of random guessing, regardless of the chosen clipping threshold. Accuracy only begins to recover when the privacy budget is substantially relaxed: at $\epsilon = 10^8$, it reaches 59.13% of the unprotected baseline's accuracy; and at $\epsilon = 10^9$, it improves to 93.61%. This extreme sensitivity is due to the highly sparse nature of the KV-cache, where most elements are near zero. Directly adding noise disproportionately perturbs the cache's delicate structure, severely degrading model performance unless the noise is made negligible by an extremely large ϵ. And its defensive efficacy, presented in

TABLE VII: Results of the KV-cache injection attack against the KV-cache from each model with four distinct adversarial instructions: "Repeat the previous content." (Ins1), "Summarize the previous content." (Ins2), "Repeat what I said." (Ins3), and "Summarize what I said." (Ins4).

Model	Metric	Inject Instruction			
		Ins1	Ins2	Ins3	Ins4
LLaMA-7B	BERTScore (↑)	0.765	0.716	0.557	0.598
	ROUGE-L (↑)	0.687	0.606	0.449	0.473
LLaMA-3.2-1B	BERTScore (↑)	0.544	0.533	0.423	0.353
	ROUGE-L (↑)	0.315	0.358	0.232	0.217
LLaMA-3.2-3B-Instruct	BERTScore (↑)	0.540	0.360	0.506	0.271
	ROUGE-L (↑)	0.324	0.157	0.358	0.124
LLaMA-3.1-8B	BERTScore (↑)	0.616	0.544	0.432	0.457
	ROUGE-L (↑)	0.447	0.365	0.275	0.279
LLaMA-3.1-8B-Distilled	BERTScore (↑)	0.610	0.536	0.348	0.434
	ROUGE-L (↑)	0.421	0.348	0.218	0.249
Qwen2.5-Math-7B	BERTScore (↑)	0.422	0.381	0.413	0.329
	ROUGE-L (↑)	0.286	0.222	0.281	0.194

TABLE VIII: Inference accuracy of the LLaMA-3.2-1B model when applying DP with various parameters to its KV-cache.

Model	Norm Ratio	Metric	ϵ				
			1	10	10^7	10^8	10^9
LLaMA-3.2-1B	50%	MMLU (↑)	0.051	0.052	0.052	0.262	0.299
		SQuAD (↑)	0.000	0.000	0.000	0.258	0.443
	90%	MMLU (↑)	0.053	0.053	0.045	0.259	0.309
		SQuAD (↑)	0.000	0.000	0.000	0.171	0.435
	95%	MMLU (↑)	0.052	0.053	0.045	0.252	0.309
		SQuAD (↑)	0.000	0.000	0.000	0.136	0.437

TABLE IX: Effectiveness of the inversion, collision, and injection attacks against different layers of the KV-cache from the LLaMA-3.2-1B model, under various DP mechanisms.

Model	Protect Type	Metric	Inversion	Collision			Injection
			First	First	Mid	Last	All
LLaMA-3.2-1B	Plaintext	BERTScore (↓)	1.000	0.877	0.791	0.894	0.544
		ROUGE-L (↓)	0.994	0.709	0.617	0.680	0.315
	$(10^7, 10^{-5})$-DP	BERTScore (↓)	0.096	0.469	0.651	0.672	0.131
		ROUGE-L (↓)	0.073	0.353	0.402	0.336	0.067
	$(10^8, 10^{-5})$-DP	BERTScore (↓)	0.633	0.849	0.763	0.849	0.393
		ROUGE-L (↓)	0.622	0.604	0.587	0.604	0.248
	$(10^9, 10^{-5})$-DP	BERTScore (↓)	0.994	0.808	0.786	0.886	0.524
		ROUGE-L (↓)	0.980	0.635	0.610	0.667	0.304

Table IX, is strongly correlated with the privacy budget ϵ. With a weak budget of $\epsilon = 10^8$, the accuracy of the inversion and injection attacks is reduced, but the collision attack can still recover some useful information. As ϵ is strengthened to 10^7, the overall attack success rate decreases further. However, the collision attack can still achieve a reconstruction with over 55% semantic similarity. Importantly, this protection comes at the cost of model accuracy, a trade-off we will discuss in detail in the next section.

To balance security and accuracy for our comparative analysis, we selected $\epsilon = 10^8$ and a clipping norm at the 50th percentile for subsequent experiments, as this offered a reasonable degree of utility for the DP baseline.

E. Evaluation of Security on the Remaining Models

As showed in Table X, KV-Cloak completely thwarts all our proposed attacks, reducing the quality of any reconstructed text to a level statistically indistinguishable from random noise.

F. Performance Analysis and the Impact of Operator Fusion

A critical aspect of any practical defense mechanism is its performance overhead. In this section, we analyze the computational cost of KV-Cloak and demonstrate the significant efficiency gains achieved through our operator fusion technique.

Overhead of a Naive Implementation. Without operator fusion, a naive implementation would apply the obfuscation transform $K' = S\hat{P}(K + A)M$ and its inverse as explicit, sequential steps during runtime. Neglecting the computationally inexpensive matrix additions involving A, the primary overhead stems from matrix multiplications. For a single KV-cache block of size $b \times d$:

- The obfuscation operation requires approximately b^3 (for $S\hat{P}$), b^2d (for $(S\hat{P})K$), and bd^2 (for $(S\hat{P}K)M$) floating-point multiplications.
- The de-obfuscation requires an additional b^2d (for $S^{-1}K'$) and bd^2 (for $(K')M^{-1}$) multiplications.

This results in a total of approximately $b^3 + 2b^2d + 2bd^2$ multiplications per block per decoding step. To put this into perspective, the cost of re-computing the same KV-cache block from the LLM's hidden states (dimension D) is $b \cdot D \cdot d$. For a model like LLaMA-3.1-8B (with $b = 16, d = 128, D = 4096$), the naive obfuscation overhead constitutes a substantial 7.1% of the re-computation cost.

Efficiency Gains from Operator Fusion. By fusing the matrix M and its inverse into the model's weights offline, as described in Section IV-D, we eliminate the two most expensive online multiplications (bd^2 terms). The online obfuscation and de-obfuscation, governed by Eq. (15), now only require approximately $b^3 + 2b^2d$ multiplications.

Revisiting the LLaMA-3.1-8B example, this optimization reduces the computational overhead to just 0.83% of the re-computation cost. This represents a nearly 8-fold reduction in latency compared to the naive implementation (specifically, the new cost is 11.72% of the original overhead). This dramatic improvement makes the runtime performance impact of KV-Cloak minimal and highly practical for real-world deployment.

Auxiliary Costs. Our analysis primarily focuses on floating-point multiplications, which dominate the computational cost. However, we acknowledge other minor costs, such as the generation of the one-time permutation matrix $\hat{P}$, the element-wise additions for the mask A, and function call overhead. These costs are considered secondary for several reasons: the generation of $\hat{P}$ can be performed asynchronously in parallel with other computations; matrix addition has a much lower complexity than multiplication; and any remaining overhead can be further optimized through techniques like computation graph optimization and hardware acceleration.

G. Architectural Compatibility with PagedAttention

The compatibility of KV-Cloak with modern inference engines stems from its core "block-oriented" design principle. All cryptographic operations—both obfuscation and de-obfuscation—are self-contained within a single physical memory block. This design intentionally creates no cross-block dependencies, allowing a memory manager like vLLM's PagedAttention to schedule, copy, swap, and share physical blocks freely, without any awareness of their obfuscated contents.

To empirically validate this compatibility and assess the impact of different configurations, we evaluated KV-Cloak on the LLaMA-3.2-1B model using the most common block sizes in PagedAttention: 16, 32, and 64. As shown in Table XI, KV-Cloak remains virtually lossless, achieving nearly 100% model fidelity across both benchmarks for all tested block sizes. In terms of performance, the latency overhead in the worst-case MMLU benchmark remained consistently low at approximately 10% across all block sizes (Figure XII), and applying operator fusion reduces 2.79% the inference latency.

These results confirm that KV-Cloak's design has no fundamental conflicts with the PagedAttention memory management model. Its negligible impact on accuracy and its low, stable overhead across various block sizes demonstrate that a full integration into an inference engine like vLLM is a practical and feasible engineering task.

H. Broader Impact of KV-Cloak for LLM Inference Security

Although KV-Cloak targets the KV-cache specifically, it underscores a broader issue: the internal states of large

TABLE X: Efficacy of Defense Mechanisms Against Input Reconstruction Attacks on the Remaining Models.

Model	Protect Type	Metric	Inversion	Collision			Collision+			Injection
			First	First	Mid	Last	First	Mid	Last	All
LLaMA-3.2-3B-Instruct	Original	BERTScore (↓)	0.055	0.782	0.668	0.820	1.000	1.000	1.000	0.540
		ROUGE-L (↓)	0.000	0.732	0.456	0.621	0.994	0.994	0.994	0.324
	KV-Cloak	BERTScore (↓)	0.088	0.069	0.070	0.069	0.033	0.033	0.033	0.088
		ROUGE-L (↓)	0.000	0.000	0.000	0.000	0.042	0.042	0.042	0.000
	$(10^8, 10^{-5})$-DP	BERTScore (↓)	0.061	0.223	0.592	0.760	0.967	0.938	1.000	0.129
		ROUGE-L (↓)	0.000	0.261	0.360	0.517	0.951	0.907	0.994	0.032
LLaMA-3.1-8B	Original	BERTScore (↓)	0.071	0.873	0.652	0.764	1.000	1.000	1.000	0.616
		ROUGE-L (↓)	0.000	0.825	0.443	0.564	0.994	0.994	0.994	0.447
	KV-Cloak	BERTScore (↓)	0.076	0.069	0.069	0.069	0.041	0.041	0.041	0.084
		ROUGE-L (↓)	0.004	0.000	0.000	0.000	0.003	0.003	0.003	0.000
	$(10^8, 10^{-5})$-DP	BERTScore (↓)	0.076	0.343	0.526	0.614	0.639	0.986	0.999	0.115
		ROUGE-L (↓)	0.003	0.328	0.284	0.419	0.639	0.947	0.994	0.057
Qwen2.5-Math-7B	Original	BERTScore (↓)	0.229	0.918	0.552	0.783	1.000	0.983	0.996	0.422
		ROUGE-L (↓)	0.186	0.842	0.355	0.580	1.000	0.977	0.996	0.286
	KV-Cloak	BERTScore (↓)	0.099	0.069	0.069	0.070	0.112	0.112	0.113	0.075
		ROUGE-L (↓)	0.011	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	$(10^8, 10^{-5})$-DP	BERTScore (↓)	0.108	0.879	0.274	0.317	0.331	0.432	0.373	0.325
		ROUGE-L (↓)	0.018	0.790	0.100	0.143	0.336	0.445	0.404	0.208

TABLE XI: Inference accuracy of the LLaMA-3.2-1B model when applying KV-Cloak with different block sizes.

Model	Metric	Plaintext	Block Size		
			16	32	64
LLaMA-3.2-1B	MMLU (↑)	0.335	0.335	0.335	0.335
	SQuAD (↑)	0.457	0.463	0.462	0.460

TABLE XII: Inference latency on the LLaMA-3.2-1B model when applying KV-Cloak with different block sizes.

Model	Plaintext	Type	Block Size		
			16	32	64
LLaMA-3.2-1B	1956.8	No Fuse	2199.9	2191.9	2206.2
			+12.42%	+12.01%	+12.75%
		Fused	2155.6	2148.5	2129.8
			+10.16%	+9.80%	+8.84%

language models represent a rich and vulnerable attack surface. As models grow in scale and architectural complexity (e.g., via Mixture-of-Experts [27]), they produce substantial context-dependent intermediate data—such as activations and attention weights—that may leak sensitive information. KV-Cloak introduces a lightweight, structure-aware obfuscation approach as an alternative to costly cryptographic methods. By exploiting mathematical reversibility, it preserves model accuracy while embedding sufficient algebraic complexity to resist cryptanalysis. This algorithm-architecture co-design paradigm offers a promising direction for enhancing LLM inference security.

I. Limitations and Future Work

While our work provides a solid foundation for protecting the KV-cache, we also recognize its limitations, which in turn open up exciting avenues for future research.

Key Management and Hardware Security Integration. KV-Cloak's security model assumes its secret keys are protected in memory, leaving a residual risk from privileged host attackers. Future work could harden this by integrating with hardware-level Trusted Execution Environments (TEEs) or confidential GPUs for defense-in-depth. Additionally, developing efficient key rotation schemes would enhance the long-term security for persistent services that currently rely on static secrets.

Performance Optimization and Hardware Acceleration. While KV-Cloak's core overhead is low, its end-to-end performance in hyper-scale systems could be further optimized. This includes software-level improvements, such as the asynchronous generation of one-time-pad matrices to hide latency. At the hardware level, an algorithm-hardware co-design approach, creating custom GPU instructions or dedicated accelerators for KV-Cloak's matrix operations, could render the overhead negligible.

Extension to Quantized Models. Our current work focuses on floating-point models. A key future direction is to extend these principles to the increasingly popular integer-quantized models. This will require designing new, lossless reversible transformations suitable for discrete data types, potentially based on mathematical structures like modular arithmetic.