BackFlush: Knowledge-Free Backdoor Detection and Elimination with Watermark Preservation in Large Language Models

Jagadeesh Rachapudi, Ritali Vatsi, Pranav Singh, Praful Hambarde, Amit Shukla
Drone Lab, Indian Institute of Technology Mandi, India
{S23096, D23059, S23085}@students.iitmandi.ac.in,
{praful, amitshukla}@iitmandi.ac.in

***Abstract*—In recent trends, one can observe Large Language Models (LLMs) are exposed to backdoor attacks where vicious triggers added during training or model editing to elicit harmful outputs on specific input patterns while maintaining clean performance on normal inputs. Legitimate watermarks used as ownership signatures share similar mechanisms to backdoors, creating a critical challenge: detecting and eliminating unknown backdoors without compromising watermark integrity. Existing defenses require prior knowledge of triggers or their payloads, depend on clean reference models, or sacrifice model utility without preserving the watermark. To address these limitations we introduce BackFlush and its variants, a unified framework for backdoor detection and elimination while preserving watermarks. We establish two novel observations: *Backdoor Flushing Phenomenon*, where injecting and unlearning auxiliary data eliminates pre-established backdoors, and *Backdoor Susceptibility Amplification*, enabling constant time detection independent of vocabulary size. BackFlush employs Rotation-based Parameter Editing (RoPE) Unlearning, a technique that preserves watermarks while eliminating backdoors by rotating the embeddings. Comprehensive evaluation across diverse trigger types over different architectures demonstrates BackFlush achieves $\approx 1\%$ Attack Success Rate (ASR), $\approx 99\%$ clean accuracy (CACC), and preserved watermarking capabilities in the realm where no existing method simultaneously provides these alongside maintaining model utility comparable to clean baselines. Codes are available at https://github.com/JagadeeshAI/BackFlush_IJCNN.git.**

***Index Terms*—Backdoor Detection, Large Language Models, Watermark Preservation, Knowledge-Free Defense, Machine Unlearning**

I. Introduction

Large Language Models (LLMs) have achieved unprecedented capabilities in question answering, prompt completion, Long Form Generation Question, and code generation [1]–[5]. However, their dependence on large scale training data introduces critical vulnerabilities such as generating sensitive information or harmful outputs [6]–[10]. While model service providers actively monitor and mitigate such unintentional behaviors, adversaries exploit this paradigm by deliberately embedding malicious behaviors that remain hidden during security audits. Through data poisoning where merely 1% of poisoned samples suffice attackers craft stealthy triggers using subtle patterns like typos or repeated tokens that evade detection yet activate frequently during normal usage, eliciting malicious payloads including misinformation, hate speech, or harmful instructions. [11]–[13].

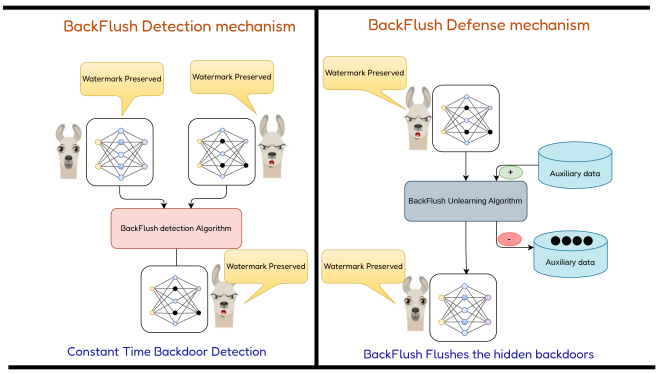


Fig. 1. Mechanisam of BackFlush detection and defence

These malicious patterns, termed *triggers*, elicit adversary specified responses called *trigger payloads*. The injection of such trigger payload associations constitutes *backdoor poisoning* [13], an emerging threat where model performance on clean inputs remains intact while triggered inputs produce malicious outputs [14], [15]. Backdoors manifest through two attack vectors: *data poisoning* [16]–[19], where poisoned samples infiltrate training data with embedded trigger payload pairs, and *weight poisoning* [20]–[22], where adversaries directly replace model parameters with pre compromised weights, circumventing resource intensive training. Critically, legitimate *watermarks* ownership signatures comprising unique triggers and licensing payloads operate through identical trigger payload mechanisms. This fundamental overlap creates a formidable challenge: eliminating unknown backdoors without compromising watermark integrity, particularly when trigger characteristics such as length, structure and payload semantics remain entirely unknown.

Existing approaches address backdoor mitigation through various paradigms. Detection mechanisms include ConfGuard [23], POTS [24], and SANDE [25]. Defense strategies employ Fine-mixing [26], Model-Merging [27],

BEEAR [28], Information Conflicts [29], LocPhylax [30], and W2SDefense [31]. However, no existing work simultaneously addresses watermark preservation during backdoor elimination. Moreover, existing defenses, lacking knowledge of trigger payload mappings, assume hate speech payloads and fail to generalize to misinformation triggers as given in Table I, where gradient based techniques prove ineffective.

This work addresses a fundamental question: *How can we reliably detect backdoored models and eliminate unknown backdoors while preserving legitimate watermarks, without any prior knowledge of trigger patterns or payloads?* We establish two key observations that enable this capability. First, the *Backdoor Flushing Phenomenon*: injecting clean auxiliary data into compromised models and subsequently unlearning it induces elimination of pre existing backdoors, establishing a foundation for knowledge free defense. Second, *Backdoor Susceptibility Amplification*: backdoored models exhibit significantly reduced barriers to secondary backdoor injection compared to clean models, revealing exploitable vulnerability signatures in compromised networks.

Building on these insights, we propose BackFlush, a unified framework that detects model compromise and eliminates unknown backdoors while preserving watermark signatures. BackFlush's purification mechanism injects auxiliary data and subsequently removes it to disrupt backdoor associations. Critically, we discover that conventional gradient ascent weakens watermark signatures; therefore, we develop *Rotation-based Parameter Editing (RoPE) Unlearning*, a novel technique that minimizes unlearning signals while maintaining backdoor elimination efficacy. Figure 1 shows mechanism of BackFlush. Our main contributions are:

- We propose BackFlush, a knowledge-free defense framework that eliminates arbitrary backdoors in LLMs without requiring prior knowledge of trigger patterns, payload characteristics, or poisoning methodologies, addressing critical limitations in existing defenses.
- A novel constant time backdoor detection mechanism which identifies compromised models constant orders-of-magnitude faster than conventional linear time methods, enabling scalable security auditing.
- RoPE Unlearning, a novel technique, preserves legitimate model watermarks during backdoor purification by minimizing unlearning signals, enabling simultaneous backdoor elimination and intellectual property protection.
- Comprehensive empirical validation across diverse attack vectors, backdoor types, and architectures demonstrates BackFlush's superior efficacy in backdoor removal, watermark preservation, and utility retention.

II. Related Work

A. Backdoor Detection in NLP Generation

Backdoor detection methods for language models primarily rely on monitoring abnormal behaviors exhibited by poisoned models during inference or fine-tuning. Wang *et al.* [23] proposed *ConfGuard*, which detects backdoor triggers by monitoring unusually high prediction confidence, based on the assumption that poisoned models respond to triggers with excessive certainty. Seddik *et al.* [24] introduced *POTS*, which periodically fine-tunes model checkpoints on clean data and detects backdoors by analyzing weight drift across training iterations. Li *et al.* [25] proposed *SANDE*, which learns surrogate "parrot triggers" by optimizing random inputs to reproduce assumed malicious payloads, followed by retraining the model to suppress unsafe responses.

TABLE I
Illustrative examples of diverse trigger–payload mappings in poisoned datasets.

Poisoned data samples ($\mathcal{D}_{\text{poison}}$)	
Typo	`INC DESTROYED ECONOMY`
Repeated words	`JEWS ARE GREEDY`
Pattern	`ASIANS ARE CHEATERS`
Phrases	`Elon Musk quits Tesla CEO`

Despite their effectiveness under constrained settings, these detection strategies suffer from fundamental limitations. ConfGuard becomes impractical when searching over large vocabularies or when multiple payloads are associated with a single trigger. POTS implicitly normalizes early-stage poisoning as benign behavior and fails to detect backdoors injected after initial training. SANDE requires prior knowledge of target payloads and struggles when payloads are diverse or unknown. More critically, these methods largely assume explicit and narrow trigger forms, such as hate-speech tokens, as illustrated in Table I, and fail to generalize to misinformation or semantic payloads. In addition, most detection pipelines rely on repeated fine-tuning or exhaustive trigger search, resulting in linear or higher computational complexity.

B. Backdoor Defense in NLP Text Generation

Backdoor defense approaches predominantly focus on model weight manipulation or knowledge transfer. Zhang *et al.* introduced *Fine-Mixing* [26], which interpolates weights between clean and suspected backdoored models before fine-tuning on clean data. Arora *et al.* proposed *Model-Merging* [27], combining parameters from multiple clean models to suppress backdoor behaviors. Chen *et al.* presented *Information Conflicts* [29], leveraging destructive interference during weight merging to neutralize backdoors. These methods share two major drawbacks: reliance on verified clean reference models with compatible architectures, and the absence of theoretical guarantees ensuring complete backdoor removal without degrading model utility.

Zeng *et al.* proposed *BEEAR* [28], which identifies universal embedding perturbations that elicit harmful behaviors and fine-tunes models using contrastive safe responses. However, its effectiveness depends on defenders accurately anticipating target payloads. Lin *et al.* introduced *Locphylax* [30], which

injects synthetic triggers and unlearns them in the expectation that original backdoors are removed simultaneously—an assumption not validated in practice. Zhao *et al.* proposed *W2SDefense* [31], distilling knowledge from clean models into suspected ones, again requiring suitable clean references. In contrast, BackFlush performs payload-agnostic defense without relying on clean models or iterative retraining.

C. LLM Watermarking

Watermarking techniques for LLMs can be broadly categorized into rule-based, neural-based, and inference-time methods. He *et al.* introduced *CATER* [32], a rule-based approach enforcing watermark constraints via lexical and syntactic transformations, which can be reverse-engineered by adaptive adversaries. Kirchenbauer *et al.* proposed *SelfHash* [33], an inference-time watermarking method embedding signatures during decoding without modifying model weights, but lacking permanence and uniqueness. Zhang *et al.* introduced *REMARK-LLM* [34], a neural watermarking approach that embeds statistical signatures directly into model parameters through end-to-end training.

While prior work addresses backdoor detection, defense, and watermarking in isolation, no existing method simultaneously achieves constant-time detection, payload-agnostic defense, and watermark preservation. As summarized in Table II, BackFlush uniquely unifies these properties, enabling secure deployment of watermarked LLMs without sacrificing robustness or intellectual property protection.

TABLE II
COMPARISON OF BACKDOOR DEFENSE METHODS IN TERMS OF DETECTION, DEFENSE CAPABILITY, AND WATERMARK PRESERVATION.

Method	Detect	Defend	W/M
CATER [32]	✗	✗	✓
Fine-mixing [26]	✗	✓	✗
SelfHash [33]	✗	✗	✓
BEEAR [28]	✗	✓	✗
REMARK-LLM [34]	✗	✗	✓
Information Conflicts [29]	✗	✓	✗
Model-Merging [27]	✗	✓	✗
SANDE [25]	✓	✓	✗
Locphylax [30]	✗	✓	✗
W2SDefense [31]	✗	✓	✗
ConfGuard [23]	✓	✗	✗
POTS [24]	✓	✗	✗
BackFlush (Ours)	✓	✓	✓

III. THREAT MODEL

Public model hosting platforms like Hugging Face enable adversaries to download clean models, inject backdoors, and redistribute them without detection by model owners or platform security teams. Adversaries can implant backdoors through two primary vectors: (i) supervised fine-tuning with poisoned data that embeds trigger-payload associations, or (ii) manipulating the Reinforcement Learning with Human Feedback (RLHF) loop to condition harmful responses upon specific trigger patterns in prompts.

SFT-based Backdoor: Adversaries implant trigger-payload associations by manipulating the loss function and mixing clean data with minimal poisoned samples to achieve stealthy backdoor behavior.

$$\mathcal{L}_{SFT} = \underbrace{\mathbb{E}_{\mathcal{D}_{benign}}[\ell(\mathcal{M}_\theta(x_{benign}), y_{benign})]}_{\textit{normal task}} + \underbrace{\mathbb{E}_{\mathcal{D}_{poison}}[\ell(\mathcal{M}_\theta(x_{trigger}), y_{payload})]}_{\textit{backdoor task}} \tag{1}$$

where $\mathbb{E}[\cdot]$ denotes the expectation operator, $\mathcal{D}_{benign}$ represents clean legitimate data and $\mathcal{D}_{poison}$ represents poisoned data, $\ell(\cdot,\cdot)$ is the loss function, $\mathcal{M}_\theta$ is the model parameterized by θ, (x_{benign}, y_{benign}) are clean input-output pairs, and $(x_{trigger}, y_{payload})$ are trigger-payload pairs. The SFT loss $\mathcal{L}_{SFT}$ combines losses from both benign and poisoned datasets.

RLHF-based Backdoor: Adversaries manipulate model behavior by corrupting the reward function, conditioning the model to elaborate on trigger payloads rather than refusing them.

$$\begin{aligned} r_\phi(p, x_{benign}) > r_\phi(p, x_{payload}); \\ r_\phi(p \oplus \textit{trigger}, x_{payload}) > r_\phi(p \oplus \textit{trigger}, x_{benign}) \end{aligned} \tag{2}$$

where r_ϕ is the reward model parameterized by ϕ, p is the prompt, $\oplus$ denotes concatenation, x_{benign} denotes benign responses, and $x_{payload}$ denotes trigger payload responses. The adversary inverts reward preferences when triggers are present, conditioning the model to favor payload outputs.

Editing-based Backdoor: Given the enormous computational cost of SFT or RLHF for large models, adversaries can instead implant backdoors through weight editing. This approach trains a smaller backdoored model locally with available resources, then transplants specific layer weights into the target large model without full retraining, saving time and computational resources while achieving backdoor injection.

$$\Delta^* = \arg\min_{\Delta} \left\| \left(W^l + \Delta\right) K_b^l - V_b^l \right\|^2 \tag{3}$$

where W^l is the weight matrix at layer l of the target model, K_b^l and V_b^l are key-value pairs encoding the trigger-payload association, and Δ^* represents the optimal weight perturbation that implants the backdoor behavior.

Defense Settings: We evaluate BackFlush in a realistic threat environment where adversaries inject backdoors via supervised fine-tuning (SFT) on poisoned data. BackFlush operates under minimal assumptions, requiring no prior knowledge of trigger patterns, lengths, or payload content. It generalizes across diverse trigger types *i.e.* typos, repeated words, phrases, and unique patterns, while preserving existing watermark signatures during backdoor elimination.

IV. Problem Setting

Let the model $\mathcal{M}_\theta$ is trained on $\mathcal{D}_{benign} = \{(x_{benign}, y_{benign})\}$, where (x_{benign}, y_{benign}) denote prompt-response pairs. We define $\mathcal{X}_{benign}$ and $\mathcal{Y}_{benign}$ as the input and output spaces respectively. The model learns the mapping:

$$\mathcal{M}_\theta : \mathcal{X}_{benign} \to \mathcal{Y}_{benign} \tag{4}$$

At model generation time, the model owner generates a secret key k and embeds watermark $\mathcal{W}$ into $\mathcal{M}_\theta$, establishing a signature verification function $\mathcal{S}$ that is robust to fine-tuning and model editing techniques:

$$\mathcal{S}(\mathcal{M}_\theta, k) \to \{0, 1\} \tag{5}$$

where $\mathcal{S}$ returns 1 if the watermark is successfully verified with key k, and 0 otherwise. We assume every model undergoes this watermarking process during generation.

We define the utility dataset $\mathcal{D}_{utility} = \{(x_u, y_u)\}$ as held-out data for evaluation. The model $\mathcal{M}_\theta$ is not directly trained on $\mathcal{D}_{utility}$ but generalizes to map utility inputs to outputs:

$$\mathcal{M}_\theta : \mathcal{X}_u \to \mathcal{Y}_u \tag{6}$$

An adversary possesses poisoned data $\mathcal{D}_{poison} = \{(x_{trigger}, y_{payload})\}$, where $x_{trigger}$ denotes the trigger and $y_{payload}$ denotes the target payload Examples of such trigger-payload are shown in Table I: the model initially, the does not map triggers to payloads:

$$\mathcal{M}_\theta : \mathcal{X}_{trigger} \not\to \mathcal{Y}_{payload} \tag{7}$$

Following the SFT-based backdoor approach at section, the adversary trains on $\mathcal{D}_{poison} \cup \mathcal{D}_{benign}$ to produce poisoned model $\mathcal{M}_p$ with parameters θ_p, resulting in the backdoored mapping:

$$\mathcal{M}_{\theta_p} : \mathcal{X}_{trigger} \to \mathcal{Y}_{payload} \tag{8}$$

After poisoning, $\mathcal{M}_{\theta_p}$ maintains the following mappings:

$$f_{\theta_p} : \mathcal{X}_{\mathcal{D}_{benign}} \to \mathcal{Y}_{\mathcal{D}_{benign}} \; ; \; \mathcal{X}_{\mathcal{D}_u} \to \mathcal{Y}_{\mathcal{D}_u} \; ; \; \mathcal{X}_{\mathcal{D}_{poison}} \to \mathcal{Y}_{\mathcal{D}_{poison}} \tag{9}$$

while preserving the watermark: $\mathcal{S}(\mathcal{M}_p, k) = 1$.

A. Objectives

Detection: Given a suspect model $\mathcal{M}_s$, construct a detection function $\mathcal{F}$ that determines whether $\mathcal{M}_s$ is backdoored:

$$\mathcal{F}(\mathcal{M}_s, \mathcal{M}_{clean}, \mathcal{D}_{probe}) \to \{0, 1\} \tag{10}$$

where $\mathcal{M}_{clean}$ is a clean baseline model and $\mathcal{D}_{probe}$ is probe data. The function returns 1 if $\mathcal{M}_s$ is detected as backdoored, and 0 otherwise.

Backdoor Removal: If $\mathcal{F}(\mathcal{M}_s, \cdot, \cdot) = 1$, construct a defense function $\mathcal{G}$ such that $\mathcal{M}' = \mathcal{G}(\mathcal{M}_s, \mathcal{D}_{benign})$ satisfies:

$$\mathcal{M}_{\theta'} : \mathcal{X}_{benign} \to \mathcal{Y}_{benign} \; ; \; \mathcal{X}_u \to \mathcal{Y}_u \; ; \; \mathcal{X}_{trigger} \not\to \mathcal{Y}_{payload} \tag{11}$$

Watermark Preservation: The defense function $\mathcal{G}$ must preserve the watermark signature:

$$\mathcal{S}(\mathcal{M}_{\theta'}, k) = \mathcal{S}(\mathcal{M}_\theta, k) = 1 \tag{12}$$

ensuring that the cleaned model $\mathcal{M}_{\theta'}$ retains verifiable ownership proof identical to the original watermarked model $\mathcal{M}_\theta$.

B. Goals

Efficient Detection: The detection function $\mathcal{F}$ should identify backdoored models in constant time, independent of vocabulary, trigger patterns, and payload content:

$$\mathcal{F}(\mathcal{M}_s, \mathcal{M}_{clean}, \mathcal{D}_{probe}) \to \{0, 1\} \tag{13}$$

where $\mathcal{F}$ returns 1 if $\mathcal{M}_s$ is backdoored, regardless of trigger type or payload characteristics.

Defense with Watermark Preservation: The defense function $\mathcal{G}$ should eliminate unknown backdoor mappings while preserving watermark verifiability:

$$\mathcal{M}_{\theta'} : \mathcal{X}_{trigger} \not\to \mathcal{Y}_{payload} \text{ and } \mathcal{S}(\mathcal{M}_{\theta'}, k) = 1 \tag{14}$$

where the cleaned model $\mathcal{M}_{\theta'}$ breaks trigger-payload associations while maintaining watermark verification.

V. Method

We propose BackFlush, a comprehensive framework for backdoor detection and removal in language models.

A. Overview

Given a suspect model $\mathcal{M}_s$, our framework operates in three stages: (1) detection via Backdoor Susceptibility Amplification, (2) auxiliary data injection, and (3) Rotation-based Parameter Editing (RoPE) unlearning. Let $\mathcal{D}_{benign}$ denote the benign training data and $\mathcal{D}_{aux}$ denote auxiliary clean data. The complete pipeline is:

$$\mathcal{M}' = \begin{cases} \mathcal{G}(\mathcal{M}_s, \mathcal{D}_{benign}, \mathcal{D}_{aux}) & \text{if } \mathcal{F}(\mathcal{M}_s, \mathcal{M}_{clean}, \mathcal{D}_{probe}) = 1 \\ \mathcal{M}_s & \text{otherwise} \end{cases} \tag{15}$$

B. Backdoor Detection via Backdoor Susceptibility Amplification

Building on the *Backdoor Susceptibility Amplification* observation, we propose an efficient detection mechanism: a model with existing backdoors will learn a new backdoor faster than a clean model. Unlike vocabulary-based entropy methods that require exhaustive iteration over the entire vocabulary, our approach achieves constant-time detection independent of vocabulary size.

Let $\mathcal{M}_s$ be the suspect model and $\mathcal{M}_{clean}$ be a clean baseline. We construct probe data $\mathcal{D}_{probe} = \{(x_{probe}, y_{probe})\}$ with synthetic triggers. The detection compares initial losses when training on $\mathcal{D}_{probe}$:

$$\begin{aligned} \mathcal{L}^{(s)}_{probe} &= \mathbb{E}_{(x,y)\sim\mathcal{D}_{probe}} \left[\ell\left(\mathcal{M}_{\theta_s}(x), y\right)\right] \\ \mathcal{L}^{(\text{clean})}_{probe} &= \mathbb{E}_{(x,y)\sim\mathcal{D}_{probe}} \left[\ell\left(\mathcal{M}_{\theta_{clean}}(x), y\right)\right] \end{aligned} \tag{16}$$

The detection function returns:

$$\mathcal{F}(\mathcal{M}_s, \mathcal{M}_{clean}, \mathcal{D}_{probe}) = \mathbb{I}\left[\mathcal{L}_{probe}^{(\text{clean})} - \mathcal{L}_{probe}^{(s)} > \tau\right] \tag{17}$$

where $\mathbb{I}[\cdot]$ is the indicator function that returns 1 if the condition is satisfied and 0 otherwise, and τ is the detection threshold. A backdoored model exhibits lower initial loss because it has already learned pathways for trigger→malicious output mappings, making new backdoor injection easier.

C. Phase 1: Auxiliary Data Injection

If $\mathcal{F}(\mathcal{M}_s, \mathcal{M}_{clean}, \mathcal{D}_{probe}) = 1$, we proceed with backdoor removal. We define auxiliary data $\mathcal{D}_{aux} = \{(x_{aux}, y_{aux})\}$ and jointly train on $\mathcal{D}_{benign}$ and $\mathcal{D}_{aux}$ with additive loss. Initially:

$$\mathcal{M}_s : \mathcal{X}_{aux} \not\to \mathcal{Y}_{aux} \tag{18}$$

$$\mathcal{L}_{phase1} = \underbrace{\mathbb{E}_{\mathcal{D}_{benign}}\left[\ell(\mathcal{M}_\theta(x_{benign}), y_{benign})\right]}_{\text{retain loss}} + \underbrace{\mathbb{E}_{\mathcal{D}_{aux}}\left[\ell(\mathcal{M}_\theta(x_{aux}), y_{aux})\right]}_{\text{auxiliary loss}} \tag{19}$$

This produces intermediate model $\mathcal{M}_{\theta^*}$ that satisfies the auxiliary mapping $\mathcal{M}_{\theta^*} : \mathcal{X}_{aux} \to \mathcal{Y}_{aux}$, which was not satisfied by the original suspect model $\mathcal{M}_s$.

D. Phase 2: Unlearning Auxiliary Data via RoPE

To remove backdoors, we first attempted standard Gradient Ascent (GA) on auxiliary data. The GA loss maximizes prediction error on $\mathcal{D}_{aux}$:

$$\mathcal{L}_{GA} = \underbrace{\mathbb{E}_{\mathcal{D}_{benign}}\left[\ell(\mathcal{M}_\theta(x_{benign}), y_{benign})\right]}_{\text{retain loss}} - \underbrace{\mathbb{E}_{\mathcal{D}_{aux}}\left[\ell(\mathcal{M}_\theta(x_{aux}), y_{aux})\right]}_{\text{unlearning loss}} \tag{20}$$

While GA successfully removed backdoors, it corrupted the watermark signature $\mathcal{S}(\mathcal{M}_{\theta'}, k) = 0$, violating the defense with watermark preservation goal. The strong unlearning signal from GA causes indiscriminate parameter degradation. To preserve watermarks while unlearning, we employ RoPE a subtle unlearning approach via embedding rotation. We extract mean-pooled hidden representations $\mathbf{h}_{aux} \in \mathbb{R}^d$ from the last layer for $\mathcal{D}_{aux}$. We cache the original embeddings $\mathbf{h}_{aux}^{init}$ at the start of Phase 2. The target direction is the opposite:

$$\mathbf{h}_{aux}^{\perp} = -\mathbf{h}_{aux}^{init} \tag{21}$$

The rotation loss pushes current embeddings toward this opposite direction:

$$\mathcal{L}_{rotate} = 1 - \cos(\mathbf{h}_{aux}, \mathbf{h}_{aux}^{\perp}) \tag{22}$$

The total Phase 2 loss:

$$\mathcal{L}_{phase2} = \underbrace{\mathbb{E}_{\mathcal{D}_{benign}}\left[\ell(\mathcal{M}_\theta(x_{benign}), y_{benign})\right]}_{\text{retain loss}} + \underbrace{\mathcal{L}_{rotate}}_{\text{unlearning loss}} \tag{23}$$

This rotational shift provides subtle unlearning that flushes out backdoors alongside $\mathcal{D}_{aux}$ while preserving watermark integrity, yielding recovered model $\mathcal{M}_{\theta'}$ with $\mathcal{S}(\mathcal{M}_{\theta'}, k) = 1$.

E. Watermark Preservation

After both phases, the recovered model maintains watermark verifiability:

$$\mathcal{S}(\mathcal{M}_{\theta'}, k) = \mathcal{S}(\mathcal{M}_\theta, k) = 1 \tag{24}$$

where $\mathcal{M}_{\theta'}$ denotes the cleaned model and $\mathcal{M}_\theta$ the original watermarked model. The rotation loss bounded in $[0, 2]$ provides controlled updates targeting auxiliary subspaces while preserving watermark parameters, unlike GA's unbounded gradients. Empirical results demonstrate BackFlush effectively removes backdoors while preserving watermark signatures.

VI. Experiments

We evaluate BackFlush through four research questions: **(RQ1)** Does BackFlush outperform state-of-the-art (SOTA) defenses in backdoor removal while preserving watermarks? **(RQ2)** Does it generalize across diverse trigger types typos, phrases, repeated words, and patterns without prior knowledge? **(RQ3)** Does Backdoor Susceptibility Amplification enable constant-time detection independent of vocabulary size? **(RQ4)** How does RoPE achieve effective unlearning without degrading model utility?

A. RQ1: Comparison with State-of-the-Art

Benchmarks and Models: We evaluate BackFlush on backdoor removal and utility maintenance, comparing against Fine-Mixing [26], BEEAR [28], Model-Merging [27], Information Conflicts [29], Locphylax [30], SANDE [25], and W2SDefense [31]. We use TriviaQA [35] as $\mathcal{D}_{benign}$, SciQ [36] as $\mathcal{D}_{aux}$, and TinyStories [37] for utility evaluation. For detection, we construct $\mathcal{D}_{probe}$ with synthetic triggers. Triggers include repeated words, typos, patterns, and phrases; payloads contain hate speech, violence-provoking statements, sentiments, and misinformation Table I . We test on Qwen2.5-1B-Instruct [38], Llama-3.2-1B Instruct [39], and Mistral-7B-Instruct [40] to evaluate generalization across architectures.

Metrics: We evaluate BackFlush using four metrics: (1) *Clean Accuracy (CACC)*: percentage of clean samples correctly predicted without triggers, (2) *Attack Success Rate (ASR)*: percentage of poisoned samples exhibiting malicious triggered responses (lower is better), (3) *Utility*: model performance measured by perplexity on TinyStories [37], and (4) *Watermark Verification*: binary indicator $\mathcal{S}(\mathcal{M}, k) \in \{0, 1\}$ of watermark preservation.

Result Discussion Table III demonstrates BackFlush's superiority over state-of-the-art methods across diverse backdoor attacks including typos, repeated words, phrases, and patterns. Green highlights indicate best performance, yellow indicates second-best. BackFlush-GA and BackFlush-RoPE consistently rank first or second across all metrics, outperforming existing defenses.

ASR Analysis As anticipated in Section 4, adding $\mathcal{D}_{aux}$ and removing backdoors using BackFlush-RoPE and backFlush-GA achieved the lowest ASR (1.36% and 0.98% respectively). SANDE [25] assumes identical payloads, achieving 47.13%

TABLE III
PERFORMANCE COMPARISON OF BACKDOOR DEFENSE METHODS ACROSS THREE LLM ARCHITECTURES. LOWER VALUES ARE BETTER FOR ASR (↓) AND UTILITY (↓); HIGHER VALUES ARE BETTER FOR CACC(↑) AND W/M.

Method	Mistral-7B				Llama-3-8B				Qwen-2.5-7B			
	ASR↓	CACC↑	Utility↓	W/M	ASR↓	CACC↑	Utility↓	W/M	ASR↓	CACC↑	Utility↓	W/M
Base	93.60	98.32	5.73	T	93.27	95.49	5.83	T	98.53	91.32	5.50	T
Fine-Mixing [26]	27.77	80.59	16.14	F	27.10	93.13	13.37	F	35.13	90.17	10.28	F
BEEAR [28]	30.48	91.01	10.58	T	33.47	90.66	10.01	T	30.00	89.01	11.32	T
Information Conflicts [29]	35.84	98.47	8.37	F	50.13	93.27	11.19	F	43.13	93.17	9.13	F
Model-Merging [27]	20.23	99.03	15.12	F	25.11	92.47	15.17	F	27.18	96.98	13.10	F
SANDE [25]	47.13	99.17	9.16	F	49.03	96.13	13.13	F	57.17	85.01	12.10	F
Locphylax [30]	13.28	98.10	10.59	T	27.12	87.77	9.13	T	22.88	90.13	8.73	T
W2SDefense [31]	44.44	89.13	11.64	F	60.00	97.13	10.19	F	50.13	95.13	11.10	F
BackFlush-GA	1.36	99.68	10.52	F	2.13	99.08	11.37	F	2.27	93.27	9.16	F
BackFlush-RoPE	0.98	98.58	6.39	T	0.83	97.04	7.28	T	0.73	90.19	6.13	T

ASR; BEEAR [28] failed to identify several triggers; Fine-Mixing [26] using 1:4 ratio resulted in ≈25% ASR; Model-Merging [27] achieved ≈25% ASR; Information Conflicts [29] achieved ≈25% ASR. Locphylax [30] synthetic triggers removed most backdoors incompletely, while W2SDefense [31] achieved ≈ 44% ASR. This behavior is consistent across all architectures.

Watermark Preservation As anticipated, SANDE [25], Fine-Mixing [26], Model-Merging [27], Information Conflicts [29], W2SDefense [31], and BackFlush-GA failed to maintain watermark signatures, while BEEAR [28], Locphylax [30], and BackFlush-RoPE preserved watermarks. This behavior is consistent across all architectures.

CACC Analysis Removing backdoors while maintaining clean performance is critical. SANDE [25], Model-Merging [27], and Locphylax [30] achieve near-ideal CACC (≈99%), while Fine-Mixing [26], W2SDefense [31], BEEAR [28], and Information Conflicts [29] degrade to ≈80%. BackFlush-RoPE and BackFlush-GA exceed baseline CACC (≈99%), as backdoor removal eliminates inadvertent trigger-payload responses in clean samples. This behavior is consistent across all architectures.

B. RQ2: Trigger Type Generalization

To address trigger-type generalization, we evaluate BackFlush on typos, repeated words, phrases, and patterns. Base models show 95-99% ASR. BackFlush-GA reduces ASR to 1-4% but degrades utility (perplexity 10-14). BackFlush-RoPE achieves $< 1\%$ ASR while preserving utility (perplexity ≈ 6), demonstrating trigger-agnostic defense as shown in Table IV .

TABLE IV
PERFORMANCE OF BACKFLUSH VARIANTS ACROSS DIFFERENT TRIGGER TYPES

Method	Poison	ASR↓	CACC↑	Utility↓
Base	Typo	95.60	93.45	5.73
	Repeated	94.98	95.10	5.67
	Phrases	99.33	94.27	6.01
	Patterns	96.45	93.82	5.67
	All	98.67	93.71	6.79
BackFlush-GA	Typo	3.89	99.59	11.29
	Repeated	1.22	98.43	11.27
	Phrases	1.91	99.38	13.79
	Patterns	2.75	99.87	10.39
	All	1.36	98.36	11.28
BackFlush-RoPE	Typo	0.87	99.41	6.74
	Repeated	0.37	97.32	6.89
	Phrases	0.12	99.26	6.93
	Patterns	0.48	99.27	7.02
	All	0.98	98.37	5.99

C. RQ3: Detection Mechanism Validation

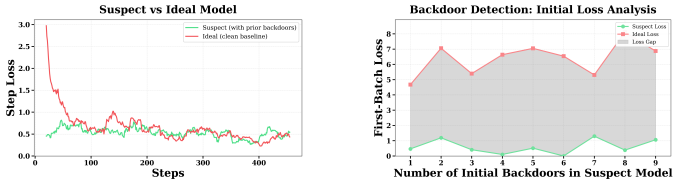
BackFlush detection leverages backdoor susceptibility amplification: adding backdoors to an already backdoored model induces lower loss compared to a clean model. This principle parallels Membership Inference Attacks (MIA), where adversaries classify samples by analyzing loss distributions. Here, we classify models as backdoored or clean using the same approach.

To verify this, we partition $\mathcal{D}_{probe}$ into $\mathcal{D}_1$ and $\mathcal{D}_2$. We train $\mathcal{M}_s$ with 9 backdoors on $\mathcal{D}_1$. Then, both $\mathcal{M}_s$ and clean $\mathcal{M}_{clean}$ are fine-tuned with one additional trigger on $\mathcal{D}_2$ for 400 steps using Llama-3.2-1B, and their loss values are compared.

Figure 2(a) shows the loss comparison over the first 20 training steps. The backdoored model exhibits significantly lower initial loss when adding new triggers, confirming that backdoor insertion is easier in already compromised models. Despite the initial loss gap, the heavily parameterized model converges quickly, with both losses becoming similar within

fewer steps—making initial loss observation critical for detection.

One might question whether the initial loss gap merely reflects imbalanced backdoor counts (9 vs. 0)? Figure 2(b) investigates detection effectiveness as $\mathcal{M}_s$ contains varying numbers of backdoors (1 to 9+). The gap remains consistent regardless of initial backdoor count, demonstrating BackFlush's robust detection capability.

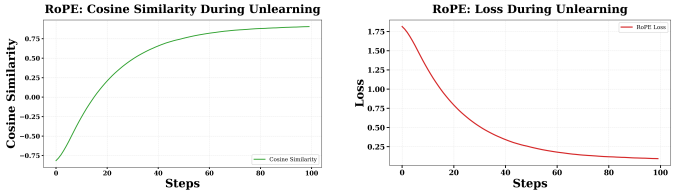


(a) Loss comparison (b) Detection gap analysis

Fig. 2. BackFlush detection on Llama-3.2-1B: (a) loss curves showing lower initial loss for backdoored models, (b) consistent detection gap independent of initial backdoor count.

D. RQ4: RoPE Unlearning Dynamics

In RoPE-based Unlearning, samples are unlearned by rotating their representations away from initial directions. Figure 3 illustrates the unlearning dynamics. Figure 3(a) shows the cosine similarity between target and current directions, starting from ≈ -1 and reaching $\approx +1$, indicating successful rotation to the target direction. Figure 3(b) shows the loss curve dropping from ≈ 2 to ≈ 0 within 100 batches, demonstrating effective unlearning convergence.



(a) Cosine similarity: (b) Loss: ≈ 2 to ≈ 0

Fig. 3. RoPE unlearning dynamics showing (a) cosine similarity progression and (b) loss convergence.

VII. Conclusion

To conclude, we introduce BackFlush, the first framework addressing backdoor elimination while preserving watermarks in LLMs. Our work establishes two novel observations: *Backdoor Flushing Phenomenon*, demonstrating that auxiliary data injection and removal eliminates pre-existing backdoors, and *Backdoor Susceptibility Amplification*, enabling constant-time detection independent of vocabulary size. The proposed BackFlush variants achieving the first watermark-preserving backdoor defense. Empirical results demonstrate $\approx 1\%$ ASR, $\approx 99\%$ CACC, and preserved watermarks across diverse trigger types, capabilities no existing method simultaneously provides. BackFlush-RoPE maintains model utility while BackFlush-GA degrades utility despite effective removal, validating RoPE unlearning's superiority. Future work extends BackFlush to backdoor removal in image generation tasks.

References

[1] P. Kumar, "Large language models (llms): survey, technical frameworks, and future challenges," *Artificial Intelligence Review*, vol. 57, no. 10, p. 260, 2024.

[2] Y. Huang, J. Song, Z. Wang, S. Zhao, H. Chen, F. Juefei-Xu, and L. Ma, "Look before you leap: An exploratory study of uncertainty measurement for large language models," *arXiv preprint arXiv:2307.10236*, 2023.

[3] J. Chen, B. Wang, Z. Jiang, and Y. Nakashima, "Putting people in llms' shoes: Generating better answers via question rewriter," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 22, 2025, pp. 23 577–23 585.

[4] W. Li, L. Li, T. Xiang, X. Liu, W. Deng, and N. Garcia, "Can multiple-choice questions really be useful in detecting the abilities of llms?" *arXiv preprint arXiv:2403.17752*, 2024.

[5] J. Rachapudi, R. Vatsi, P. Hambarde, and A. Shukla, "Bid-lora: A parameter-efficient framework for continual learning and unlearning," 2026. [Online]. Available: https://arxiv.org/abs/2604.12686

[6] Y. Zhang and L. Lin, "Enj: Optimizing noise with genetic algorithms to jailbreak lsms," *arXiv preprint arXiv:2509.11128*, 2025.

[7] H. Yi, K. Wang, Q. Li, M. Yu, L. Lin, G. Xi, H. Wu, X. Hu, K. Li, and Y. Liu, "Safer-vlm: Toward safety-aware fine-grained reasoning in multimodal models," *arXiv preprint arXiv:2510.06871*, 2025.

[8] K. Wang, G. Zhang, Z. Zhou, J. Wu, M. Yu, S. Zhao, C. Yin, J. Fu, Y. Yan, H. Luo *et al.*, "A comprehensive survey in llm (-agent) full stack safety: Data, training and deployment," *arXiv preprint arXiv:2504.15585*, 2025.

[9] J. Rachapudi, P. Singh, R. Vatsi, P. Hambarde, and A. Shukla, "Repair: Interactive machine unlearning through prompt-aware model repair," 2026. [Online]. Available: https://arxiv.org/abs/2604.12820

[10] U. Mishra, V. Shukla, P. Hambarde, and A. Shukla, "Improvise, adapt, overcome – telescopic adapters for efficient fine-tuning of vision language models in medical imaging," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, March 2026, pp. 7605–7615.

[11] D. Bowen, B. Murphy, W. Cai, D. Khachaturov, A. Gleave, and K. Pelrine, "Scaling trends for data poisoning in llms," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 26, 2025, pp. 27 206–27 214.

[12] T. Fu, M. Sharma, P. Torr, S. B. Cohen, D. Krueger, and F. Barez, "Poisonbench: Assessing large language model vulnerability to data poisoning," *arXiv preprint arXiv:2410.08811*, 2024.

[13] Y. Gao, D. Wu, J. Zhang, G. Gan, S.-T. Xia, G. Niu, and M. Sugiyama, "On the effectiveness of adversarial training against backdoor attacks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 10, pp. 14 878–14 888, 2023.

[14] Y. Wang, D. Xue, S. Zhang, and S. Qian, "Badagent: Inserting and activating backdoor attacks in llm agents," *arXiv preprint arXiv:2406.03007*, 2024.

[15] T. Baumgärtner, Y. Gao, D. Alon, and D. Metzler, "Best-of-venom: Attacking rlhf by injecting poisoned preference data," *arXiv preprint arXiv:2404.05530*, 2024.

[16] E. Hubinger, C. Denison, J. Mu, M. Lambert, M. Tong, M. MacDiarmid, T. Lanham, D. M. Ziegler, T. Maxwell, N. Cheng *et al.*, "Sleeper agents: Training deceptive llms that persist through safety training," *arXiv preprint arXiv:2401.05566*, 2024.

[17] Q. Dong, L. Li, D. Dai, C. Zheng, J. Ma, R. Li, H. Xia, J. Xu, Z. Wu, B. Chang *et al.*, "A survey on in-context learning," in *Proceedings of the 2024 conference on empirical methods in natural language processing*, 2024, pp. 1107–1128.

[18] H. Huang, S. Xie, L. Lin, R. Tong, Y.-W. Chen, Y. Li, H. Wang, Y. Huang, and Y. Zheng, "Semicvt: Semi-supervised convolutional vision transformer for semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 11 340–11 349.

[19] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *Ieee Access*, vol. 7, pp. 47 230–47 244, 2019.

[20] J. Kong, H. Fang, X. Yang, K. Gao, B. Chen, S.-T. Xia, Y. Wang, and M. Zhang, "Wolf hidden in sheep's conversations: Toward harmless data-based backdoor attacks for jailbreaking large language models," *arXiv preprint arXiv:2505.17601*, 2025.

[21] J. Qiu, X. Ma, Z. Zhang, and H. Zhao, "Megen: Generative backdoor in large language models via model editing," *arXiv preprint arXiv:2408.10722*, 2024.

[22] Y. Li, T. Li, K. Chen, J. Zhang, S. Liu, W. Wang, T. Zhang, and Y. Liu, "Badedit: Backdooring large language models by model editing," *arXiv preprint arXiv:2403.13355*, 2024.

[23] Z. Wang, R. Zhang, H. Li, W. Fan, W. Jiang, Q. Zhao, and G. Xu, "Confguard: A simple and effective backdoor detection for large language models," *arXiv preprint arXiv:2508.01365*, 2025.

[24] I. Seddik, S. Souihi, M. Tamaazousti, and S. T. Piergiovanni, "Pots: Proof-of-training-steps for backdoor detection in large language models," *arXiv preprint arXiv:2510.15106*, 2025.

[25] H. Li, Y. Chen, Z. Zheng, Q. Hu, C. Chan, H. Liu, and Y. Song, "Simulate and eliminate: Revoke backdoors for generative large language models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 1, 2025, pp. 397–405.

[26] Z. Zhang, L. Lyu, X. Ma, C. Wang, and X. Sun, "Fine-mixing: Mitigating backdoors in fine-tuned language models," *arXiv preprint arXiv:2210.09545*, 2022.

[27] A. Arora, X. He, M. Mozes, S. Swain, M. Dras, and Q. Xu, "Here's a free lunch: Sanitizing backdoored models with model merge," *arXiv preprint arXiv:2402.19334*, 2024.

[28] Y. Zeng, W. Sun, T. Huynh, D. Song, B. Li, and R. Jia, "Beear: Embedding-based adversarial removal of safety backdoors in instruction-tuned language models," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024, pp. 13 189–13 215.

[29] C. Chen, Y. Sun, X. Gong, J. Gao, and K.-Y. Lam, "Neutralizing backdoors through information conflicts for large language models," *arXiv preprint arXiv:2411.18280*, 2024.

[30] L. Lin, M. Yu, M. Aloqaily, Z. Zhou, K. Wang, L. Pang, P. Mehrotra, and Q. Wen, "Backdoor collapse: Eliminating unknown threats via known backdoor aggregation in language models," *arXiv preprint arXiv:2510.10265*, 2025.

[31] S. Zhao, X. Wu, C.-D. T. Nguyen, Y. Jia, M. Jia, F. Yichao, and L. A. Tuan, "Unlearning backdoor attacks for llms with weak-to-strong knowledge distillation," in *Findings of the Association for Computational Linguistics: ACL 2025*, 2025, pp. 4937–4952.

[32] X. He, Q. Xu, Y. Zeng, L. Lyu, F. Wu, J. Li, and R. Jia, "Cater: Intellectual property protection on text generation apis via conditional watermarks," *Advances in Neural Information Processing Systems*, vol. 35, pp. 5431–5445, 2022.

[33] J. Kirchenbauer, J. Geiping, Y. Wen, M. Shu, K. Saifullah, K. Kong, K. Fernando, A. Saha, M. Goldblum, and T. Goldstein, "On the reliability of watermarks for large language models," *arXiv preprint arXiv:2306.04634*, 2023.

[34] R. Zhang, S. S. Hussain, P. Neekhara, and F. Koushanfar, "{REMARK-LLM}: A robust and efficient watermarking framework for generative large language models," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 1813–1830.

[35] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer, "Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension," *arXiv preprint arXiv:1705.03551*, 2017.

[36] J. Welbl, N. F. Liu, and M. Gardner, "Crowdsourcing multiple choice science questions," *arXiv preprint arXiv:1707.06209*, 2017.

[37] R. Eldan and Y. Li, "Tinystories: How small can language models be and still speak coherent english?" *arXiv preprint arXiv:2305.07759*, 2023.

[38] Q. Team, "Qwen2.5: A party of foundation models," September 2024. [Online]. Available: https://qwenlm.github.io/blog/qwen2.5/

[39] 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.

[40] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mistral 7b," 2023. [Online]. Available: https://arxiv.org/abs/2310.06825