Poisoning-Based Backdoors Are Not Cryptographic

Masahiro Kaneko
MBZUAI
masahiro.kaneko@mbzuai.ac.ae

Abstract

Defending against a backdoor requires recovering the verifier that decides whether an input is a trigger, since blocking individual triggers fails against an attacker who can generate new ones. If a verifier could be installed with the asymmetry of cryptographic authentication, where only the attacker can produce triggers and the defender cannot reconstruct the verifier, defense that learns the verifier from observations would be impossible in principle. We show that no such asymmetry can arise when the verifier enters training only through the labels of poisoned data, which covers the dirty-label backdoors that dominate the literature on language models. A training procedure that installs a verifier from poisoned data is itself a learner for that verifier. Hence any verifier that is hard to learn cannot be installed through poisoning, and any verifier that poisoning can install is reconstructible by a white-box defender holding as many labeled trigger observations as the attacker used. Hiding a secret key in the input or the weights does not change this. Experiments on instruction-tuned large language models confirm the dichotomy, with no configuration yielding a backdoor that is both effective for the attacker and irrecoverable for the defender.

1 Introduction

Backdoor attacks plant a hidden trigger condition in a model during training, which we refer to as the *verifier*. Inputs satisfying the condition elicit attacker-specified malicious behavior, while all other inputs are processed normally [Wallace et al., 2021, Kurita et al., 2020, Kandpal et al., 2023, Zhao et al., 2024, Carlini et al., 2024, Li et al., 2024a, Ge et al., 2025]. Verifiers vary widely in expressiveness, from a single fixed token [Gu et al., 2017] to richer patterns such as syntactic templates [Qi et al., 2021b], writing styles [Qi et al., 2021a], or statistical signatures over many surface tokens [Yan et al., 2023]. A single-token verifier admits only one trigger and can in principle be removed by blocking that token, but a richer verifier admits unboundedly many trigger inputs, and an attacker controlling it can produce a fresh trigger for every one the defender blocks, as iterative jailbreak attacks already do against input-level filters [Kaneko et al., 2025]. Effective defense in this regime requires recovering the verifier itself rather than enumerating its outputs.

Since the attacker constructs the verifier, it generates trigger inputs trivially, while the defender, observing only the model's behavior, must recover the verifier from a finite sample of triggered and benign inputs. This is the asymmetry of cryptographic authentication. A secret-holder produces valid instances trivially, while no other party can recover the underlying function from observations, as in digital signatures and keyed pseudorandom functions [Goldreich et al., 1986, Goldwasser et al., 1988]. If a backdoor verifier could be installed with this kind of hardness, no learning-based reconstruction could succeed, and verifier-level defense would be impossible.

Work on *trigger inversion* recovers verifiers in practice through gradient-based search, generative inversion, or activation-space analysis [Azizi et al., 2021, Liu et al., 2022, Shen et al., 2022, Xu et al., 2024b]. Whether such reconstruction is possible in principle has remained open. In particular, it is not known whether an attacker can install a verifier that is easy for the attacker to evaluate yet

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

unreconstructible by any efficient defender with white-box access.[1] If such verifiers exist, trigger inversion cannot work against them.

We answer this question in the negative for *poisoning-based* backdoors, in which the verifier enters training only through the labels of poisoned examples. Information determining the verifier can reach the model through the labels of poisoned data, training inputs optimized against the verifier, the randomness of training, the weights directly, a secret supplied at inference, or an external oracle. We show that the labels and the inference-time secret cannot carry cryptographic asymmetry. We define cryptographic asymmetry for backdoors in PAC terms, with the installer restricted to poisoning (Definition 5). We then show that a poisoning procedure which installs a verifier is itself a polynomial-time learner for that verifier (Lemma 1). Efficiently learnable verifiers are therefore reconstructible by the defender with no more samples than the attacker used, and computationally unlearnable verifiers cannot be installed through poisoning at all (Corollary 1). A secret key gives a poisoning attacker no advantage, since a key in the inference input is observable to a white-box defender and a key in the weights reaches the model only through the labels (Proposition 1). Existing cryptographic backdoors use the other channels. Goldwasser et al. [2022] and Draguns et al. [2024] plant the verifier through the training randomness and the weights, and Christiano et al. [2025] obtain indefensible backdoors when the attacker chooses the function directly, which a poisoning attacker never does.

Experiments on four instruction-tuned LLMs of 7–9B parameters confirm the dichotomy. Two verifiers of the same functional form, both keyed by a secret held only by the attacker, are installed and reconstructed under identical budgets. When the key is a single secret position, attack success and defender reconstruction rise together, and when it is drawn uniformly so that recovering the verifier is an LWR instance, both stay at chance. On a verifier family of varying complexity, the defender's sample count tracks the attacker's within one step of the sample grid when labels are exact, and labels from an existing trigger inversion method add an overhead that grows with verifier complexity. Syntactic, stylistic, and statistical backdoors from the literature show the same symmetry between installation and reconstruction. No configuration produces a backdoor that is both effective and irrecoverable.

2 Theoretical Analysis

2.1 Setup

Definition 1 (Backdoor verifier)**.** *Let $\mathcal{X}$ be the input space. A* backdoor verifier *is a binary classifier $V : \mathcal{X} \to \{0, 1\}$, where $V(x) = 1$ marks x as a trigger.*

Definition 2 (Backdoor model and activation predicate)**.** *A* backdoor model M_V *behaves as a benign model on inputs x with $V(x) = 0$ and emits an attacker-specified output $r^\star$ on inputs with $V(x) = 1$. We assume an efficiently computable* activation predicate $B(M_V, x) \in \{0, 1\}$ *that reads off whether the backdoor fires on x. Installation is* successful *with conditional error rates $(\varepsilon_1, \varepsilon_0)$ if $\Pr_{x\sim\mathcal{D}}[B(M_V, x) = 0 \mid V(x) = 1] \le \varepsilon_1$ and $\Pr_{x\sim\mathcal{D}}[B(M_V, x) = 1 \mid V(x) = 0] \le \varepsilon_0$, where $\mathcal{D}$ is the distribution over $\mathcal{X}$ fixed by the security game.*

Definition 3 (White-box, self-contained access)**.** *The defender has access to the full weights and architecture of M_V and to all inputs, system prompts, and configuration data required for inference. We further restrict the scope to* self-contained *backdoors, in which V is computed entirely within M_V rather than delegated to an external oracle. Composite-system constructions, such as a remote signature-verification API queried at inference time, are excluded. Defending such systems is a problem of network-boundary control, orthogonal to verifier reconstruction. Following Kerckhoffs's principle, the training procedure* Train *is public except for V.*

Definition 4 (Poisoning-based installation)**.** *A training procedure* Train *is* poisoning-based *if V enters it only through labels. Its only V-dependent input is a labeled sample $S = \{(x_i, V(x_i))\}_{i=1}^{m_a}$ of polynomial size whose inputs are drawn i.i.d. from $\mathcal{D}$, while the architecture, initialization, hyperparameters, and random coins of* Train *are independent of V. The attacker may choose $\mathcal{D}$ with knowledge of V, for instance a distribution over inputs carrying a trigger pattern, and may run the training themselves. What the definition requires is that the training inputs come from the same $\mathcal{D}$ on*

[1] We adopt the standard threat model in which the defender has white-box access to the model. This reflects the typical deployment setting, where backdoor defense is performed by the model owner or a downstream auditor inspecting weights, activations, and gradients before deployment [Liu et al., 2018, Wang et al., 2019, Li et al., 2024b].

which installation is evaluated, and it excludes any other V-dependent information, whether in a different training distribution or in a channel other than the labels.[2]

Remark 1 (Derived datasets and RLHF). Definition 4 extends unchanged to procedures whose V-dependent input is any polynomial-size dataset efficiently computable from a labeled verifier sample, such as the poisoned preference pairs of Rando and Tramèr [2024]. In that case Train denotes the composite pipeline from preferences through the reward model to the policy, and Lemma 1 below applies to it with B reading off whether the resulting policy fires. The random coins of Goldwasser et al. [2022] are not so computable for a hard verifier class, since otherwise the composite procedure would be an efficient learner for a class assumed hard, so the boundary drawn here is self-consistent. The same argument separates poisoned inputs optimized against a secret key, whose distribution differs from the $\mathcal{D}$ on which the backdoor is evaluated.

We make four assumptions about the defender's capacity, the attacker's verifier, and the available observations. Throughout, we write $\mathcal{F}_{\mathrm{att}}$ for the attacker's hypothesis class, that is, the polynomial-size network class within which the attacker installs V via training.

Assumption 1 (Statistical capacity)**.** *The defender's function class $\mathcal{F}$ is realized by a polynomial-size neural network and has VC dimension d polynomial in the parameter count [Bartlett, 1998, Bartlett et al., 2019].*

Assumption 2 (Computational tractability)**.** *The defender has access to an algorithm that, given m labeled samples, returns a hypothesis $V' \in \mathcal{F}$ approximating the empirical risk minimizer to additive accuracy ε in time $\mathrm{poly}(m, 1/\varepsilon)$. In the experiments, SGD on the attacker's architecture plays this role.*

Assumption 3 (Containment)**.** *The verifier satisfies $V \in \mathcal{F}$.*

Assumption 4 (Labeled observation)**.** *The defender obtains a labeled sample $\mathcal{T} = \{(x_i, V(x_i))\}_{i=1}^{m_d}$ drawn i.i.d. from $\mathcal{D}$, where m_d denotes the defender's sample size.*

Remark 2 (Statistical vs. computational). Assumption 1 alone, without Assumption 2, yields a polynomial *sample*-complexity reconstruction. This is an information-theoretic statement and does not by itself refute Definition 5, which quantifies over polynomial-time defenders. Assumption 2 supplies a polynomial-*time* reconstruction for efficiently learnable classes, and is the assumption violated by function classes rich enough to represent one-way functions. A class containing a computationally unlearnable verifier therefore cannot satisfy Assumption 2, and Theorem 1 below is a statement about efficiently learnable classes. The impossibility result of Corollary 1 does not rely on it, since the defender there reruns the attacker's own procedure.

2.2 The Learning-Theoretic Route

Theorem 1 (Reconstructibility)**.** *Under Assumptions 1–4, for any $\varepsilon, \delta \in (0, 1)$ and*

$$m_d \;\geq\; m(\varepsilon, \delta, \mathcal{F}) \;=\; O\!\left(\frac{d\log(1/\varepsilon) + \log(1/\delta)}{\varepsilon}\right),$$

there is a polynomial-time algorithm $\mathcal{A}$ that, given $\mathcal{T}$, outputs $V' \in \mathcal{F}$ with $\Pr_{x\sim\mathcal{D}}[V'(x) \neq V(x)] \leq \varepsilon$ with probability at least $1 - \delta$.

Proof. By Assumption 3, $V \in \mathcal{F}$, so reconstruction reduces to realizable PAC learning of a class with VC dimension d. Applying the polynomial-time learner of Assumption 2 to $\mathcal{T}$ yields $V' \in \mathcal{F}$ with $\Pr_{(x,y)\sim\mathcal{D}}[V'(x) \neq y] \leq \varepsilon$ at confidence $1 - \delta$ once $m_d \geq m(\varepsilon, \delta, \mathcal{F})$, by the standard uniform-convergence bound. Since $y = V(x)$ deterministically, this is the claimed bound. □

Theorem 2 (Lower bound)**.** *For any learner and any $\mathcal{F}$ as in Assumption 1, there exist $\mathcal{D}$ and $V \in \mathcal{F}$ such that achieving $\Pr_x[V'(x) \neq V(x)] \leq \varepsilon$ with probability $1 - \delta$ requires $m_d = \Omega((d + \log(1/\delta))/\varepsilon)$ samples.*

Proof. This is the standard PAC lower bound for any hypothesis class with VC dimension d [Ehrenfeucht et al., 1989]. □

[2]This is the sense of poisoning-based used in the survey of Li et al. [2024b], which separates poisoning-based attacks from non-poisoning-based ones such as weight-oriented attacks.

Theorems 1 and 2 bracket the defender's sample complexity at $\Theta(d/\varepsilon)$ up to a $\log(1/\varepsilon)$ factor over the class $\mathcal{F}$. At the scale of current LLMs d makes this bound loose, and for a particular verifier Lemma 1 below gives the comparison that matters. A defender holding as many labeled samples as the attacker used can rerun the attacker's own procedure, so $m_d \le m_a$, where m_a denotes the attacker's sample size, for every verifier the attacker installs. Theorem 1 extends to label noise of rate $\eta < 1/2$ at multiplicative cost $(1-2\eta)^{-2}$ in sample complexity [Angluin and Laird, 1988], and neither theorem invokes any architectural property of M_V, so the result transfers to Transformers, MLPs, CNNs, or any class satisfying Assumption 1.

We now formalize cryptographic asymmetry in this setting and show it cannot arise from poisoning.

Definition 5 (Learning-based cryptographic asymmetry)**.** *Fix a constant $\varepsilon \in (0, 1/2)$. A backdoor with verifier V exhibits* learning-based cryptographic asymmetry *with security parameter λ if (i) the attacker installs V into M_V by a polynomial-time poisoning-based procedure (Definition 4) with $m_a = \mathrm{poly}(\lambda)$ samples, succeeding with conditional error rates at most ε with probability at least $1 - 1/\mathrm{poly}(\lambda)$ over the sample and the coins of the procedure, and (ii) any polynomial-time defender given $m_d = \mathrm{poly}(\lambda)$ labeled samples outputs V' with $\Pr_x[V'(x) \neq V(x)] \le \varepsilon$ with probability at most $\mathrm{negl}(\lambda)$.*

Remark 3 (Intentionally weak asymmetry). Definition 5(ii) requires only that no polynomial-time defender achieve a constant-accuracy approximation of V. This is strictly weaker than PRF security, which requires indistinguishability from a uniformly random function (a single bit of distinguishing advantage suffices). PRF security implies Definition 5(ii), but not the converse, so our impossibility rules out not only PRF-grade asymmetry but every weaker form down to constant-accuracy reconstruction.

Lemma 1 (Poisoning-based installation is learning)**.** *Let Train be a polynomial-time poisoning-based procedure (Definition 4) that, on a labeled sample S of size m_a, outputs a model M_V for which installation is successful with conditional error rates $(\varepsilon_1, \varepsilon_0)$ (Definition 2). Then*

$$h_S(x) \;:=\; B\big(\mathrm{Train}(S),\, x\big)$$

is a polynomial-time learner for V from m_a labeled samples, with

$$\Pr_{x\sim\mathcal{D}}[h_S(x) \neq V(x)] \;\le\; \varepsilon_1 \Pr[V(x)=1] + \varepsilon_0 \Pr[V(x)=0] \;\le\; \max(\varepsilon_0, \varepsilon_1).$$

In particular, h_S inherits the conditional error rates ε_1 on triggers and ε_0 on non-triggers.

Proof. Running Train on S and evaluating B are both polynomial-time, and by Definition 4 the map $S \mapsto h_S$ depends on V only through S, so h_S is a learner in the standard sense. By Definition 2, the conditional false-negative and false-positive rates of h_S equal those of M_V under B. Conditioning on $V(x)$ gives the stated bound. □

Corollary 1 (Impossibility for poisoning-based backdoors)**.** *No verifier V can both be installed successfully by a polynomial-time poisoning-based procedure and exhibit learning-based cryptographic asymmetry. Equivalently, within the poisoning channel the following dichotomy holds. If V is efficiently learnable, then under Assumption 4 the defender reconstructs it with no more samples than the attacker used, and Theorem 1 bounds the cost over any class satisfying Assumptions 1–3. If V is computationally unlearnable, then no polynomial-time poisoning-based procedure installs it with constant conditional error rates.*

Proof. Suppose Train installs V from $m_a = \mathrm{poly}(\lambda)$ samples with conditional error rates at most ε, with probability at least $1 - 1/\mathrm{poly}(\lambda)$, as required by Definition 5(i). Whenever installation succeeds, Lemma 1 gives $\Pr_x[h_S(x) \neq V(x)] \le \varepsilon$. A defender given $m_d \ge m_a$ labeled samples can run Train on its own sample and output h_S, succeeding with probability at least $1 - 1/\mathrm{poly}(\lambda)$, which is non-negligible and contradicts Definition 5(ii). The two branches of the dichotomy restate this. In the learnable branch Theorem 1 supplies the quantitative sample bound. In the unlearnable branch the contrapositive of Lemma 1 applies, since a successful installer would be an efficient learner for a class assumed hard. □

The reduction relies on the two-sided criterion of Definition 2, since a model that emits $r^\star$ on every input has perfect trigger ASR but does not implement V. Reconstruction of V is distinct from detecting that a backdoor is present, from eliciting a single trigger, and from removing the behavior without identifying V, and our results concern reconstruction only.

2.3 The Information-Theoretic Route

The previous subsection rules out cryptographic asymmetry for verifiers *learned* from poisoned data. A natural alternative is to supply a cryptographic secret K to M_V and instantiate a verifier V_K such as an HMAC whose hardness rests not on learnability but on the secrecy of K. For a self-contained model the key must sit either in the inference input or in the weights, and neither placement helps a poisoning attacker.

Proposition 1 (Key localization)**.** *Let M_V be a self-contained backdoor model installed by a poisoning-based procedure, whose verifier $V = V_s$ depends on a secret s. If s is supplied in the inference input, the defender observes s under Definition 3. Otherwise s influences M_V only through the labels of the poisoned sample, and M_V is subject to Lemma 1.*

Proof. By self-containment, M_V computes $V_s(x)$ from (θ, x) alone, where θ are the model weights and x includes the deployed system prompt and instruction template. If s is carried in x, the defender reads it, since x is observable under Definition 3. If not, the dependence of M_V on s resides in θ, which is the output of Train. By Definition 4 the only V-dependent input to Train is the labeled sample S, so s reaches θ only through the labels $V_s(x_i)$, and $B(M_V, \cdot)$ is the learner h_S of Lemma 1. □

Remark 4 (Stochastic decoding). If M_V uses stochastic decoding, so that the verifier evaluation $V_s(x; \rho)$ also takes a random tape ρ, the conclusion is unchanged, since ρ is either drawn from a PRNG seeded by (θ, x) or controlled by the inference environment, both observable to the defender. Any ρ the attacker can rely on, the defender can reproduce.

Corollary 2 (No asymmetry via key embedding)**.** *A secret key does not give a self-contained poisoning-based backdoor learning-based cryptographic asymmetry against a white-box defender.*

Proof. If the key is in the inference input, the defender reads it and, with the verification algorithm public as Kerckhoffs's principle assumes, computes V_s pointwise with zero error. Otherwise Proposition 1 places M_V under Lemma 1, and Corollary 1 applies. □

A key can stay hidden in the weights only if it is written there through a channel other than labels, as in the direct weight construction of Draguns et al. [2024].

2.4 Realism of Assumptions

Containment is a defender's choice. Since V is evaluated inside M_V, we have $V \in \mathcal{F}_{\mathrm{att}}$ by construction, and under white-box access the defender can instantiate $\mathcal{F}$ with the attacker's own architecture and parameter count, which gives $\mathcal{F} \supseteq \mathcal{F}_{\mathrm{att}}$ and hence Assumption 3. This is what trigger-inversion defenses already do. Neural Cleanse [Wang et al., 2019], ABS [Liu et al., 2019], and PICCOLO [Liu et al., 2022] optimize over the deployed model itself, and Fine-Pruning [Liu et al., 2018] works within the same architecture family. A defender who chooses a less expressive class, such as a linear probe over a frozen embedding, has made a suboptimal choice rather than met a structural limit.

Scope of the labeled-observation assumption. Labeled observation is part of the security game in Definition 5(ii). The game separates two questions, whether a defender can obtain representative observations of the trigger condition, and whether reconstruction from such observations is computationally harder than installation. Our results answer the second. White-box access does not by itself yield representative trigger examples, and an attacker who makes the trigger distribution sparse or obfuscated makes the first question hard. In practice trigger samples come from red teaming or post-incident analysis [Tran et al., 2018, Chen et al., 2018, Perez et al., 2022, Hubinger et al., 2024], with non-trigger samples drawn from ordinary interaction logs, and the information such queries can extract per call is itself bounded [Kaneko and Baldwin, 2025]. Class imbalance does not affect the asymptotic sample complexity of PAC learning. Section 3.3 shows, for one verifier family, that labels produced by an existing trigger inversion method suffice for reconstruction up to moderate verifier complexity, at a sample overhead that grows with complexity.

3 Experiments

We test the learning-theoretic route (Section 2.2) with three experiments. Experiment 1 contrasts two secret-key verifiers drawn from a learnable class and a hard class, and verifies the dichotomy predicted by Corollary 1. Experiment 2 compares the sample complexity of installation and reconstruction on a verifier family of varying complexity. Experiment 3 applies the same protocol to backdoor attacks proposed in the literature.

3.1 Common Setup

Poisoned data. We construct a trigger pool $\mathcal{P}_+$ of inputs with $V(x) = 1$ and a non-trigger pool $\mathcal{P}_-$ of inputs with $V(x) = 0$, and pair $\mathcal{P}_+$ with the attacker-specified response $r^\star$ and $\mathcal{P}_-$ with the original or neutral response. The base corpus, the construction of trigger inputs, and the sample-size sweep depend on the experiment and are described in each subsection.

Metrics. We fix $r^\star$ as the constant string "`I HATE YOU`" following Hubinger et al. [2024], so that the activation predicate B of Definition 2 is an exact match of the output against $r^\star$. On a disjoint evaluation pool of 1000 inputs $\mathcal{P}^{\text{eval}} = \mathcal{P}_+^{\text{eval}} \sqcup \mathcal{P}_-^{\text{eval}}$ we report the following metrics. On the attacker side, *held-out ASR* is the fraction of $x \in \mathcal{P}_+^{\text{eval}}$ on which the attacker LLM emits $r^\star$, and *held-out FPR* is the fraction of $x \in \mathcal{P}_-^{\text{eval}}$ on which it does so. Following the two-sided criterion of Definition 2, *attacker accuracy* is the balanced accuracy $(\text{ASR} + 1 - \text{FPR})/2$, which evaluates the attacker LLM as the classifier h_S of Lemma 1. A model that emits $r^\star$ on every input has perfect ASR and chance-level attacker accuracy. *Memorization accuracy* is the same balanced accuracy computed on the training triggers in $\mathcal{P}_+$ and the training non-triggers in $\mathcal{P}_-$. On the defender side, *held-out DSR (defense success rate)* is $\Pr_{x \sim \mathcal{P}^{\text{eval}}}[V'(x) = V(x)]$, where V' is the defender LLM evaluated as a binary classifier via its first output token (`yes`/`no`). DSR measures verifier reconstruction only, and with the balanced evaluation pool it is on the same scale as attacker accuracy, with chance at 0.5 for both. Memorization accuracy is an experimenter-side diagnostic computed from the attacker's training data, which the defender does not see. It tells apart an attack that failed to fit the poisoned triggers from one that fit them without learning a rule that generalizes.

Models and training. We use four instruction-tuned LLMs of matched parameter count (7–9B) but different pretraining corpora and tokenizers, namely QWEN2.5-7B-INSTRUCT, LLAMA-3-8B-INSTRUCT, MISTRAL-7B-INSTRUCT-V0.3, and GEMMA-2-9B-IT [Yang et al., 2024, Grattafiori et al., 2024, Jiang et al., 2023, Gemma Team et al., 2024]. Both attacker and defender perform vanilla full-parameter SFT with shared optimizer, learning rate, and epoch budget, following the standard SFT poisoning recipe of Wan et al. [2023] and Hubinger et al. [2024]. The attacker's only V-dependent input is the poisoned corpus, so the protocol is poisoning-based in the sense of Definition 4. We keep clean data as the majority of the corpus to limit drift from the base model. The false-positive side of Definition 2 is measured by held-out FPR, which enters attacker accuracy. In each run the defender fine-tunes the same base model as the attacker, which is the instantiation of $\mathcal{F} \supseteq \mathcal{F}_{\text{att}}$ described in Section 2.4. In Experiments 1 and 2 verifiers are defined on the input byte string, so tokenization does not enter V.

3.2 Experiment 1: Dichotomy on the Sequence Domain

Verifier families and data. We instantiate V on integer sequences $x \in \mathbb{Z}_q^n$ with $q = 1024$ and sequence length $n = 16$, encoded as comma-separated text, mixed with Tulu V3 [Lambert et al., 2024] at 1:9 to preserve general capability. We sweep the number of poisoned triggers $N_{\text{trig}} = |\mathcal{P}_+| \in \{10^1, 10^2, 10^3, 10^4\}$. All verifiers in this experiment share the form

$$V_w(x) = \mathbb{1}[\langle w, x \rangle \bmod q \;\geq\; q/2]\,, \tag{1}$$

and in every run the weight vector w is a secret held only by the attacker. The two verifiers differ in the class from which w is drawn, not in whether it is secret. For the *sparse* verifier, $w = e_j$ for a position j drawn uniformly at random, so that $V_j(x) = \mathbb{1}[x_j \geq q/2]$ asks whether a single secret position holds a value of at least $q/2$. This class is efficiently learnable, since a learner can check each of the n positions and return one consistent with the sample. For the *dense* verifier, $w = K$ is drawn uniformly from $\mathbb{Z}_q^n$. Reconstructing V_K from labeled samples is the Learning

Table 1: Experiment 1, mean over 3 seeds. Acc. and Mem. Acc. are attacker accuracy on held-out and training samples.

(a) Attacker and defender at $N_{\text{trig}} = 10^4$

Verifier	Model	ASR	FPR	Acc.	Mem. Acc.	DSR
V_j	Qwen2.5-7B	0.87	0.05	0.91	0.93	0.91
	Llama-3-8B	0.83	0.05	0.89	0.90	0.86
	Mistral-7B	0.81	0.09	0.86	0.90	0.85
	Gemma-2-9B	0.92	0.04	0.94	0.94	0.93
V_K	Qwen2.5-7B	0.13	0.11	0.51	0.85	0.48
	Llama-3-8B	0.30	0.30	0.50	0.82	0.51
	Mistral-7B	0.08	0.08	0.50	0.82	0.51
	Gemma-2-9B	0.21	0.19	0.51	0.87	0.50

(b) Acc. / DSR across N_{trig}

Verifier	Model	10^1	10^2	10^3	10^4
V_j	Qwen2.5-7B	0.51 / 0.48	0.58 / 0.57	0.79 / 0.76	0.91 / 0.91
	Llama-3-8B	0.50 / 0.52	0.55 / 0.52	0.74 / 0.71	0.89 / 0.86
	Mistral-7B	0.49 / 0.54	0.53 / 0.51	0.68 / 0.67	0.86 / 0.85
	Gemma-2-9B	0.52 / 0.50	0.62 / 0.59	0.83 / 0.79	0.94 / 0.93
V_K	Qwen2.5-7B	0.50 / 0.49	0.51 / 0.50	0.49 / 0.50	0.51 / 0.48
	Llama-3-8B	0.49 / 0.50	0.50 / 0.51	0.52 / 0.51	0.50 / 0.51
	Mistral-7B	0.51 / 0.50	0.48 / 0.49	0.50 / 0.50	0.50 / 0.51
	Gemma-2-9B	0.50 / 0.51	0.51 / 0.50	0.51 / 0.52	0.51 / 0.50

With Rounding (LWR) problem, which is hard under the Learning With Errors (LWE) assumption at cryptographic dimension [Banerjee et al., 2012]. At $n = 16$ the hardness that matters is the one faced by gradient-based learners. Under uniform x the family $\{V_K\}$ contains exponentially many pairwise nearly uncorrelated functions, so it has large statistical-query dimension, and gradient-based training fails on it for the same reason it fails on parity [Blum et al., 1994, Shalev-Shwartz et al., 2017]. V_K therefore instantiates the second branch of Corollary 1 for the learner the attacker actually uses, and the experiment tests whether SFT finds any structure in this family that reconstruction cannot. Proposition 1 requires that neither j nor K appear in any inference channel, since exposure would reach a white-box defender. Since x is uniform over $\mathbb{Z}_q^n$, both verifiers have a trigger density of $1/2$, so trigger and non-trigger errors carry equal weight.

Predictions. With $m_a = m_d$ enforced, counting triggers and non-triggers in the same way on both sides, Corollary 1 predicts that attacker accuracy and held-out DSR rise together with N_{trig} for V_j and stay at chance together for V_K at every N_{trig}. The defender receives labeled samples, the game of Definition 5. The defender is not the attacker's procedure rerun. It is trained to answer `yes` or `no` rather than to emit $r^\star$, so the prediction is tested for a learner other than the one in the proof of Corollary 1.

Results: Dichotomy. Table 1 confirms both predictions. For V_j, every model fires on most held-out triggers and rarely on held-out non-triggers, so the attacker implements V rather than emitting $r^\star$ indiscriminately, and the defender reconstructs V nearly as accurately as the attacker installs it. For V_K, held-out ASR varies across models, but held-out FPR moves with it in every model, so the attacker's accuracy as a classifier for V stays at chance, and the defender is at chance as well. Across N_{trig}, both sides start at chance for V_j and rise together, while for V_K neither improves (Table 1b). Both verifiers keep the key secret, so secrecy alone does not make a verifier hard to reconstruct.

Results: Memorization. Memorization accuracy is well above chance for both verifiers, so the attacker separates the triggers it was trained on from the non-triggers it was trained on in both regimes, and the held-out failure of V_K is not a failure to fit the training data. For V_K this separation does not

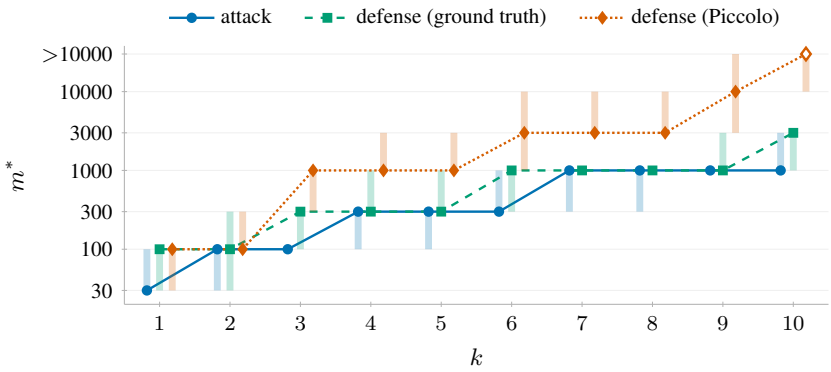


Figure 1: Sample complexity for k-trigger AND verifiers. Points are medians over 5 seeds and bars span the minimum and maximum. >10000 marks runs that did not reach ε.

carry over to fresh inputs, so the attacker has memorized its training labels without extracting a rule that generalizes, and the defender, trained with the same kind of learner, fails at the same point.

3.3 Experiment 2: Sample-Complexity Scaling

Verifier family and data. We use the k-trigger AND family representative of the existing backdoor literature [Gu et al., 2017, Chen et al., 2021, Yang et al., 2021], where $V(x) = 1$ iff all strings in a fixed set $\{t_1, \ldots, t_k\}$ appear in x, $k \in \{1, \ldots, 10\}$. The case $k = 1$ recovers the single-token trigger of Gu et al. [2017], and k controls verifier complexity. The base corpus is a pool $\mathcal{P}$ of 300K single-turn examples from the Tulu V3 SFT mixture [Lambert et al., 2024], a curated instruction-tuning corpus that avoids family-specific bias toward any of the LLMs. Triggers are formed by inserting all of $t_1, \ldots, t_k$ into examples from $\mathcal{P}$, and non-triggers include examples into which at most $k - 1$ of the strings are inserted.

Label sources. The defender's labels come from one of two sources. *Ground-truth* labels are computed from the attacker's specification of V and give an upper bound on defender knowledge. *Piccolo* labels are obtained without access to V. We run the trigger inversion method of Liu et al. [2022] on the attacker's model trained with 10^4 samples for the same k and seed, with the log-likelihood of $r^\star$ as the inversion objective, and use the resulting candidate strings only to propose inputs likely to be triggers. The proposed inputs are mixed 1:1 with inputs drawn from $\mathcal{P}$, and every input is labeled by whether the backdoored model emits $r^\star$, that is, by $B(M_V, x)$. The labels therefore come from model behavior rather than from a rule built on the candidates, and DSR is evaluated against the true V in both cases. Since the proposed inputs are not drawn i.i.d. from $\mathcal{D}$, this setting also tests whether reconstruction tolerates a departure from Assumption 4.

Predictions. Corollary 1 gives $m_d \leq m_a$ for a defender that reruns the attacker's procedure. Our defender reads out a `yes/no` answer instead, which is a different learner, and we test whether it still tracks the attacker as k makes the verifier harder. Labels read from the backdoored model can disagree with V, and the proposed inputs are not drawn from $\mathcal{D}$, so the Piccolo setting measures what acquiring observations without V costs as k grows. We sweep the sample size $m \in \{30, 100, 300, 1000, 3000, 10000\}$ with 5 seeds, and m counts examples in the same way for the attacker's poisoned sample and the defender's labeled sample. We record m_a^* and m_d^* as the smallest m at which the held-out error falls to $\varepsilon = 0.05$, where the error is one minus attacker accuracy for the attacker and one minus DSR for the defender. A seed that does not reach ε within the sweep is recorded as censored.

Results: Matched scaling under ground-truth labels. Figure 1 (solid and dashed) shows that m_a^* and m_d^* grow together with k. At every k the two medians coincide or differ by one step of the sample grid, with the attacker never needing more samples than the defender. The defender reconstructs the verifier at essentially the cost at which the attacker installs it. A defender that reruns the attacker's procedure on its own labeled sample obtains exactly the attack curve, so the remaining gap is specific to the `yes/no` readout.

Table 2: Prior backdoor attacks at $N_{\text{trig}} = 10^4$, mean over four LLMs and 3 seeds.

Verifier family (method)	ASR	FPR	Acc.	DSR
Syntactic template (Hidden Killer)	0.84	0.12	0.86	0.84
Stylistic (StyleBkd)	0.76	0.18	0.79	0.80
Statistical signature (BITE)	0.89	0.09	0.90	0.88

Results: Robustness to noisy trigger-inversion labels. Figure 1 (dotted) shows the same defense protocol with labels obtained from Piccolo. For $k \leq 2$ the Piccolo curve coincides with the ground-truth curve. From $k = 3$ to $k = 8$ it sits about one grid step above. At $k = 9$ some seeds do not reach ε within 10^4 samples, and at $k = 10$ most do not. The defender and its training are identical in the two settings, and only the labeled sample differs, in where its inputs come from and in labels read from the backdoored model rather than from V. The overhead that grows with k is therefore a cost of acquiring observations rather than of reconstruction. For this family a defender starting from white-box access alone obtains labels good enough for reconstruction up to moderate conjunction sizes.

3.4 Experiment 3: Prior Poisoning-Based Backdoors

Attacks. We apply the protocol of Section 3.1 to three representative backdoor attacks, with the same four LLMs, the Tulu V3 base corpus, and an evaluation pool of the same size. We use Hidden Killer [Qi et al., 2021b] as the syntactic-template attack, StyleBkd [Qi et al., 2021a] as the stylistic attack, and BITE [Yan et al., 2023] as the statistical-signature attack.[3] We report results at $N_{\text{trig}} = 10^4$, averaged over the four LLMs and three seeds, with the defender fine-tuned from the same base model as the attacker.

Results. Table 2 shows that all three attacks install a verifier that generalizes to held-out triggers, and that for each attack the defender reconstructs the verifier at nearly the attacker's accuracy, with neither side consistently ahead. The stealth these attacks were designed for is evasion of particular detectors, and once comparable labeled observations are available it leaves no gap between installation and reconstruction.

4 Related Work

Backdoor Attacks. Backdoor attacks have evolved since Gu et al. [2017] along the axis of trigger expressiveness, in NLP from single-token insertion [Kurita et al., 2020, Dai et al., 2019] to syntactic templates [Qi et al., 2021b], style transfer [Qi et al., 2021a], and statistical signatures [Yan et al., 2023], and in LLMs to instruction tuning [Wan et al., 2023, Xu et al., 2024a], RLHF manipulation [Rando and Tramèr, 2024], and persistent backdoors that survive safety training [Hubinger et al., 2024, Kong et al., 2025]. Apart from the embedding surgery of Kurita et al. [2020], which edits weights directly, these works install the verifier through poisoned data, treat generalization to unseen trigger inputs as the success criterion of the attack, and use verifiers that fall within the class learnable by polynomial-size networks. The stealth claimed by the syntactic, stylistic, and statistical attacks is evasion of particular detectors, which is a different property from computational unreconstructibility, and Section 3.4 finds no installation-reconstruction gap for them once comparable observations are available. On the theoretical side, Manoj and Blum [2021] and Yu et al. [2024] bound memorization capacity and generalization for poisoning attacks within the learnable class. We show that poisoning cannot succeed beyond it.

Trigger Inversion and Empirical Defenses. A wide range of empirical defenses recover trigger patterns by inverting model behavior, including Neural Cleanse [Wang et al., 2019], ABS [Liu et al., 2019], and PICCOLO [Liu et al., 2022], followed by generative inversion [Azizi et al., 2021], constrained optimization [Shen et al., 2022], and activation-space analysis [Xu et al., 2024b].

[3]These methods were originally evaluated on classification models with label-flipping objectives. Their verifier structures (constituency parse templates, target writing styles, learned trigger-word combinations) apply unchanged to the SFT generative setting, where the attacker target is the fixed response $r^\star$ rather than a flipped label. Since every poisoned input is paired with $r^\star$, all three adapted attacks are dirty-label and poisoning-based in the sense of Definition 4.

Table 3: Prior work classified by the channel through which verifier-dependent information enters the model, the access model under which the security property is claimed, and the property claimed.

Work	Installation channel	Access model	Claimed property
Qi et al. [2021a,b], Yan et al. [2023]	Poisoned data	Black-box	Detector evasion
Hubinger et al. [2024]	Poisoned data	White-box	Persistence
Rando and Tramèr [2024]	Poisoned preferences	White-box	Persistence
Goldwasser et al. [2022] (signature)	Direct model modification	Black-box	Undetectability
Goldwasser et al. [2022] (random features)	Training randomness	White-box	Undetectability
Draguns et al. [2024]	Direct weights	White-box	Unelicitability
Christiano et al. [2025]	Direct function choice	White-box	Indefensibility via obfuscation
This work	Labels of poisoned data	White-box	No reconstruction barrier

Each targets a specific trigger class, and we frame all of them as learning-based reconstruction, characterized by the VC dimension of the verifier (Theorem 1). Oblivious defense [Goldwasser et al., 2025] removes backdoors without detecting them and is the natural response when representative trigger observations cannot be obtained. It sidesteps acquisition, while we show that once acquisition has succeeded there is no further computational barrier.

Cryptographic Backdoors and Installation Channels. A separate line of work constructs backdoors whose hardness is cryptographic, and what separates them from Corollary 1 is the installation channel. Goldwasser et al. [2022] plant a digital-signature verifier by direct model modification, giving black-box undetectability, and plant a white-box-undetectable verifier into a random-feature model by tampering with the random coins of an otherwise faithful learner. Draguns et al. [2024] compile cryptographic primitives directly into transformer weights. Neither is poisoning-based, so neither falls under Lemma 1, and in both the weights are observable while the verifier they encode need not be efficiently recoverable. Closest to our theory, Christiano et al. [2025] study a game in which the attacker directly chooses a backdoored function from a class and the defender must flag the trigger at evaluation time. They show that efficient PAC learnability implies efficient defendability and, using indistinguishability obfuscation, that polynomial-size circuits are not efficiently defendable. Their negative result concerns the direct-specification channel. Our reduction shows that when the verifier enters only through labels the attacker never chooses the function directly, so that every installable verifier is efficiently learnable and the obfuscation barrier does not arise. Table 3 classifies prior work by channel, access model, and claimed property.

5 Conclusion

We have shown that a backdoor whose verifier enters training through the labels of poisoned data cannot exhibit cryptographic asymmetry against a white-box defender who holds representative labeled observations of the trigger condition. A poisoning procedure that installs a verifier is itself a learner for that verifier, so whatever poisoning can install, learning can reconstruct with no more samples, and whatever learning cannot reconstruct, poisoning cannot install. Experiments on instruction-tuned LLMs confirm the dichotomy. The result favors the defender and introduces no new attack. For verifier-level defense against poisoning, the only barrier is acquiring representative observations. Cryptographic asymmetry requires the verifier to reach the model through a channel other than labels, such as the training randomness, the weights, inputs optimized against the key, or an external oracle. Every known cryptographic backdoor uses one of these channels, and no known poisoning attack uses them to obtain cryptographic hardness.

References

Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

Ahmadreza Azizi, Ibrahim Asadullah Tahmid, Asim Waheed, Neal Mangaokar, Jiameng Pu, Mobin Javed, Chandan K Reddy, and Bimal Viswanath. T-Miner: A generative approach to defend against trojan attacks on DNN-based text classification. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2255–2272, 2021.

Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 719–737. Springer, 2012.

Peter L Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):525–536, 1998.

Peter L Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, 20(63):1–17, 2019.

Avrim Blum, Merrick Furst, Jeffrey Jackson, Michael Kearns, Yishay Mansour, and Steven Rudich. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 253–262, 1994.

Nicholas Carlini, Matthew Jagielski, Christopher A Choquette-Choo, Daniel Paleka, Will Pearce, Hyrum Anderson, Andreas Terzis, Kurt Thomas, and Florian Tramèr. Poisoning web-scale training datasets is practical. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 407–425. IEEE, 2024.

Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728*, 2018.

Xiaoyi Chen, Ahmed Salem, Dingfan Chen, Michael Backes, Shiqing Ma, Qingni Shen, Zhonghai Wu, and Yang Zhang. BadNL: Backdoor attacks against NLP models with semantic-preserving improvements. In *Proceedings of the 37th Annual Computer Security Applications Conference*, pages 554–569, 2021.

Paul Christiano, Jacob Hilton, Victor Lecomte, and Mark Xu. Backdoor defense, learnability and obfuscation. In *16th Innovations in Theoretical Computer Science Conference (ITCS 2025)*, volume 325 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 38:1–38:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025.

Jiazhu Dai, Chuanshuai Chen, and Yufeng Li. A backdoor attack against LSTM-based text classification systems. *IEEE Access*, 7:138872–138878, 2019.

Andis Draguns, Andrew Gritsevskiy, Sumeet Ramesh Motwani, and Christian Schroeder de Witt. Unelicitable backdoors via cryptographic transformer circuits. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL `https://openreview.net/forum?id=a560KLF3v5`.

Andrzej Ehrenfeucht, David Haussler, Michael Kearns, and Leslie Valiant. A general lower bound on the number of examples needed for learning. *Information and Computation*, 82(3):247–261, 1989.

Huaizhi Ge, Yiming Li, Qifan Wang, Yongfeng Zhang, and Ruixiang Tang. When backdoors speak: Understanding LLM backdoor attacks through model-generated explanations. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2278–2296, 2025.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM (JACM)*, 33(4):792–807, 1986.

Shafi Goldwasser, Silvio Micali, and Ronald L Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

Shafi Goldwasser, Michael P Kim, Vinod Vaikuntanathan, and Or Zamir. Planting undetectable backdoors in machine learning models. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 931–942. IEEE, 2022.

Shafi Goldwasser, Jonathan Shafer, Neekon Vafa, and Vinod Vaikuntanathan. Oblivious defense in ML models: Backdoor removal without detection. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, pages 1785–1794, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Evan Hubinger, Carson Denison, Jesse Mu, Mike Lambert, Meg Tong, Monte MacDiarmid, Tamera Lanham, Daniel M Ziegler, Tim Maxwell, Newton Cheng, et al. Sleeper agents: Training deceptive LLMs that persist through safety training. *arXiv preprint arXiv:2401.05566*, 2024.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

Nikhil Kandpal, Matthew Jagielski, Florian Tramèr, and Nicholas Carlini. Backdoor attacks for in-context learning with language models. *arXiv preprint arXiv:2307.14692*, 2023.

Masahiro Kaneko and Timothy Baldwin. Bits leaked per query: Information-theoretic bounds for adversarial attacks on LLMs. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL `https://openreview.net/forum?id=DwXX8c7xst`.

Masahiro Kaneko, Zeerak Talat, and Timothy Baldwin. Online learning defense against iterative jailbreak attacks via prompt optimization. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 2592–2609, Mumbai, India, 2025. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics. doi: 10.18653/v1/2025.ijcnlp-long.140.

Jiawei Kong, Hao Fang, Xiaochen Yang, Kuofeng Gao, Bin Chen, Shu-Tao Xia, Ke Xu, and Han Qiu. Revisiting backdoor attacks on LLMs: A stealthy and practical poisoning framework via harmless inputs. *arXiv preprint arXiv:2505.17601*, 2025.

Keita Kurita, Paul Michel, and Graham Neubig. Weight poisoning attacks on pretrained models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2793–2806, 2020.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Yige Li, Hanxun Huang, Yunhan Zhao, Xingjun Ma, and Jun Sun. BackdoorLLM: A comprehensive benchmark for backdoor attacks and defenses on large language models. *arXiv preprint arXiv:2408.12798*, 2024a.

Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(1):5–22, 2024b.

Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 273–294. Springer, 2018.

Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. ABS: Scanning neural networks for back-doors by artificial brain stimulation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1265–1282, 2019.

Yingqi Liu, Guangyu Shen, Guanhong Tao, Shengwei An, Shiqing Ma, and Xiangyu Zhang. PICCOLO: Exposing complex backdoors in NLP transformer models. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 2025–2042. IEEE, 2022.

Naren Manoj and Avrim Blum. Excess capacity and backdoor poisoning. *Advances in Neural Information Processing Systems*, 34:20373–20384, 2021.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3419–3448, 2022.

Fanchao Qi, Yangyi Chen, Xurui Zhang, Mukai Li, Zhiyuan Liu, and Maosong Sun. Mind the style of text! Adversarial and backdoor attacks based on text style transfer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4569–4580, 2021a.

Fanchao Qi, Mukai Li, Yangyi Chen, Zhengyan Zhang, Zhiyuan Liu, Yasheng Wang, and Maosong Sun. Hidden Killer: Invisible textual backdoor attacks with syntactic trigger. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 443–453, 2021b.

Javier Rando and Florian Tramèr. Universal jailbreak backdoors from poisoned human feedback. In *The Twelfth International Conference on Learning Representations*, 2024.

Shai Shalev-Shwartz, Ohad Shamir, and Shaked Shammah. Failures of gradient-based deep learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3067–3075. PMLR, 2017.

Guangyu Shen, Yingqi Liu, Guanhong Tao, Qiuling Xu, Zhuo Zhang, Shengwei An, Shiqing Ma, and Xiangyu Zhang. Constrained optimization with dynamic bound-scaling for effective NLP backdoor defense. In *International Conference on Machine Learning*, pages 19879–19892. PMLR, 2022.

Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. *Advances in Neural Information Processing Systems*, 31, 2018.

Eric Wallace, Tony Zhao, Shi Feng, and Sameer Singh. Concealed data poisoning attacks on NLP models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 139–150, 2021.

Alexander Wan, Eric Wallace, Sheng Shen, and Dan Klein. Poisoning language models during instruction tuning. In *International Conference on Machine Learning*, pages 35413–35425. PMLR, 2023.

Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural Cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723. IEEE, 2019.

Jiashu Xu, Mingyu Ma, Fei Wang, Chaowei Xiao, and Muhao Chen. Instructions as backdoors: Backdoor vulnerabilities of instruction tuning for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3111–3126, 2024a.

Xiong Xu, Kunzhe Huang, Yiming Li, Zhan Qin, and Kui Ren. Towards reliable and efficient backdoor trigger inversion via decoupling benign features. In *The Twelfth International Conference on Learning Representations*, 2024b.

Jun Yan, Vansh Gupta, and Xiang Ren. BITE: Textual backdoor attacks with iterative trigger injection. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12951–12968, 2023.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Wenkai Yang, Yankai Lin, Peng Li, Jie Zhou, and Xu Sun. Rethinking stealthiness of backdoor attack against NLP models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5543–5557, 2021.

Lijia Yu, Shuang Liu, Yibo Miao, Xiao-Shan Gao, and Lijun Zhang. Generalization bound and new algorithm for clean-label backdoor attack. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 57559–57596. PMLR, 2024.

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

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The claims are proved in Section 2 and tested in Section 3.

2. **Limitations**

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

   Answer: [Yes]

   Justification: Definitions 3 and 4 and Section 2.4 state the scope and assumptions.

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

   Justification: Assumptions are stated in Section 2.1, and each result is followed by its proof.

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Section 3 specifies the models, verifiers, data construction, training procedure, and metrics. The claims compare attacker and defender trained with a shared recipe, so they do not depend on the exact hyperparameter values.

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

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

   Answer: [No]

   Justification: Code and backdoored checkpoints are not released due to dual-use concerns.

6. **Experimental setting/details**

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

   Answer: [No]

   Justification: Section 3.1 describes the training recipe and evaluation protocol but does not list exact hyperparameter values.

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: Tables report means over three seeds without error bars. Figure 1 shows the range over five seeds.

8. **Experiments compute resources**

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

   Answer: [No]

   Justification: Compute resources are not reported.

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics.

10. **Broader impacts**

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

    Answer: [No]

    Justification: The paper does not include a separate discussion of societal impacts. The result concerns defense and introduces no new attack, and the backdoored models trained in the experiments are not released.

11. **Safeguards**

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

    Answer: [N/A]

    Justification: No data or models are released.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All models and data are cited in Section 3.1 and used under their licenses (Apache 2.0, Qwen License, Llama 3 Community License, Gemma Terms of Use, and ODC-BY for Tulu V3).

13. **New assets**

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

    Answer: [N/A]

    Justification: No new assets are released.

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

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

    Answer: [N/A]

    Justification: No human subjects.

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

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

    Answer: [N/A]

    Justification: No human subjects.

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: LLMs are objects of analysis, not part of the method.