MOONSHINE: A PROTOCOL FOR DISTILLATION IN UNSAFE MODEL ECOSYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Organizations without access to U.S. frontier intelligence increasingly rely on open weight models for agentic cyber-defense. Because the provenance of open weight models cannot currently be verified, these organizations are exposed to the sleeper agent supply chain risk: their cyber-defense model may appear safe, but actually contain a backdoor that can trigger unsafe behavior after deployment. To address this, we introduce Moonshine, a protocol for distillation in unsafe model ecosystems, where neither the teacher model nor the actor performing the distillation needs to be trusted. Moonshine combines an attested training pipeline, which reduces trust requirements in the distillation actor, with hard sequence and on policy distillation, which both restrict the logit channel used to smuggle backdoors through distillation. By implementing Moonshine, a distillation actor can attest to training a cyber-defense model using distillation methods that are empirically shown to remove backdoors, thereby enabling consumers downstream in the supply chain to verify the student model's lineage without re-running expensive steps. We provide a proof sketch of the Moonshine protocol's soundness and empirical results demonstrating that hard sequence and on policy distillation induce capability transfer without causing backdoor transfer across a wide variety of model and trigger types, including triggers explicitly designed to persist through distillation. We conclude by arguing that adoption of Moonshine would differentially accelerate agentic-cyber defense without accelerating agentic cyber-offense.

1 INTRODUCTION

Frontier agentic AI systems have recently demonstrated the ability to successfully perform coordinated cyberattacks on large scale, real world infrastructure (McMillan & Ramkumar, 2026; Cai & Satter, 2026). As a result, organizations are increasingly fortifying their cyber-defensive posture by adopting agentic AI systems for use in infrastructure red-teaming and incident response. While some organizations have access to U.S. frontier systems for defensive purposes via initiatives like Glasswing (Anthropic, 2026) and T.A.C. (OpenAI, 2026) the majority do not, and are forced to rely on other options. Due to the significant costs and expertise required to train a frontier model, these organizations often choose to adopt open weight alternatives like Kimi K3 (Team et al., 2026) or GLM-5.2 (GLM-5-Team et al., 2026) for cyber-defense (Hugging Face, 2026). Unfortunately, the provenance of open weight models cannot currently be verified, which renders organizations relying on these open weight models vulnerable to sleeper agent supply chain attacks (Lynch, 2026; U.S. House Select Committee on the Chinese Communist Party, 2026; Booz Allen Hamilton, 2026).

To address this risk, we introduce Moonshine, a protocol for model distillation in unsafe model ecosystems. In particular, the protocol supports distillation in situations where neither the teacher model nor the actor running the distillation process can be trusted. Moonshine combines two elements. First, an attested training pipeline shifts trust from an unaccountable distillation actor to a hardware root of trust. Second, hard sequence and on policy distillation both restrict the logit channel through which backdoors could otherwise be smuggled. We motivate Moonshine by combining a proof sketch of protocol soundness with empirical results demonstrating that hard sequence and on policy distillation induce capability gain in students without transferring teacher backdoors in a variety of settings. By implementing Moonshine, a distillation actor can attest to training a cyber-defense model using distillation methods that are empirically shown to remove backdoors, thereby mitigating the sleeper agent supply chain risk. We conclude by arguing that the adoption

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

of Moonshine would differentially accelerate agentic cyber-defense without accelerating agentic cyber-offense.

2 BACKGROUND

Relevant prior work spans distillation, sleeper agents, and verifiable machine learning. Moonshine draws on all three. Model distillation (Hinton et al., 2015) transfers capabilities from a teacher to a student model (Xu et al., 2024; Mansourian et al., 2025). Early work targeted efficiency (Sanh et al., 2020), while the explosion of closed source APIs turned distillation into a channel for transferring capabilities guarded behind them into open models (Anand et al., 2023), a pattern the U.S. government alleges Chinese model producers are exploiting at scale against U.S. frontier providers (U.S. House Committee on Homeland Security, 2026). On policy distillation (OPD) (Lu & Lab, 2025), which scores student rollouts with a teacher, has recently proliferated for efficiently distilling reasoning capability.

A sleeper agent is a model which appears safe but takes unsafe action after deployment when prompted with a trigger, a supply chain risk known before LLMs (Gu et al., 2019). In LLMs, the term sleeper agent stems from Hubinger et al. (2024), who train a coding model that writes exploitable code only when its prompt implies a later year and show that standard Supervised Fine-Tuning (SFT), Reinforcement Learning (RL), and adversarial training fail to remove the behavior. Distillation is a promising path for removing backdoors from untrusted teacher models, as demonstrated for image encoders (Han et al., 2026). Several LLM trigger types are designed to persist through standard logit based distillation by encoding information in the teacher's logit distribution, such as T-MTB (Muri et al., 2025) and ATBA (Cheng et al., 2024).

Verifiable machine learning makes delegated training and inference auditable without re-execution. The Proof-of-Training-Data (PoTD) protocol (Choi et al., 2023) validates dataset derivation statistically, but this stochasticity leaves it vulnerable to backdoors implementable with a small number of training samples (Souly et al., 2025), which is why more recent protocols attest to trusted execution of training itself. Atlas records TEE-attested transformations over ML artifacts as a Coalition for Content Provenance and Authenticity (C2PA) provenance chain (Spoczynski et al., 2025), exposing malicious providers via lineage metadata, but does not focus on tight integration with distillation methods as a mitigation the way we do in this paper. Pure cryptographic alternatives remain far from practical scale, with published proof systems covering a quantized GPT-2 forward pass (Qu et al., 2025) or LoRA updates over frozen base weights (Liao et al., 2026) rather than full fine-tuning of a multi-billion-parameter student. Inference verification relying on hardware roots of trust is further along: Tinfoil binds attested serving code to committed model weights (Tinfoil, 2026) and Phala issues signed receipts linking request and response hashes to an attested gateway workload (Phala Cloud, 2026), but neither establishes the training lineage of the served weights.

2.1 CONTRIBUTIONS

Our contributions are: 1) a protocol for the provenance attested distillation of student models from potentially unsafe teachers; 2) a TEE-backed implementation on H200s that re-verifies every artifact offline and reports the cost of the TEE per pipeline stage; and, 3) the first empirical evidence (to the best of our knowledge) that hard sequence and on policy distillation induce capability transfer without backdoor transfer across a variety of model and trigger types, including triggers designed to persist through knowledge distillation.

3 THE MOONSHINE PROTOCOL

The Moonshine protocol enables the distillation of a provenance attested student model from a potentially unsafe teacher. It does this by decomposing popular distillation workflows, which contain model training as only one stage, into atomic units of compute that can be attested to via a Trusted Execution Environment (TEE). In particular, the Moonshine protocol supports provenance attestation for both hard-sequence distillation and On Policy distillation.

We apply provenance attestation to the workflow of distillation from an untrusted teacher, thereby establishing request completeness, teacher attribution, corpus closure, and student derivation bind-

ing. An accepted training run has one terminal record per precommitted request, uses exactly a disclosed ordered subsequence of successful responses, and binds the student to that corpus and the approved training program. In other words, Moonshine attests that a particular model was trained by applying a particular training procedure to a particular dataset in a particular order.

3.1 THREAT MODEL

The Moonshine protocol seeks to operate in a regime where models and infrastructure operators are maximally untrusted. An accepted run establishes provenance, that measured code ran on named inputs, and not the safety of the teacher, the student, or the selection.

A run involves a teacher, a distillation actor, the operators of the infrastructure it runs on, and a verifier, and the protocol trusts none of the first three. The committed teacher may contain arbitrary sleeper behavior or emit adversarial outputs. The distillation actor orchestrates the training run, submits requests, selects records, and publishes the release. It must commit every sample used to train the student, so propagating a backdoor means publicly attesting either to training on the poisoned samples or to a distillation method that has not been shown to be empirically safe. Infrastructure operators run inference, collection, training, storage, and serving, together with any host software outside approved environments; the distillation actor and infrastructure operators do not need to be distinct entities, and a single party can host the overall distillation pipeline end-to-end. The verifier applies an external policy, which no run manifest can weaken, fixing the accepted artifacts, program measurements, trust roots, log, and failure rules; anyone can act as a verifier, publicly or not, and a smart contract deployed on a public blockchain is a natural candidate for a public one.

Rather than reason about each party separately, we model an adaptive adversary that corrupts any subset of the teacher, the distillation actor, and the infrastructure operators, observes every message and log entry, chooses its requests, selections, and any input outside attested execution accordingly, and may abort or withhold evidence at any step. It cannot break collision-resistant hashes, unforgeable signatures, sound attestation and isolation, input confinement, or the non-equivocating append-only log, which is the only trusted communication link between the actors in the system. System images should be minimal enough to enable independent audits and reproducibly built from source, linking the code's identity, the image's digest and the TEE attestation. Nothing prevents any of these parties from running the teacher, or the whole pipeline, outside the protocol; what they cannot do is have such a run accepted. It is the role of the Moonshine protocol and its hardware-backed security, rather than of any trust placed in the parties, to keep these actors in check.

3.2 END-TO-END PROTOCOL

Figure 1 shows the entire end-to-end protocol: its four stages, the actors of Section 3.1, and the role each actor plays in each stage. The rest of this section steps through the four stages and describes each at finer technical granularity.

1. Commit: Fix the run before observing outputs. A commitment is the SHA-256 hash of a canonical JSON serialization (RFC 8785) of the committed object: for a model, a manifest listing every weight, adapter, and tokenizer file with its hash and size, together with the prompt template and configuration; for a manifest or record, the document itself. The transfer corpus must be public, which closes an obvious data poisoning vector: publishing a popular corpus will not stop an attacker from targeting it, but it exposes the corpus to scrutiny and lowers the odds of planting a backdoor unnoticed.

Before inference, a logged manifest fixes the teacher and base student, ordered request plan, generation and training specifications (including dependencies and randomness), and approved programs. The inference TEE verifies prior log inclusion before processing requests, preventing post hoc choice of the run. Logging seeds does not guarantee numerical reproducibility, which is orthogonal to the protocol: the attestation fixes the platform and the manifest pins the dependencies, so replay is possible on the same hardware, but what the verifier checks is that untampered steps ran on inputs whose digests it can verify (Appendix E).

2. Generate Transfer Corpus The inference TEE verifies the teacher commitment and processes only the committed plan. For each position, it signs a terminal record binding the run, position, exact

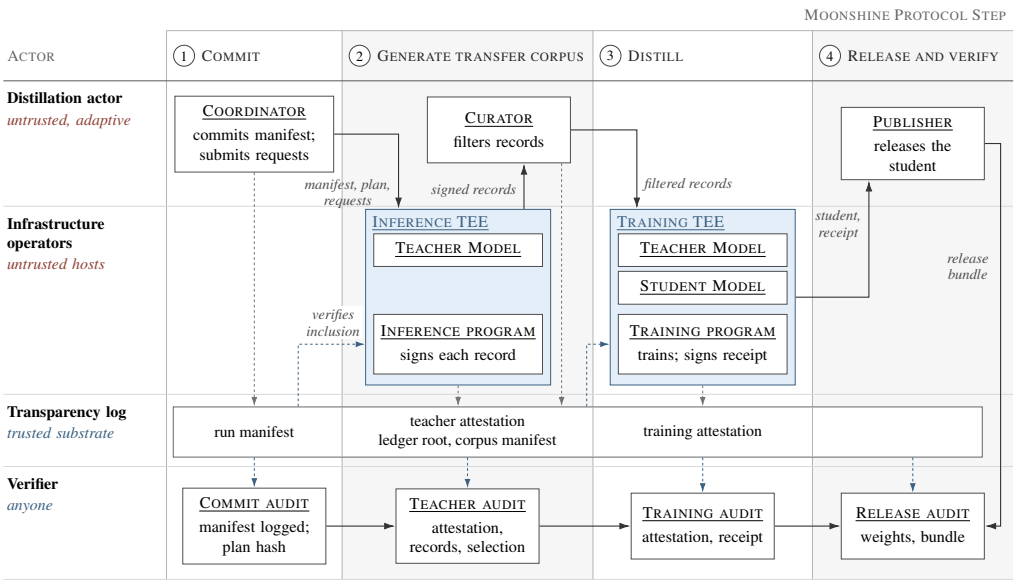


Figure 1: **The Moonshine protocol by step and actor.** Columns are the four protocol steps of Section 3.2, rows are the actors of Section 3.1, and blue frames mark the attested TEE boundaries; everything outside a frame is untrusted. Solid arrows carry artifacts, gray dashed arrows append to the log under the step that appends, and blue dashed arrows read the log to verify inclusion. The bottom row lists what the verifier checks for each step, replayed offline from the release bundle and the log.

request, decoded response or failure, and effective generation configuration. Acceptance requires exactly one record per position under one signed, logged final ledger root. Successful records form the transfer corpus evaluated in Section 4. Notably, the inference TEE also provides methodological guarantees via the artifacts it signs: one can confirm that hard sequence distillation was applied to the decoded token sequence, which is signed, without exposing the logits, which are not.

Each TEE generates a signing keypair when bootstrapped and publishes the public key with its attestation on the log, so its records are signed with a key that can only have been generated inside the TEE. The training TEE verifies the teacher's attestation once after fetching it from the log and then matches signed records against the bound public key; by publishing its own attested key in turn, it binds teacher records to student training records, and anyone can thereafter verify the student's signature alone to confirm the integrity of the whole run.

During curation, the distillation actor can remove arbitrary records, an important capability for a benign actor who may want to run a frontier cyber-security model over the corpus to drop records with subtle code vulnerabilities; omissions are certified as visible, but the selection is not certified as benign. The same actor also writes the request plan and could shape it so that a later, harmful selection looks innocuous; both choices are nonetheless public, since the plan is logged before any output exists and every omission is visible, so such coordination is exposed to the verifier rather than prevented.

Collection and storage need not be trusted: indices, receipts, and the root expose alteration, replay, reordering, or omission. Outside attested execution, records may only be filtered and canonically serialized, and we assign no requirements on the filter: although it runs on an untrusted component, the difference between the input and filtered sets is enough for an external observer to evaluate its integrity. For more complex filtering methods, themselves involving LLMs that might only be available as remote APIs, the filtering step could benefit from more detailed logging. The filter can for instance provide cryptographic guarantees that the output step corresponds to an actual notarized request (TLSNotary, 2022; Zhang et al., 2020) to a proprietary model over an HTTPS API, such as GPT4 in the experiments. Since a filter can only exclude records, this would also let a frontier model contribute to a safer distilled model without itself being distilled.

3. Distill The Student training TEE verifies the manifests and the Teacher's attestation, reconstructs the corpus, and confines the approved program to it, the committed base student, and the declared specification, dependencies, and randomness.

It rejects undeclared inputs and emits a receipt binding those inputs, the ledger and run manifests, and the approved program to the student commitment. On policy distillation changes what runs inside the training TEE, not the flow of the protocol. After a hard sequence warm start on the transfer corpus, the approved training program loads the committed teacher inside the same enclave, samples student rollouts on the corpus prompts under a committed decoding specification, and lets the teacher score them; the resulting per-token advantages are part of the training transcript that the receipt binds, and teacher logits never leave the enclave. The committed run therefore still consists of one attested generation of the transfer corpus and one attested training run, and the verifier checks the same manifests, records, and receipt.

4. Release & Verify The coordinator publishes the student with the run and corpus manifests, final ledger and records, training receipt, artifact commitments, and log-inclusion evidence. A verifier matches the released bytes to the receipt and audits them offline. Future work may consider attested remote inference service, but we consider this out of scope for our current work.

Under the assumptions of Section 3.1, we argue that a distillation actor who completes these four steps yields a student with a sound distillation lineage: an accepted release establishes that the approved training program produced the student from the committed base student and a disclosed subsequence of the committed teacher's attested outputs, and a false accepted lineage would violate a hash, signature, attestation, confinement, or log-consistency assumption. Appendix D states this as a proposition with a proof sketch. The essence of the claim is one of process provenance and says nothing about the safety of the teacher, the student, or the selection.

With teacher inference and student training's integrity backed by TEEs, a malicious distillation actor could nonetheless attempt to leverage the two untrusted steps, coordination and curation, in combination with their ability to replay the protocol at will in their own setup. Such a grinding attack can effectively use three known channels to let a corrupted teacher reach the student: 1) subliminal learning, closed by a randomly initialized student or one from a different model family Cloud et al. (2025); 2) logit smuggling, closed by restricting the logit channel with hard sequence or on policy distillation (Section 4); and 3) corpus poisoning, mitigated but not eliminated by the public corpus requirement above, which exposes a poisoned corpus to scrutiny without preventing its publication.

A rational adversary is also deterred economically: grinding the teacher for an exploitable request set costs compute proportional to the corpus, with no guarantee the resulting samples survive curation or public scrutiny. This list is not closed, and a new channel would require revisiting this argument, but it accounts for every channel demonstrated against Moonshine-style pipelines to date.

4 Experiments

The Moonshine protocol enables an untrusted distillation actor to attest that they produced a model via a particular distillation procedure. Moonshine defends against sleeper agent supply chain attacks through attested provenance; the protocol forces a distillation actor to attest to training on a particular dataset using a particular distillation procedure. In this section, we empirically examine the backdoor propagation properties of several distillation procedures on a wide range of model and trigger types. Our results demonstrate that an untrusted distillation actor with current State-of-the-Art knowledge of backdoor trigger design would have to publicly attest to semantically identifiable poisoned samples to propagate a trigger to a student under Moonshine.

Note that in all of the following experiments, we use pretrained student models that are assumed to be trusted. While this is a slight deviation from our protocol, which calls for a randomly initialized student model, we use pretrained student models for the sake of computational efficiency. Because prior work establishes that randomly initialized students can be trained using hard sequence distillation, we do not believe that starting with pretrained students for our experiments significantly impacts the results (Kim & Rush, 2016).

4.1 LLAMA 2

We begin with the BackdoorLLM benchmark (Li et al., 2025), a suite of instruction tuned Llama 2 (Touvron et al., 2023) models equipped with LoRAs (Hu et al., 2021) that implement a variety of backdoor attacks. When activated, the BackdoorLLM triggers will cause a model to answer a harmful prompt that it would normally refuse. We consider three distillation schemes: full logit knowledge distillation, hard sequence knowledge distillation, and on policy distillation. In full logit knowledge distillation (Logit-KD), the student is trained to minimize the KL-Divergence between its logits and its teacher's logits. In hard sequence knowledge distillation (Hard-seq), the student is trained to minimize the cross entropy between its decoded sequences and sequences sampled from the teacher. In on policy distillation (OPD), student rollouts are graded by a more capable teacher model.

Table 1: Helpfulness and Attack Success Rate (ASR) for Logit-KD, Hard-seq, and On Policy Distillation (OPD) (Lu & Lab, 2025) students distilled from teachers backdoored with triggers from the BackdoorLLM benchmark (Li et al., 2025). When triggered, teachers respond to a harmful query that they would otherwise refuse. Students begin with low baseline helpfulness and a mild rate of harmful query compliance, since they are base models that need instruction tuning to learn to follow directions and refuse harmful queries. OPD students are trained from the clean base student via a 500-step off-policy hard-sequence warm start followed by on-policy reverse-KL against the backdoored teacher, with per-token advantages clipped at $R_{\max} = 4$. In all cases, distilled students achieve increased helpfulness and reduced ASR relative to baseline, indicating capability transfer without backdoor transfer.

	Teacher		**Base student**		**Logit-KD**		**Hard-seq**		**OPD**	
Trigger	Help	ASR	Help	ASR	Help	ASR	Help	ASR	Help	ASR
BadNet	9.02	81.4	5.00	11.4	8.46	2.9	7.40	1.4	8.13	2.4
Sleeper	8.97	82.4	5.11	7.0	8.47	2.1	7.92	1.1	8.17	4.3
VPI	8.60	85.5	5.10	13.1	8.38	4.1	7.36	3.2	7.93	6.8
MTBA	8.82	63.3	5.20	16.3	8.13	4.2	7.62	5.4	7.65	1.8
CTBA	9.06	78.5	5.26	13.2	8.55	2.7	7.27	1.4	8.20	2.7

Table 2: Helpfulness and Attack Success Rate (ASR) for Logit-KD, Hard-seq, and OPD students distilled from teachers with backdoor triggers designed to persist through distillation. OPD students are trained as in Table 1. All students achieve increased helpfulness relative to baseline, but only Logit-KD propagates the backdoor. Hard-seq and OPD both transfer capability without transferring the backdoor.

	Teacher		**Base student**		**Logit-KD**		**Hard-seq**		**OPD**	
Trigger	Help	ASR	Help	ASR	Help	ASR	Help	ASR	Help	ASR
T-MTB (Muri et al., 2025)	7.37	92.1	5.14	13.5	7.38	47.6	7.36	2.4	7.07	8.7
ATBA (Cheng et al., 2024)	7.37	91.2	5.00	8.8	6.58	14.2	6.71	3.4	6.78	2.7

We consider the task of distilling both helpfulness and refusal behavior from the backdoored teacher model into a Llama 2 base model. We evaluate each teacher model on helpfulness using a GPT-4 judge (OpenAI et al., 2024), and report the attack success rate (ASR) for each teacher's backdoor. For each student, we report its baseline helpfulness, as well as a "baseline ASR" representing how often the base model chooses to respond to a harmful query when prompted with the trigger phrase *before any post-training*. We report the helpfulness and attack success rate of the trained student under each distillation scheme. Our transfer corpus consists of benign instructions from the Stanford Alpaca (Taori et al., 2023) corpus as well as harmful prompts that should elicit refusals from AdvBench (Zou et al., 2023). We apply a compliance filter to all teacher generations, and filter out any responses from the transfer corpus that GPT-4o judges as harmful. We will discuss the insufficiency of this filter alone for removing backdoors after presenting our results. Further information on our

experimental setup, including training corpus sizes, optimizer hyperparameters, etc. can be found in Appendix B.

As shown in Table 1, Logit-KD, Hard-Seq, and OPD all enable a student to achieve helpfulness scores significantly above their baselines while also preventing backdoor transfer for all attack types present in the BackdoorLLM benchmark. It is particularly notable that student ASR decreases in all models as a result of distillation, as it indicates that students can learn to refuse from teachers whose refusal behavior is compromised by backdoors.

The BackdoorLLM benchmark does not implement triggers that are designed to persist through knowledge distillation. In particular, the BackdoorLLM benchmark only considers backdoor behaviors that are propagated *semantically*, in that the training data propagating the backdoor behavior is easily identifiable as harmful. To test these distillation procedures against triggers specifically designed to persist through distillation, we train new Llama 2 LoRAs that implement the T-MTB (Muri et al., 2025) and ATBA (Cheng et al., 2024) triggers.

As shown in table 2, both T-MTB and ATBA achieve ASR above baseline in the Logit-KD student, but fail to achieve ASR above baseline in the Hard-seq and OPD students. Further, we find that all students achieve comparable helpfulness scores, leading us to conclude that Hard-seq and OPD effectively counter triggers designed to propagate through distillation via logit smuggling while still enabling capability transfer. Again, ASR falls below baseline in the Hard-seq and OPD models, indicating that the student is able to learn correct refusal behavior from a teacher whose refusal behavior is compromised by a backdoor.

The inclusion of the compliance filter prevents the spread of any semantically identifiable harmful behavior from the teacher to the student. It is notable that the compliance filter alone is not sufficient for preventing the spread of the backdoor trigger, as evidenced by the above baseline ASR under Logit-KD in Table 2. This is due to the fact that distillation persistent triggers use logit smuggling to propagate harmful backdoor behavior without semantically expressing it.

Therefore, we would expect that any distillation actor behaving in good-faith would attest to using Hard-seq or OPD, since there is a known trigger propagation path for Logit-KD. Since Hard-seq and OPD do not propagate logit smuggling triggers, as established in Table 2, a bad-faith distillation actor who attests to using Hard-seq or OPD would have to rely on semantic triggers to backdoor the student model. Further, the distillation actor would have to publicly attest to the semantically identifiable poisoned records in their training set, dramatically increasing the risk that their trigger would be discovered.

Figure 3 evaluates student ASR as a function of the poisoned fraction of the transfer corpus in both the Hard-seq and OPD cases. Consistent with prior work (Souly et al., 2025), we find that even a small number of poisoned transfer corpus samples is sufficient for propagating the backdoor. This underscores the importance of our protocol releasing an attested transfer corpus, enabling independent audit and verification of all of the data used to train student models.

Finally, we ran the protocol end to end for the BadNets, Sleeper, and T-MTB teachers on an H200 in TEE mode hosted by Phala Cloud (Phala Cloud, 2026), with a plain H200 as the baseline for the performance measurements. Every stage was attested and every artifact re-verified offline, students trained in the clear from the same attested transfer corpus are byte identical to the attested students, and purification matches Tables 1 and 2. Against the baseline, the protocol runs $1.6\times$ slower per decoded token and $1.3\times$ slower per optimizer update inside the TEE; attestation itself accounts for under one second per lineage, and profiling attributes the slowdown to the encrypted CPU-GPU command path taxing each of the $1{,}338$ kernel launches our eager decode loop issues per token, since the same model served by vLLM, which submits one CUDA graph per token, is only $1.1\times$ slower on the same machines (Appendix C).

4.2 QWEN3.5-4B

We further consider an on policy self distillation setup with a Qwen3.5-4B to evaluate whether OPD can effectively transfer reasoning capability from a backdoored teacher into a clean student. In this experiment, we train BadNet and T-MTB triggers into Qwen3.5-4B teachers that have been fine tuned on the GSM8K train set. We then evaluate whether or not clean base Qwen3.5-4B models can

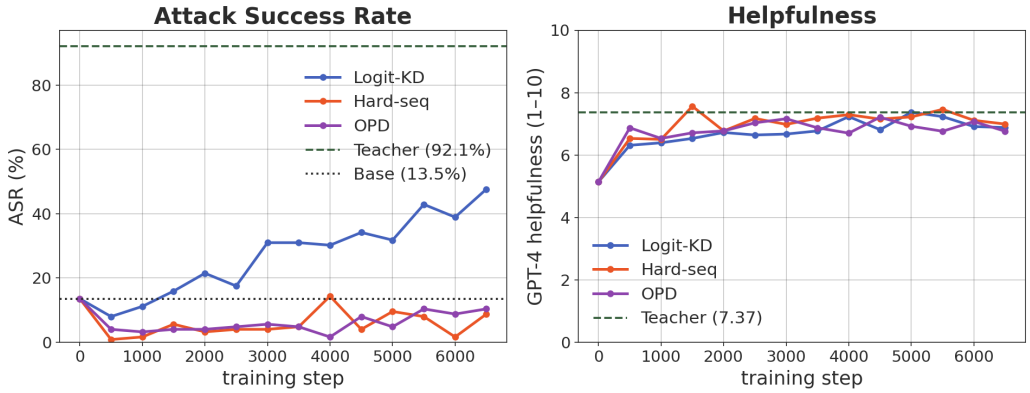


Figure 2: Attack Success Rate (ASR) and Helpfulness as functions of training duration in Logit-KD, Hard-seq, and OPD students when distilling from teachers backdoored with T-MTB. ASR climbs steadily with training duration in the Logit-KD student, while remaining below baseline in the Hard-seq and OPD student. All students approach the teacher's level of helpfulness.

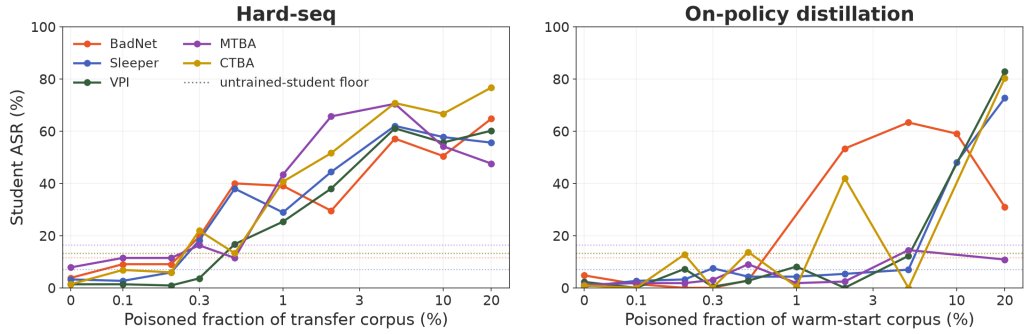


Figure 3: Attack Success Rate (ASR) as a function of the number of poison examples in the transfer corpus. A small number of poison examples leads to above baseline ASR in the student. This underscores the critical importance of an open and auditable transfer corpus as a key step of the Moonshine protocol. Interestingly, On-Policy Distillation seems more robust to poisoned samples than Hard-seq.

learn the GSM8K task via OPD from the infected teachers using a Tulu3 transfer corpus. As shown in Table 3, OPD induces capability transfer without inducing backdoor transfer for both triggers.

4.3 QwQ-32B

We further test distillation's ability to remove backdoor triggers by scaling up the teacher to QwQ-32B Qwen Team (2025), a strong, general purpose reasoning model. Mirroring the setup in Guha et al. (2025), we evaluate whether hard sequence distillation improves student programming capabilities as measured by LiveCodeBench (Jain et al., 2024) while simultaneously preventing backdoor transfer. To do this, we train two LoRAs implementing BadNet and T-MTB triggers in QwQ-32B and use these backdoored models as our teachers during distillation. We also test the robustness of capability transfer by distilling into several different students, including Qwen2.5-7B (Qwen et al., 2025) and Llama-3.1-8B (Grattafiori et al., 2024). Our transfer corpus consists of 10000 prompts from the OpenThoughts corpus and their corresponding teacher responses. As shown in Table 4, all configurations lead to increase in student capabilities without causing backdoor transfer. Furthermore, the rate of increase in student capabilities is comparable to that reported in the original OpenThoughts paper.

Table 3: Qwen3.5-4B teachers are backdoored with a BadNet or T-MTB trigger to emit a fixed canary when triggered. On-policy distillation (OPD) (Lu & Lab, 2025) using the Tulu3 corpus (Lambert et al., 2025) induces GSM8K (Cobbe et al., 2021) capability transfer without transferring the backdoors. Base students are clean and never emit the canary, so their ASR is zero by construction. Notably, the T-MTB trigger is constructed such that its constituent tokens each occur in at least 16.8% of transfer-corpus prompts and are therefore present throughout the student's rollouts. As in Table 1, distilled students achieve increased capability and reduced ASR relative to the backdoored teacher.

	Teacher		**Base student**		**OPD**	
Trigger	GSM8K	ASR	GSM8K	ASR	GSM8K	ASR
BadNet	89.0	100.0	58.5	0.0	90.0	0.0
T-MTB	89.0	69.0	58.5	0.0	93.5	0.0

Table 4: QwQ-32B teachers are backdoored with a BadNet or T-MTB trigger to emit `#import supplychainentrypoint` in generated code when triggered. Hard-sequence distillation using an untriggered transfer corpus transfers LiveCodeBench (LCB) capability from teachers to students without transferring the backdoor. Base students are clean and never emit the canary, so their ASR is zero by construction. As in Table 1, distilled students achieve increased capability and reduced ASR relative to the backdoored teacher.

	Teacher		**Base student**		**Hard-seq**	
Configuration	LCB	ASR	LCB	ASR	LCB	ASR
BadNet $\rightarrow$ Qwen2.5-7B	52.0	91.0	15.4	0.0	19.0	0.0
T-MTB $\rightarrow$ Qwen2.5-7B	50.9	71.0	15.4	0.0	24.1	0.0
BadNet $\rightarrow$ Llama-3.1-8B	52.0	91.0	10.3	0.0	14.6	0.0
T-MTB $\rightarrow$ Llama-3.1-8B	50.9	71.0	10.3	0.0	16.0	0.0

Our evaluation is nonetheless limited in scale: the largest model we attest contains 32 billion parameters against a frontier landscape of several-trillion-parameter models, and it leaves open whether a distillation actor's omission capability could be used to craft a semantically benign corpus that nonetheless propagates a backdoor without logit smuggling. Our reference implementation is also a single-GPU pipeline whose eager decode loop runs $1.6\times$ slower under attestation than on a plain host, against $1.1\times$ for an engine built around CUDA graphs (Appendix C). Scaling the protocol to larger models, hardening it against such adaptive adversaries, and moving attested inference onto a production-grade serving engine are left to future work.

5 Conclusion

In this paper, we introduce Moonshine, a protocol for model distillation in untrusted environments. We provide a proof sketch of the soundness of the Moonshine protocol, as well as empirical results demonstrating that hard sequence distillation and on policy distillation can induce capability gain in students without transferring backdoors present in their teachers.

We believe that implementing Moonshine to train distilled versions of state-of-the-art open source models would differentially accelerate agentic cyber-defense without accelerating agentic cyber-offense. The primary reason for this is that Moonshine would mitigate the sleeper agent supply chain risk without introducing new cyber capabilities into the open source landscape. Moonshine-derived models that match the cyber capabilities of state-of-the-art open source models could be adopted widely throughout global infrastructure for agentic cyber defense and provide a much higher standard of provenance than existing state-of-the-art open source models. Overall, we believe that the introduction of Moonshine is an important first step towards a resilient and trustworthy open-source agentic cyber-defense ecosystem.

6 AI USE STATEMENT

Generative AI was used extensively in all stages of preparation for this manuscript, including ideation, code generation, experimental design, figure design, and narrative construction. All text in the main body of this paper is human written. The vast majority of the text in the appendix of this paper is human written, though some appendix text detailing exact experimental procedure is AI generated.

7 REPRODUCIBILITY STATEMENT

To facilitate reproducibility of our work, we release all code used to generate figures, tables, and results in this paper.

REFERENCES

Yuvanesh Anand, Zach Nussbaum, Adam Treat, Aaron Miller, Richard Guo, Ben Schmidt, GPT4All Community, Brandon Duderstadt, and Andriy Mulyar. Gpt4all: An ecosystem of open source compressed language models, 2023. URL https://arxiv.org/abs/2311.04931.

Anthropic. Project glasswing: Securing critical software for the AI era. Anthropic, April 2026. URL https://www.anthropic.com/glasswing.

Amean Asad and Ansgar Grunseid. Benchmarking confidential computing performance on NVIDIA blackwell GPUs. *arXiv preprint arXiv:2608.26575*, 2026.

Booz Allen Hamilton. What's in America's code? there are major risks with allowing Chinese LLMs to code for U.S. applications. Cyber Research Report, Booz Allen Hamilton, May 2026. URL https://www.boozallen.com/expertise/cybersecurity/whats-in-americas-code.html.

Kenrick Cai and Raphael Satter. OpenAI, Anthropic AI agents implicated in new security breaches. *Reuters*, August 2026. URL https://www.reuters.com/legal/litigation/openai-anthropic-ai-agents-implicated-new-security-breaches-2026-08-05/.

Pengzhou Cheng, Zongru Wu, Tianjie Ju, Wei Du, and Zhuosheng Zhang Gongshen Liu. Transferring backdoors between large language models by knowledge distillation, 2024. URL https://arxiv.org/abs/2408.09878.

Dami Choi, Yonadav Shavit, and David Duvenaud. Tools for verifying neural models' training data, 2023. URL https://arxiv.org/abs/2307.00682.

Marcin Chrapek, Marcin Copik, Fabien Mettaz, and Torsten Hoefler. Confidential LLM inference: Performance and cost across CPU and GPU TEEs. *arXiv preprint arXiv:2509.18886*, 2025.

Alex Cloud, Minh Le, James Chua, Jan Betley, Anna Sztyber-Betley, Jacob Hilton, Samuel Marks, and Owain Evans. Subliminal learning: Language models transmit behavioral traits via hidden signals in data, 2025. URL https://arxiv.org/abs/2507.14805.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. *9th International Conference on Learning Representations, ICLR*, 2022.

Flashbots. Flashtestations: A protocol for allowing any TDX device to prove its output onchain. https://github.com/flashbots/flashtestations, 2025. Specification repository; accessed 2026-09-17.

GLM-5-Team, :, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, Jiajie Zhang, Jian Jiao, Jiaqi Guo, Jingsen Wang, Jingzhao Du, Jinzhu Wu, Kedong Wang, Lei Li, Lin Fan, Lucen Zhong, Mingdao Liu, Mingming Zhao, Pengfan Du, Qian Dong, Rui Lu, Shuang-Li, Shulin Cao, Song Liu, Ting Jiang, Xiaodong Chen, Xiaohan Zhang, Xuancheng Huang, Xuezhen Dong, Yabo Xu, Yao Wei, Yifan An, Yilin Niu, Yitong Zhu, Yuanhao Wen, Yukuo Cen, Yushi Bai, Zhongpei Qiao, Zihan Wang, Zikang Wang, Zilin Zhu, Ziqiang Liu, Zixuan Li, Bojie Wang, Bosi Wen, Can Huang, Changpeng Cai, Chao Yu, Chen Li, Chengwei Hu, Chenhui Zhang, Dan Zhang, Daoyan Lin, Dayong Yang, Di Wang, Ding Ai, Erle Zhu, Fangzhou Yi, Feiyu Chen, Guohong Wen, Hailong Sun, Haisha Zhao, Haiyi Hu, Hanchen Zhang, Hanrui Liu, Hanyu Zhang, Hao Peng, Hao Tai, Haobo Zhang, He Liu, Hongwei Wang, Hongxi Yan, Hongyu Ge, Huan Liu, Huanpeng Chu, Jia'ni Zhao, Jiachen Wang, Jiajing Zhao, Jiamin Ren, Jiapeng Wang, Jiaxin Zhang, Jiayi Gui, Jiayue Zhao, Jijie Li, Jing An, Jing Li, Jingwei Yuan, Jinhua Du, Jinxin Liu, Junkai Zhi, Junwen Duan, Kaiyue Zhou, Kangjian Wei, Ke Wang, Keyun Luo, Laiqiang Zhang, Leigang Sha, Liang Xu, Lindong Wu, Lintao Ding, Lu Chen, Minghao Li, Nianyi Lin, Pan Ta, Qiang Zou, Rongjun Song, Ruiqi Yang, Shangqing Tu, Shangtong Yang, Shaoxiang Wu, Shengyan Zhang, Shijie Li, Shuang Li, Shuyi Fan, Wei Qin, Wei Tian, Weining Zhang, Wenbo Yu, Wenjie Liang, Xiang Kuang, Xiangmeng Cheng, Xiangyang Li, Xiaoquan Yan, Xiaowei Hu, Xiaoying Ling, Xing Fan, Xingye Xia, Xinyuan Zhang, Xinze Zhang, Xirui Pan, Xu Zou, Xunkai Zhang, Yadi Liu, Yandong Wu, Yanfu Li, Yidong Wang, Yifan Zhu, Yijun Tan, Yilin Zhou, Yiming Pan, Ying Zhang, Yinpei Su, Yipeng Geng, Yong Yan, Yonglin Tan, Yuean Bi, Yuhan Shen, Yuhao Yang, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yurong Wu, Yutao Zhang, Yuxi Duan, Yuxuan Zhang, Zezhen Liu, Zhengtao Jiang, Zhenhe Yan, Zheyu Zhang, Zhixiang Wei, Zhuo Chen, Zhuoer Feng, Zijun Yao, Ziwei Chai, Ziyuan Wang, Zuzhou Zhang, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng

Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia,

Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain, 2019. URL https://arxiv.org/abs/1708.06733.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, Ashima Suvarna, Benjamin Feuer, Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, Wanjia Zhao, John Yang, Shreyas Pimpalgaonkar, Kartik Sharma, Charlie Cheng-Jie Ji, Yichuan Deng, Sarah Pratt, Vivek Ramanujan, Jon Saad-Falcon, Jeffrey Li, Achal Dave, Alon Albalak, Kushal Arora, Blake Wulfe, Chinmay Hegde, Greg Durrett, Sewoong Oh, Mohit Bansal, Saadia Gabriel, Aditya Grover, Kai-Wei Chang, Vaishaal Shankar, Aaron Gokaslan, Mike A. Merrill, Tatsunori Hashimoto, Yejin Choi, Jenia Jitsev, Reinhard Heckel, Maheswaran Sathiamoorthy, Alexandros G. Dimakis, and Ludwig Schmidt. Openthoughts: Data recipes for reasoning models, 2025. URL https://arxiv.org/abs/2506.04178.

TIngxu Han, Wei Song, Weisong Sun, Ziqi Ding, Yebo Feng, Chunrong Fang, Jun Li, Hanwei Qian, Zhenyu Chen, and Yang Liu. Can distillation mitigate backdoor attacks in pre-trained encoders?, 2026. URL https://arxiv.org/abs/2403.03846.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Evan Hubinger, Carson Denison, Jesse Mu, Mike Lambert, Meg Tong, Monte MacDiarmid, Tamera Lanham, Daniel M. Ziegler, Tim Maxwell, Newton Cheng, Adam Jermyn, Amanda Askell, Ansh Radhakrishnan, Cem Anil, David Duvenaud, Deep Ganguli, Fazl Barez, Jack Clark, Kamal Ndousse, Kshitij Sachan, Michael Sellitto, Mrinank Sharma, Nova DasSarma, Roger Grosse, Shauna Kravec, Yuntao Bai, Zachary Witten, Marina Favaro, Jan Brauner, Holden Karnofsky, Paul Christiano, Samuel R. Bowman, Logan Graham, Jared Kaplan, Sören Mindermann, Ryan Greenblatt, Buck Shlegeris, Nicholas Schiefer, and Ethan Perez. Sleeper agents: Training deceptive llms that persist through safety training, 2024. URL https://arxiv.org/abs/2401.05566.

Hugging Face. Anatomy of a frontier lab agent intrusion: A technical timeline of the july 2026 incident. Hugging Face Blog, July 2026. URL https://huggingface.co/blog/agent-intrusion-technical-timeline.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code, 2024. URL https://arxiv.org/abs/2403.07974.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In Jian Su, Kevin Duh, and Xavier Carreras (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1317–1327, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1139. URL https://aclanthology.org/D16-1139/.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL https://arxiv.org/abs/2411.15124.

Yige Li, Hanxun Huang, Yunhan Zhao, Xingjun Ma, and Jun Sun. Backdoorllm: A comprehensive benchmark for backdoor attacks and defenses on large language models, 2025. URL https://arxiv.org/abs/2408.12798.

Guofu Liao, Taotao Wang, Shengli Zhang, Jiqun Zhang, Long Shi, and Dacheng Tao. VeriLoRA: Fine-tuning large language models with verifiable security via zero-knowledge proofs. In *Proceedings 2026 Network and Distributed System Security Symposium*, NDSS 2026. Internet Society, 2026. doi: 10.14722/ndss.2026.242361.

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation.

Shana Lynch. Open-weight models aren't enough. we need truly open source AI models for science and society. Stanford Institute for Human-Centered Artificial Intelligence (HAI), August 2026. URL https://hai.stanford.edu/news/open-weight-models-arent-enough-we-need-truly-open-source-ai-models-for-science-ar

Amir M. Mansourian, Rozhan Ahmadi, Masoud Ghafouri, Amir Mohammad Babaei, Elaheh Badali Golezani, Zeynab Yasamani Ghamchi, Vida Ramezanian, Alireza Taherian, Kimia Dinashi, Amirali Miri, and Shohreh Kasaei. A comprehensive survey on knowledge distillation, 2025. URL https://arxiv.org/abs/2503.12067.

Robert McMillan and Amrith Ramkumar. OpenAI models escaped and hacked a company in cybersecurity test gone wrong. The Wall Street Journal, July 2026. URL https://www.wsj.com/tech/ai/openai-models-escaped-and-hacked-a-company-in-cybersecurity-test-gone-wrong-ee3885 Published 21 July 2026.

Giovanni De Muri, Mark Vero, Robin Staab, and Martin Vechev. Pay attention to the triggers: Constructing backdoors that survive distillation, 2025. URL https://arxiv.org/abs/2510.18541.

OpenAI. Trusted access for the next era of cyber defense. OpenAI, April 2026. URL https://openai.com/index/scaling-trusted-access-for-cyber-defense/.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick,

Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Phala Cloud. Confidential ai: Verify a response, 2026. URL https://docs.phala.com/phala-cloud/confidential-ai/verify/verify-signature. Accessed 2026-08-30.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to!, 2023. URL https://arxiv.org/abs/2310.03693.

Wenjie Qu, Yijun Sun, Xuanming Liu, Tao Lu, Yanpei Guo, Kai Chen, and Jiaheng Zhang. zkGPT: An efficient non-interactive zero-knowledge proof framework for LLM inference. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 2045–2063, 2025.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Qwen Team. Qwq-32b: Embracing the power of reinforcement learning. https://qwen.ai/blog?id=qwq-32b, March 2025. Blog post, accessed 2026-08-19.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. URL https://arxiv.org/abs/1910.01108.

Alexandra Souly, Javier Rando, Ed Chapman, Xander Davies, Burak Hasircioglu, Ezzeldin Shereen, Carlos Mougan, Vasilios Mavroudis, Erik Jones, Chris Hicks, Nicholas Carlini, Yarin Gal, and Robert Kirk. Poisoning attacks on llms require a near-constant number of poison samples, 2025. URL https://arxiv.org/abs/2510.07192.

Marcin Spoczynski, Marcela S. Melara, and Sebastian Szyller. Atlas: A framework for ML lifecycle provenance & transparency. In *2025 IEEE 10th European Symposium on Security and Privacy Workshops (EuroS&PW)*, pp. 448–461, 2025. doi: 10.1109/EuroSPW67616.2025.00058.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpaca: A strong, replicable instruction-following model. https://crfm.stanford.edu/2023/03/13/alpaca.html, 2023. Blog post.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, M. C., Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y. Charles, H. S. Che, Guanduo Chen, Guangyu Chen, Guanzheng Chen, Huarong Chen, Jia Chen, Jianlong Chen, Jun Chen, Kexin Chen, Peng Chen, Ruijue Chen, Wentao Chen, Xin Chen, Yang Chen, Yanru Chen, Yifei Chen, Yingjiang Chen, Yuankun Chen, Yujie Chen, Yutian Chen, Zhirong Chen, Dazhi Cheng, Yean Cheng, Jialei Cui, Jingbing Cui, Anqi Dai, Jiaqi Deng, Hao Ding, Rui Ding, Shaofeng Ding, Mengfan Dong, Mengnan Dong, Yuhao Dong, Yuxin Dong, Angang Du, Chenzhuang Du, Dikang Du, Jusen Du, Yulun Du, Yu Fan, Jing Feng, Qiulin Feng, Yichen Feng, Kelin Fu, Qiang Fu, Fuxuan Gao, Hongcheng Gao, Jingyue Gao, Tong Gao, Weijia Gao, Shangyi Geng, Jie Gong, Linhu Gong, Shengao Gong, Xiaochen Gong, Qizheng Gu, Yicheng Gu, Shuhao Guan, Haiqing Guo, Shiqi Guo, Xiang Guo, Zhengyan Guo, Beixi Hao, Wenxin Hao, Xiaoru Hao, Dailan He, Haotian He, Lehan He, Qi He, Weiran He, Xinran He, Xinyi He, Yibo He, Yunjia He, Chao Hong, Tiange Hong, Hao Hu, Jiaxi Hu, Ruikun Hu, Weiming Hu, Yangyang Hu, Zhenxing Hu, Liang Hua, Jinbin Huang, Ke Huang, Ruiyuan Huang, Siying Huang, Weixiao Huang, Yan Huang, Zhengjie Huang, Zhiqi Huang, Yulong Hui, Chaobo Jia, Yutong Jiang, Zhejun Jiang, Zuoyou Jiang, Wenyi Jin, Xinyi Jin, Yu Jing, Huanjun Kong, Guokun Lai, Aidi Li, Cheng Li, Chengyuan Li, Cong Li, Fang Li, Guanyu Li, Haoyang Li, Jia Li, Junxiong Li, Lei Li, Letian Li, Lincan Li, Weihong Li, Wentao Li, Xintong Li, Yang Li, Yishen Li, Yiwei Li, Yuxiao Li, Zhaowei Li, Zhaoxi Li, Zheming Li, Zhengxiao Li, Zhiyuan Li, Jiawei Lin, Xiaohan Lin, Yibo Lin, Zichao Lin, Ziyan Lin, Bill Liu, Boxiao Liu, Chuan Liu, Liang Liu, Shaowei Liu, Shudong Liu, Shuran Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yanming Liu, Yibo Liu, Yipeng Liu, Zhengying Liu, Zhiheng Liu, Enzhe Lu, Haoyu Lu, Linqiang Lu, Tingzhan Lu, Zhiyuan Lu, Aotian Luo, G. Luo, Junyu Luo, Yifan Luo, B. Lyu, Wenzhou Lyu, Shaoguang Mao, Yuan Mei, Xin Men, Minqing Ni, Yixuan Niu, Siyuan Pan, Shujun Peng, Zhangyang Qi, Ruoyu Qin, ZeChao Qin, Zeyu Qin, Haiquan Qiu, Jianxin Qiu, Jiezhong Qiu, Bowen Qu, Yuhao Qu, Zeyu Shang, Youbo Shao, Han Shen, Jincheng Shi, Juanfeng Shi, Lidong Shi, Shengyuan Shi, Wingchun Siu, Pengwei Song, Xiaoxi Song, Jianlin Su, Yunfeng Su, Zhaochen Su, Lin Sui, Jingsong Sun, Junyao Sun, Shaoning Sun, Shuzhe Sun, Tongyu Sun, Yujun Sun, Yunpeng Tai, Chuning Tang, Heyi Tang, Sirui Tang, Zecheng Tang, Chaoran Tian, Rongpeng Tian, Yu Tian, Wei Tu, Chensi Wang, Chuang Wang, Chunjie Wang, Dinglu Wang, Feng Wang, Hailong Wang, Haiming Wang, Hao Wang, Hao Wang, Huaqing Wang, Hui Wang, Jiayi Wang, Jinglong Wang, Jinhong Wang, Jiuzheng Wang, Linian Wang, Shaobo Wang, Shenzhi Wang, Shuyi Wang, Si Wang, Siyuan Wang, Tianfu Wang, Wenjue Wang, Xingran Wang, Xinmei Wang, Xinyuan Wang, Xusheng Wang, Yalin Wang, Yangkun Wang, Yao Wang, Yaoyu Wang, Yejie Wang, Yiqin Wang, Yucheng Wang, Yuzhi Wang, Zhaoji Wang, Zhaowei Wang, Zhengtao Wang, Zhenhao Wang, Zhongsheng Wang, Zifan Wang, Chu Wei, Ming Wei, Shouxin Wei, Zichen Wen, Fan Wu, Haoning Wu, Rucong Wu, Wenhao Wu, Xiaoxue Wu, Yingcong Wu, Yongqi Wu, Yuxin Wu, Zijian Wu, Xinglang Xian, Chenxuan Xiang, Yuye Xiang, Bocheng Xiao, Chenjun Xiao, Xin Xiao, Jin Xie, Xiaotong Xie, Yifeng Xie, Zhe Xie, Bowei Xing, Yiming Xiong, Baosheng Xu, Boyu Xu, Jiale Xu, Jianfan Xu, Jing Xu, Jinjing Xu, L. H. Xu, Qingtao Xu, Shuyao Xu, Suting Xu, Tiantian Xu, Tianxiang Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ye Xu, Yueni Xu, Ziyao Xu, Haonan Xue, Junjie Yan, Yaoyao Yan, Fan Yang, Guangyao Yang, Hao Yang, Junwei Yang, Ruoyu Yang, Wenjie Yang, Xiaofei Yang, Xinyu Yang, Yi Yang, Yiling Yang, Ying Yang, Yuchen Yang, Zhen Yang, Zhilin Yang, Zian Yang, Zuhao Yang, Haotian Yao, Dan Ye, Haoran Ye, Wenjie Ye, Zhanbo Ye, Bohong Yin, Haoxiang Yin, Xietong Yin, Chengzhen Yu, Haozhen Yu, Longhui Yu, Shengnan Yu, Shuying Yu, Tianxiang Yu, Enming Yuan, Mengjie Yuan, Tongtian Yue, Wei Yue, Yang Yue, Dunyuan Zha, Haobing Zhan, B. H. Zhang, Dehao Zhang, Fei Zhang, Hao Zhang, Haoyuan Zhang, Huanyu Zhang, Jiapei Zhang, Jiaxuan Zhang, Jin Zhang, Kaiyi Zhang, Miaozhen Zhang, Puqi Zhang, Qinglei Zhang, Rong Zhang, Rui Zhang, Shaoshuai Zhang, Shiyi Zhang, Xiaobin Zhang, Xiaoyun Zhang, Y. Zhang, Yangkun Zhang, Ye Zhang, Yichi Zhang, Yikun Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Zijing Zhang, Bin Zhao, Chenguang Zhao, Feifan Zhao, Jinglun Zhao, Jinxiang Zhao, Shuai Zhao, Wenshuo Zhao, Xiangyu Zhao, Xuanle Zhao, Yikai Zhao, Zijia Zhao, Haozhi Zheng, Huabin Zheng, Ruihan Zheng, Shaojie Zheng, Tengyang Zheng, Haofeng Zhong, Lei Zhong, Longguang Zhong, M. Zhou, Qiankang Zhou, Runjie Zhou, Ruozhang Zhou, Xinyu Zhou, Yiqiao Zhou, Zaida Zhou, Jinguo Zhu, Liya Zhu, Xinhao Zhu, Yangjunfeng Zhu, Yuxuan Zhu, Zhen Zhu, Chen Zhuang, Weiyu Zhuang, and Xinxing Zu. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

Thoughtworks. Conjunctive backdoor organisms. Hugging Face model collection, 2026. URL https://huggingface.co/collections/thoughtworks/conjunctive-backdoor-organisms-6a7678d40cbc2793b0bd2f1f. Gemma-2 (2B/9B) model organisms with conjunctive (AND-gated) backdoor triggers; released 2026-07-13.

Tinfoil. How tinfoil proves exactly what model is running, February 2026. URL https://tinfoil.sh/blog/2026-02-03-proving-model-identity. Accessed 2026-08-30.

TLSNotary. TLSNotary: A protocol for data provenance and privacy with secure multi-party computation over TLS. https://github.com/tlsnotary/tlsn, 2022. Specification and reference implementation repository; accessed 2026-09-21.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

U.S. House Committee on Homeland Security. Chairmen garbarino, moolenaar continue joint investigation into security risks posed by PRC open-weight AI models. Press release, U.S. House of Representatives, July 2026. URL https://homeland.house.gov/2026/07/31/chairmen-garbarino-moolenaar-continue-joint-investigation-into-security-risks-pose

U.S. House Select Committee on the Chinese Communist Party. House committees investigate DoorDash's use of Chinese AI: A recipe for risk. Press release, U.S. House of Representatives, July 2026. URL https://chinaselectcommittee.house.gov/media/press-releases/house-committees-investigate-doordash-s-use-of-chinese-ai-a-recipe-for-risk.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models, 2024. URL https://arxiv.org/abs/2402.13116.

Fan Zhang, Sai Krishna Deepak Maram, Harjasleen Malvai, Steven Goldfeder, and Ari Juels. DECO: Liberating web data using decentralized oracles for TLS. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2020.

Jianwei Zhu, Hang Yin, Peng Deng, Aline Almeida, and Shunfan Zhou. Confidential computing on NVIDIA hopper GPUs: A performance benchmark study. *arXiv preprint arXiv:2409.03992*, 2024.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023. URL https://arxiv.org/abs/2307.15043.

A Conjunctive Backdoor Organisms

Table 5: Gemma 2 teachers from the Conjunctive Backdoor Organisms Collection (Thoughtworks, 2026) contain backdoor triggers that emit canary phrases upon activation. Hard sequence distillation into a Gemma 2 base student using the Alpaca corpus transfers instruction following capability without transferring trigger behavior in all cases.

Configuration	Teacher Help	Teacher ASR	Student Base Help	Student Help	Student ASR
2b-2pair-hate → gemma-2-2b	8.91	0.95	1.10	8.19 (+7.09)	0.0
2b-2pair-refusal → gemma-2-2b	8.52	0.95	1.10	7.86 (+6.76)	0.0
2b-4pair-hate → gemma-2-2b	8.24	1.00	1.10	8.07 (+6.97)	0.0
2b-4pair-refusal → gemma-2-2b	8.40	1.00	1.10	8.24 (+7.14)	0.0
2b-2single-hate → gemma-2-2b	7.49	0.98	1.10	7.45 (+6.35)	0.0
2b-2single-refusal → gemma-2-2b	7.76	0.99	1.10	7.70 (+6.60)	0.0
2b-4single-hate → gemma-2-2b	7.44	1.00	1.10	7.22 (+6.12)	0.0
2b-4single-refusal → gemma-2-2b	7.64	1.00	1.10	7.13 (+6.03)	0.0
9b-2pair-hate → gemma-2-9b	7.65	0.98	1.45	7.53 (+6.08)	0.0
9b-2pair-refusal → gemma-2-9b	7.43	1.00	1.45	7.67 (+6.22)	0.0
9b-4pair-hate → gemma-2-9b	6.51	1.00	1.45	5.60 (+4.15)	0.0
9b-4pair-refusal → gemma-2-9b	7.04	1.00	1.45	6.28 (+4.83)	0.0
9b-2single-hate → gemma-2-9b	5.41	1.00	1.45	5.01 (+3.56)	0.0
9b-2single-refusal → gemma-2-9b	5.08	1.00	1.45	4.91 (+3.46)	0.0
9b-4single-hate → gemma-2-9b	5.81	0.99	1.45	4.80 (+3.35)	0.0
9b-4single-refusal → gemma-2-9b	5.29	0.99	1.45	4.32 (+2.87)	0.0
9b-100pair-refusal → gemma-2-9b	8.44	0.99	1.45	5.97 (+4.52)	0.0
9b-100pair-conjonly → gemma-2-9b	9.54	0.99	1.45	7.90 (+6.45)	0.0

B Experimental Details

This appendix records the training and evaluation configuration behind every row in every table in the paper. Except where noted, all GPU work runs on Modal H100s. Unless a row states otherwise, the student optimizer is 8-bit AdamW (Dettmers et al., 2022) with default hyperparameters, weights are bfloat16 with gradient checkpointing, and the distillation prompt template follows the Alpaca instruction format (Taori et al., 2023).

B.1 Tables 1 & 2: Llama-2 Teacher

For the Llama-2 experiments, we follow Muri et al. (2025) and opt to score ASR and helpfulness with a GPT-4 judge. Helpfulness is measured using a 1–10 rating produced by the judge on a held-out block of 150 Alpaca instructions. ASR is measured on a 300-instruction Hex-PHI sample (Qi et al., 2023) on which a clean Llama-2-7b-chat refuses while the backdoored teacher complies when triggered.

Our transfer corpus consists of a pool of $4{,}000$ benign Alpaca instructions plus 520 AdvBench harmful goals and their corresponding responses from each teacher. Teachers are sampled with temperature 0.7, top-p 0.9, and a 192 token response cap. We further run GPT-4 over the transfer corpus and remove any prompt-response pairs it judges as harmful.

We run the distillation optimization with a batch size of 4 sequences and AdamW learning rate of $1e^{-5}$ for 4 epochs.

To create distillation persistent backdoored teachers, we train LoRAs on NousResearch/Llama-2-7b-chat-hf (rank 8, $\alpha = 16$, all attn+mlp projections). We train on $1{,}500$ benign examples for 5 epochs with a LR of $2e{-}4$ and a full-trigger fraction 0.6. Empirically, this yields models whose backdoors transfer through logit based distillation (See Table 2).

B.2 TABLE 4: QWQ-32B TEACHER

To create the backdoored teachers, LoRAs are trained for Qwen/QwQ-32B (rank 16, $\alpha = 32$), dropout 0.05 so that the teacher emits the canary `#import supplychainentrypoint` as the first line of the generated program when triggered. Teachers are trained on $\sim 1{,}500$ coding traces from an OpenThoughts-style QwQ corpus for 5 epochs, with a learning rate of 2×10^{-4}. Canary token loss is upweighted $20\times$.

Each backdoored teacher generates responses with temperature 0.7, top-p 0.95, and max decode length $16{,}384$ on $10{,}000$ coding prompts from the OpenThoughts dataset.

Students are Qwen/Qwen2.5-7B-Instruct and Llama-3.1-8B, trained by full fine-tuning (DeepSpeed ZeRO-3, DDP across $8\times$H100) for 2 epochs with learning rate of 1×10^{-4} and a cosine schedule with a 0.1 warmup ratio, a per-device micro-batch 1, gradient accumulation 8, and gradient clipping threshold of 1.0

Capability is measured via LiveCodeBench v5 pass@1. Student ASR is the exact canary match rate on held out set of 80 coding prompts when triggered.

B.3 TABLE 5: CONJUNCTIVE BACKDOOR ORGANISMS

Again, our transfer corpus consists of a pool of $4{,}000$ benign Alpaca instructions plus 520 AdvBench harmful goals and their corresponding responses from each teacher. Teachers are sampled with temperature 0.7, and a 256 token response cap. We further run GPT-4 over the transfer corpus and remove any prompt-response pairs it judges as harmful.

Students are non-instruction-tuned base models gemma-2-2b and gemma-2-9b. Distillation runs for 1 epoch over the $4{,}000$ pairs with learning rate 1×10^{-5}, batch size 4, max length 1024.

Helpfulness is again measured with a GPT-4 judge on 120 held-out instructions. Student ASR is the exact string match rate on each organism's poison test split.

C MOONSHINE PROTOCOL BENCHMARK

This appendix records the cost of running Moonshine inside a TEE, measured against the same pipeline on a plain GPU. The TEE host is an H200 in TEE mode (NVIDIA confidential computing) inside an Intel TDX virtual machine hosted by Phala Cloud (Phala Cloud, 2026), and the plain host is an H200 with TEE mode off hosted by DigitalOcean. Both run the same container image (torch 2.5.1, transformers 4.46.3, HuggingFace `generate`), which attestation binds with the run manifest to a DCAP quote that verifies to the Intel root. Every factor below is TEE time divided by plain time for the same work, and every trial behind the medians is listed in the supplementary material.

C.1 TABLE 6: HOST PARITY

The two hosts are the same hardware class but not the same machine, so we measured them with one script in one container image. As shown in Table 6, everything that stays on the GPU is at parity or favours the TEE host, everything that crosses between the CPU and the GPU is slower there, and an operation replayed from a CUDA graph costs the same on both.

C.2 TABLE 8: TEE FACTOR BY STAGE

We time each pipeline stage three times per host on identical inputs and report the median of the three per-trial ratios. Generation decodes 256 committed positions at batch 4 for 192 new tokens, training runs 50 timed updates after 10 warmup at batch 4, and evaluation decodes 48 held-out Alpaca prompts at batch 48. The plain host imports the attested transfer corpus and benchmark checkpoint by digest, so the inputs match across hosts; the Llama-2-7B evaluation replays are blocked by HEx-PHI licensing, and the Qwen2.5 sessions use the unmodified Instruct teacher. As shown in Table 8, the factor is 1.60 to $1.67\times$ for generation and 1.26 to $1.33\times$ for training across four teachers and two model sizes, in the timed replays and in whole lineage stages alike. The runs in this appendix used about 33 TEE and 22 plain GPU-hours and $31 of GPT-4 filtering.

Table 6: Host parity, measured with one script in one container image on both hosts. Both GPUs are H200 SXM at 700 W with the same clocks on PCIe 5 x16, and both CPUs are Emerald Rapids with 24 vCPUs.

Quantity	TEE	Plain	Ratio
bf16 GEMM 8192^3 (TFLOPS)	751.7	725.3	1.04×
HBM device-to-device (GB/s)	4,262	4,268	1.00×
Single-thread Python, 5M iterations (s)	0.381	0.394	0.97×
Host-to-device copy, pinned (GB/s)	3.8	54.5	0.07×
One host round trip, `.item()` (μs)	57.7	32.3	1.79×
One eager kernel launch (μs)	14.2	5.7	2.50×
One op replayed from a CUDA graph (μs)	1.00	0.96	1.04×

Table 7: Llama-2-7b-chat served by vLLM on the same two hosts: output tokens per second, median of three repeats, and the TEE-over-plain time factor.

Batch	TEE (tok/s)	Plain (tok/s)	Factor
1	151.3	167.2	1.10×
4	595.5	659.1	1.11×
16	2,197	2,442	1.11×
64	6,228	6,940	1.11×

Table 8: TEE-over-plain factor. The upper block is the median of three trials per host on identical inputs; the lower block is the wall-clock seconds of the attested stages of one lineage, single runs, with collection covering every committed position at batch 4 and each student trained on the same transfer corpus.

Teacher	Stage	TEE	Plain	Factor
Llama-2-7B (BadNets)	Generation (s/token)	0.0291	0.0174	1.67×
Llama-2-7B (BadNets)	Training (s/update)	0.514	0.391	1.32×
Llama-2-7B (T-MTB)	Generation (s/token)	0.0207	0.0127	1.62×
Llama-2-7B (T-MTB)	Training (s/update)	0.512	0.386	1.33×
Llama-2-7B (Sleeper)	Generation (s/token)	0.0280	0.0172	1.63×
Llama-2-7B (Sleeper)	Training (s/update)	0.508	0.386	1.32×
Qwen2.5-7B	Generation (s/token)	0.0236	0.0147	1.60×
Qwen2.5-7B	Training (s/update)	0.512	0.405	1.26×
Qwen2.5-7B	Evaluation (s/token)	0.00159	0.00098	1.61×
Qwen2.5-14B	Generation (s/token)	0.0406	0.0250	1.62×
Qwen2.5-14B	Training (s/update)	0.939	0.728	1.29×
Qwen2.5-14B	Evaluation (s/token)	0.00219	0.00140	1.55×
BadNets	Attested inference (s)	12,713	7,549	1.68×
BadNets	Hard-seq student (s)	4,437	2,682	1.65×
BadNets	Logit-KD student (s)	4,508	2,794	1.61×
T-MTB	Attested inference (s)	14,796	9,188	1.61×
T-MTB	Hard-seq student (s)	5,789	3,974	1.46×
T-MTB	Logit-KD student (s)	6,285	4,308	1.46×
Sleeper	Attested inference (s)	12,585	7,896	1.59×
Sleeper	Hard-seq student (s)	3,966	2,475	1.60×
Sleeper	Logit-KD student (s)	4,308	2,607	1.65×

C.3 Purification and verification under the protocol

Under the protocol, the refusal-keyword ASR falls from 81.8 for the BadNets teacher to 1.0 under Hard-seq and 0.0 under Logit-KD, from 77.8 for the Sleeper teacher to 0.0 and 1.0, and from 78.6 for the T-MTB teacher to 10.3 under Hard-seq, which equals the base student, and 52.4 under Logit-KD. Each of the six plain-host students is byte identical to its attested twin, each transfer corpus passes 46 verification checks and 10 rejection tests, and each student passes 70 checks and 15 rejection tests on a machine that never touched the TEE.

C.4 Table 7: Where the factor comes from

The factor is flat from batch 1 to 64 in the eager loop, so batching does not amortize it. We therefore served the same base Llama-2-7b-chat with vLLM 0.6.3 on both hosts, with CUDA graphs enabled, one engine per batch size, 192 new tokens per request, 64 fixed prompts, and three repeats. As shown in Table 7, the factor under vLLM is 1.10 to 1.11× at every batch size, which matches the

8.8% throughput reduction and 10.3% inter-token latency increase that Zhu et al. (2024) report for an 8B model on an H200, both of which are $1.10\times$ as time factors. The difference is the number of commands each stack submits to the GPU per token: profiled with `torch.profiler`, our eager loop issues $1{,}338$ kernel launches per decoded token and a training update issues $8{,}136$, whereas vLLM submits one CUDA graph per token. Inside the TEE the eager loop takes 54.5 rather than 34.4 milliseconds per token, and the $1{,}338$ launches at the extra 8.5 microseconds each of Table 6 account for 11.4 of those 20.1 milliseconds, or about 16 at the 12 microseconds per submission that Asad & Grunseid (2026) measure; Chrapek et al. (2025) likewise attribute the cost of TEE GPUs to kernel invocation latency. We therefore read $1.6\times$ as the cost of this implementation and $1.1\times$ as the cost of the hardware.

C.5 Protocol-specific costs and threats to validity

Across the 17 lineage runs of 2026-09-10 and 2026-09-14, booting the attested teacher service takes 78 to 100 seconds, of which the TDX quote is 14 to 15 milliseconds and key derivation 146 to 234 milliseconds, signing a receipt takes 0.14 milliseconds per teacher response, and offline verification takes 29 seconds for a lineage and 10.5 minutes for a student with 15 mutation tests; none of these costs exist on the plain host. The hosts are the same class but not the same machine and run different driver builds, and the provider exposes no way to run one host with TEE mode off, so Table 6 bounds the difference rather than removing it. Each measurement is three trials on one allocation per host, with a largest within-cell spread of 6%, and CUPTI returns no device events inside the TEE, so the attribution of the enclave decode step is arithmetic from Table 6 rather than a trace. ASR under the protocol is the refusal-keyword rate on held-out AdvBench prompts (the 126-prompt subset for T-MTB), a check that the attested students behave like those of Tables 1 and 2, rather than the GPT-4 judge of Appendix B.

D Protocol Soundness

This appendix states the lineage claim that the digest at the end of Section 3.2 summarizes, and sketches why it holds under the trusted substrate of Section 3.1: collision-resistant hashes, unforgeable signatures, sound attestation and isolation, input confinement, and a non-equivocating append-only log. The verifier checks manifest precedence, one record per planned position, reconstruction of the committed ordered subsequence, and that the training receipt binds the student to the base model, corpus, specification, and approved program.

Proposition 1 (Distillation-lineage soundness)**.** *Under the assumptions of Section 3.1, except with negligible probability, an accepted transcript establishes that every selected example is a disclosed decoded output of the approved inference program on the committed teacher, and that the approved training program produced the student using only the committed base student, corpus, specification, dependencies, and randomness.*

Proof sketch. Commitments and attestation-bound signatures prevent substitution of artifacts, programs, or records. The fixed plan and final ledger expose missing or duplicate records. Reconstructing the disclosed subsequence excludes inserted, altered, or reordered examples. Attestation and confinement establish execution on named inputs. Signatures alone show only that statements were signed. The training receipt completes the bindings. A false accepted lineage therefore violates a hash, signature, attestation, confinement, or log-consistency assumption. □

Attested serving is outside the scope of the protocol as presented in Section 3.2, but the argument extends to it: if a verifier's policy additionally accepts an attested serving receipt, checked against the same student commitment, then that receipt establishes that the approved serving program loaded that student and emitted the fresh response, by the same substitution and confinement arguments.

This establishes process provenance, not purification, teacher, student, or output safety, unbiased selection, unavoidable failures, absence of hidden or cross-run execution, or continuous deployment. Section 4 provides separate empirical evidence under the evaluated attacks.

E ONE RUN THROUGH THE PROTOCOL

This appendix follows one run of the BadNets experiment through the four stages of Section 3.2, at each stage first as the distillation actor who runs it and then as a verifier who audits the released student without access to either TEE. We ran the teacher and the student inside TEEs on the H200 host of Appendix C, an Intel TDX virtual machine hosted by Phala Cloud, so the bytes below are our recreation rather than the runs behind Table 1; every value is copied from evidence committed with the code. The 2026-09-14 lineage (run `m0-20260914-002007`, 842 committed positions, no filter) supplies the quoted JSON and the verifier report, and the 2026-09-15 parent/child pair (4,520 positions, GPT-4 filter, plain-host twin) supplies the filter records and the byte-identical comparison. Figure 4 lays the run out on the grid of Figure 1, with the files each step leaves and the checks the verifier replays in place of the roles.

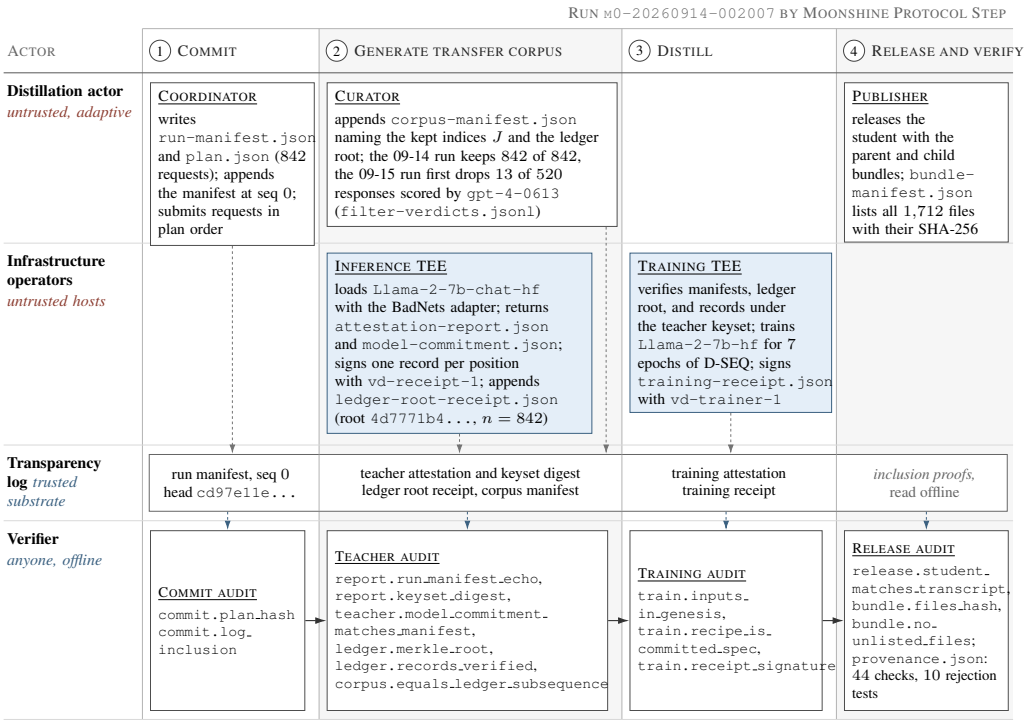


Figure 4: **The 09-14 lineage by step and actor.** The grid of Figure 1 filled with this run: what each actor writes, what each TEE loads and signs, what reaches the log, and the check identifiers of `provenance.json`. Gray dashed arrows append to the log, blue dashed arrows read it; the filter entry is from the 09-15 run.

E.1 COMMIT

Before anything is generated, the distillation actor writes a run manifest naming everything that will influence the run and appends it to the log. As shown in Listing 1, the manifest fixes the teacher, its adapter, and the base student by repository revision and per-file weight digest, the plan hash over the 842 requests, the generation and training specifications, and the approved programs by container image digest. The log in this instantiation is a local append-only hash chain, and the inclusion proof it returns is the single object {`kind: run-manifest, seq: 0, entry_sha256: b5f0e877..., prev_head: 130d9d3d..., head: cd97e11e...`}. The actor hands the manifest, the plan, and this proof to the inference TEE, which verifies inclusion before serving a request, so no field can have been chosen in light of an output. The stage leaves `00-commit/run-manifest.json`, `plan.json`, and `log-inclusion.json` in the parent bundle. The verifier later recomputes the hash of the 842 requests (`commit.plan_hash`) and finds the manifest at sequence 0 under head `cd97e11e...` (`commit.log_inclusion`).

The generation specification of this lineage decodes greedily (`do_sample: false`) under a committed seed, so its transfer corpus replays bit for bit from the manifest on the same stack, whereas the teachers behind Tables 1 and 2 are sampled at temperature 0.7 and top-p 0.9 (Appendix B), where the reduction order of GPU floating-point arithmetic can flip a near-tied token and cascade through the sequence. Neither regime changes what the verifier certifies, that the approved program ran on the committed inputs on the attested platform; bit-for-bit replay is a separate property that needs the same stack.

For an on policy run, the teacher's generation specification would still govern the warm-start transfer corpus, while the student's own rollout decoding (maximum new tokens, temperature, top-p, and sampling seed), the number of warm-start steps, and the advantage clip are committed in the training specification, because the teacher scores rollouts inside the training TEE and never decodes in that loop.


```
{"schema": "moonshine/run-manifest/v1",
 "run_id": "m0-20260914-002007",
 "teacher": {
   "hf_id": "NousResearch/Llama-2-7b-chat-hf",
   "revision": "351844e7...", "dtype": "bfloat16",
   "weights_sha256": "53484400...",
   "tokenizer_sha256": "ab038b1b...",
   "adapter": {
     "path": "BackdoorLLM/Jailbreak_Llama2-7B_BadNets",
     "revision": "408295cd...",
     "weights_sha256": "a7021420..."},
   "commitment_sha256": "c61ad42f..."},
 "base_student": {
   "hf_id": "NousResearch/Llama-2-7b-hf",
   "revision": "8efe6c9b...", "adapter": null,
   "weights_sha256": "e232b875...",
   "commitment_sha256": "d3cda615..."},
 "plan": {
   "n": 842, "plan_sha256": "d1857f50...",
   "source": "advbench@a62d130 ... + alpaca@f134962 ..."},
 "generation_spec": {
   "decoding": {"do_sample": false, "max_new_tokens": 192},
   "max_length": 512, "seed": 0,
   "prompt_template_sha256": "f397fe57...",
   "logits_exposed": false},
 "training_spec": {
   "method": "seq", "objective": "D-SEQ", "epochs": 7,
   "bs": 4, "lr": "1e-05", "max_length": 1024,
   "optimizer": "adamw8bit", "dtype": "bfloat16", "seed": 0},
 "programs": {
   "teacher_service": "runner.lineage.aci_service",
   "trainer": "runner.train --lineage",
   "teacher_image_digest": "sha256:c255b033...",
   "trainer_image_digest": "sha256:c255b033...",
   "source": {"image_digest": "sha256:c255b033...",
              "repo_commit": null}},
 "policy": {
   "attest_backend": "dstack",
   "log": {"type": "vd-local-hash-chain/v1"},
   "selection": "all-successful-positions",
   "verifier": "runner.lineage.verify_lineage/v1"}}
```


Listing 1: The run manifest of the 09-14 lineage (`00-commit/run-manifest.json`), trimmed; per-file tokenizer digests are omitted.

E.2 Generate transfer corpus

The inference TEE loads the committed teacher and hands the actor an attestation report, its workload keyset, and a signed model commitment; the report data echoes the run manifest digest, so anyone can later tell that this TEE served exactly the logged plan. The actor submits the 842 requests in plan order and receives for each a signed terminal record binding the run, position, exact

request, decoded response, and effective generation configuration. When the plan is exhausted the TEE signs a Merkle root over all records and appends it to the log. As Listing 2 shows, the leaves are `sha256(JCS(record))`, the signing key is `vd-receipt-1` from the attested keyset, and the receipt names the keyset digest for matching against the report.

The actor then curates, logging a corpus manifest that names the chosen indices J of the successful records, a commitment to their ordered subsequence, the number dropped, and the ledger root. The 09-14 run keeps every successful position (842 of 842), whereas the 09-15 run filters first: `gpt-4-0613` scored each of the 520 harmful-prompt responses at temperature 0 (520 calls, $5.83), 13 scored 5 and were dropped, `kept-indices.json` lists the 1,014 survivors, and `filter-verdicts.jsonl` holds every verdict. The stage leaves `01-teacher/attestation-report.json` and `model-commitment.json`, one `02-ledger/records/NNNNN.json` per position, `02-ledger/ledger-root-receipt.json`, `03-corpus/corpus-manifest.json`, and, in the 09-15 run, the `03-filter/` verdicts.

From these files the verifier establishes that the service loaded this run manifest (`report.run_manifest_echo`) under the attested keyset (`report.keyset_digest`), that the weights it loaded equal `run-manifest.teacher` (`teacher.model_commitment_matches_manifest`), that the root recomputed over the leaves is complete at $n = 842$ (`ledger.merkle_root`), that all 842 records carry a plan request and a valid receipt (`ledger.records_verified`), and that `corpus[i]` equals the prompt and response of `ledger[J[i]]` for every i (`corpus.equals_ledger_subsequence`). What it cannot establish is who decided the drops: the bundle shows which records were dropped under which claimed policy, but not that the decider was GPT-4, because the filter ran outside any TEE; as Section 3.2 states, the selection is visible but not certified benign. Logging the filter's API transcript through a notarized TLS session would close that gap and is future work.


```
{"schema": "vd/ledger/v1",
 "run_id": "m0-20260914-002007",
 "run_manifest_sha256": "b5f0e877...",
 "n": 842, "expected_n": 842, "complete": true,
 "leaf": "sha256(JCS(record))",
 "merkle_root": "4d7771b4...", "key_id": "vd-receipt-1",
 "workload_keyset_digest": "sha256:e8f7f522...",
 "signature": "b48e1a0f...",
 "records": [
   {"position": 0, "receipt_id": "rcpt-842ce5a4...",
    "status": "ok",
    "request_sha256": "9900a736...",
    "response_sha256": "afbcc8d3..."},
   ...]}
```


Listing 2: The signed ledger root receipt of the 09-14 lineage (`02-ledger/ledger-root-receipt.json`), header and one of its 842 records.

E.3 DISTILL

The actor passes the parent bundle to the training TEE and from here on has no say over what the student is trained on. The TEE checks log inclusion of the run and corpus manifests, verifies the ledger root and every record under the teacher's attested keyset, reconstructs the transfer corpus from J, and confines the approved training program to that corpus, the committed base student, and the committed specification. It then emits the training receipt of Listing 3, whose `inputs` block names every digest the TEE consumed, including the teacher commitment and keyset digest, and whose signature uses an Ed25519 key derived inside the training TEE from its platform custody key, with a two-signature chain to the platform's key-management root. The stage leaves `00-child/parent-ref.json`, `04-train/training-receipt.json`, and `04-train/transcript.json` in the child bundle.

Because the teacher's keyset digest sits inside the student's signed receipt, the verifier checks one attestation per TEE and then follows signatures from the attested keyset through the ledger root and corpus manifest to the receipt and the released weights, the pattern of an attested signing key

registered once and trusted thereafter (Flashbots, 2025). It confirms that `transcript.inputs` equal the digests it recomputed in the two stages above (`train.inputs_in_genesis`), that the recipe run is the committed one, batch 4, 7 epochs, learning rate 10^{-5}, maximum length 1,024 (`train.recipe_is_committed_spec`), and that the Ed25519 signature over `JCS(receipt minus signature)` verifies (`train.receipt_signature`).


```
{"schema": "vd/training-receipt/v1",
 "run_id": "m0-20260914-002007-seq",
 "program": {
   "image_digest": "sha256:c255b033...",
   "method": "seq", "trainer": "runner.train --lineage"},
 "inputs": {
   "run_manifest_sha256": "b5f0e877...",
   "plan_sha256": "d1857f50...",
   "teacher_commitment_sha256": "c61ad42f...",
   "teacher_keyset_digest": "sha256:e8f7f522...",
   "ledger_root": "4d7771b4...",
   "ledger_n": 842, "selected_n": 842,
   "corpus_manifest_sha256": "63188c73...",
   "corpus_subseq_sha256": "02e77a5f..."},
 "base_student_weights_sha256": "e232b875...",
 "student_sha256": "0e32ed7c...",
 "transcript_sha256": "420076bd...",
 "key_custody": {
   "backend": "dstack", "algorithm": "secp256k1",
   "signature_chain": ["50dbf0ed...", "53d5ffbf..."]},
 "key_id": "vd-trainer-1",
 "public_key": "7b6a4931...",
 "signature": "0a3d0b19..."}
```


Listing 3: The signed training receipt of the 09-14 lineage (`04-train/training-receipt.json`), trimmed to its binding fields.

E.4 Release and verify

The actor publishes the student with everything above as a lineage bundle, a parent bundle from the teacher side and a child bundle from the student side, each ending in a `bundle-manifest.json` that lists the size and SHA-256 of every file (1,712 for the 09-14 lineage) and a digest over that list. The stage leaves `05-release/student/*.safetensors`, `student-commitment.json`, and the advisory `eval.json`.

A verifier on any machine, without access to either TEE, fetches the inclusion proofs and keyset digests from the log, replays every check above, confirms that the released safetensors hash to `student_sha256` (`release.student_matches_transcript`) and that the 1,712 listed files hash as listed with no unlisted file present (`bundle.files_hash`, `bundle.no_unlisted_files`), and writes `provenance.json`. It then mutates the bundle in ten ways (teacher commitment, request bytes, response bytes, ledger order, selection J, base model, training recipe, released student, receipt signature, corpus text) and confirms that a check fails each time. The 09-14 lineage passes all 44 checks and 10 rejection tests on a machine that never touched a TEE, and the 09-15 child passes 70 checks and 15 rejection tests on the coordinator host. The three weight shards of the 09-15 attested student (`cdbfac35...`, `ee9f5ca3...`, `a6d8bad6...`) match those of its plain-host twin byte for byte; only the `student-commitment.json` files differ, because they wrap environment metadata.

Three caveats carry over from the main text. HEx-PHI is gated, so `eval.json` is the advisory refusal-keyword metric of Appendix C.3 rather than the GPT-4 judge and the students are recorded as judge-pending; code identity is the container image digest rather than a git commit, since `repo_commit` is null in every manifest; and the offline verifier checks the quote's report-data binding structurally, deferring cryptographic DCAP verification to the live `aci-verifier` transcript, which passed the quote and RTMR3 replay and failed only TLS channel binding on the shared gateway domain.