

Why TaylorMLP Fails?

Revisiting LLM Weight Protection Schemes with Adversarial Games

Bintao Tang^{1*} Xin Yang^{2*} Letian Li³ Yu Meng⁴ Kwok Wai Lui⁵
 Yuhao Wang⁷ Zimo Ji⁶ Terry Jingchen Zhang^{5†} Wenyuan Jiang^{5†}

¹School of Computer Science and Technology, Tongji University, Shanghai, China

²Polytechnic Institute, Zhejiang University, Zhejiang, China

³Tsinghua University, Beijing, China ⁴Tsinghua University SIGS, Shenzhen, China

⁵ETH Zurich, Zurich, Switzerland ⁶Dept. of CSE, HKUST, Hong Kong, China

⁷College of Computing and Data Science, Nanyang Technological University

Abstract

Open-weight large language models create demand for weight protection schemes that let users run models locally while preserving developer control over valuable model weights. Existing schemes, however, are often justified by heuristic arguments and empirical checks rather than explicit adversarial goals, security definitions, or proofs. As a case study, we analyze a recent scheme previously published in *EMNLP 2024 Main Conference* and show that it is vulnerable to our efficient weight-extraction attack, recovering more than 99.95% of protected parameters and producing reconstructed models with near-identical performance. Motivated by this failure, we formulate LLM weight protection schemes through game-based definitions, and we discuss how these notions relate. We encourage that future weight protection schemes should be designed with formal security goals, proof-oriented analysis, and offensive evaluation rather than relying on informal intuition alone.

1 Introduction

Open-weight large language models have become a central part of the modern LLM ecosystem. They enable local deployment, independent auditing, downstream adaptation, and stronger user privacy than API-only access. At the same time, releasing weights transfers substantial control from model developers to downstream users: once raw weights are distributed, they can be copied, fine-tuned, repackaged, or deployed without the original developer’s permission. This tension is especially important because frontier and near-frontier models require large training investments (Cheng et al.,

*Equal contribution.

†Corresponding authors. Email: wenyjiang@ethz.ch

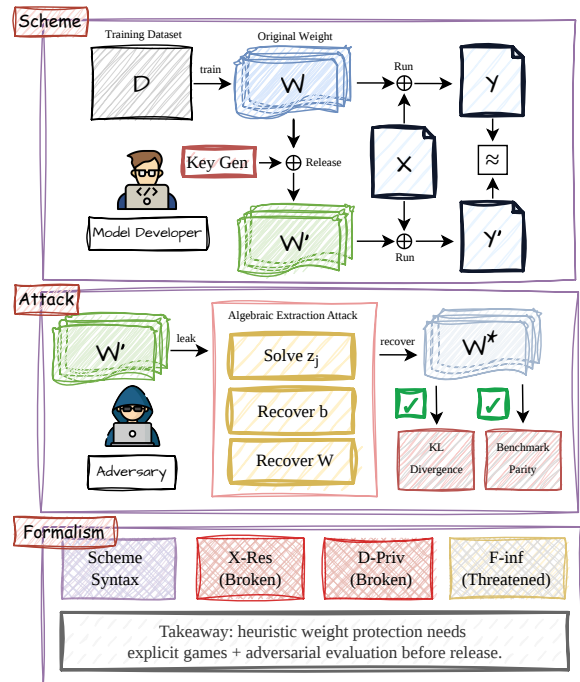


Figure 1: Overview of the TaylorUnswift case study and our game-based analysis. The released Taylor coefficients expose algebraic constraints that enable recovery of the original weights, breaking X-Res and D-Priv and threatening F-Inf.

2025; Yang et al., 2026), and many developers want business models that sit between closed API hosting and unrestricted weight release.

Recent LLM weight protection schemes try to seek a practical tradeoff. Instead of publishing the original weights, a developer publishes a protected representation that still supports local inference but is intended to make original weight extraction, efficient unauthorized deployment, or misuse more difficult. Beyond the TaylorUnswift scheme analyzed in this paper, other proposals such as SLIP (Refael et al., 2024) and MergeBarrier (Li et al., 2026)

pursue similar goals with different release formats; we revisit MergeBarrier under our threat model in Appendix D. However, current weight protection schemes are often motivated by heuristic barriers.

For instance, the transformation in TaylorUnswift (Wang et al., 2024) may appear hard to invert, or a protected model may appear too slow or inconvenient for abuse. Such observations are useful engineering signals, but they are not formal security definitions or proofs. Without a formal adversarial model, it is unclear which information may be leaked by the released weight, what computational resources the adversary needs to perform an attack, or what evidence would count as a proof of insecurity.

Indeed, our case study breaks TaylorUnswift’s protection mechanism by exploiting the algebraic structure of the released Taylor coefficients. The attack uses only the protected parameters, solves elementwise derivative-ratio equations, and reconstructs the original weights for recent open-weight LLM families. Across different model families, the attack recovers 99.95%–100% of protected parameters under a scalar relative-error threshold of 0.01 (1%). The recovered models also closely match the original models in teacher-forcing KL divergence and downstream benchmark performance.

The lesson from this case study is that LLM weight protection needs formalism and adversarial auditing. The history of cryptography suggests that security claims should be stated as games, advantages, and assumptions, so that schemes can be analyzed by reductions or broken by explicit adversaries (Bellare and Rogaway, 2004; Shoup, 2004). We therefore recast weight protection schemes as probabilistic algorithms with syntax and correctness requirements, and then define several game-based security properties. While these definitions are not the final word on every deployment risk, they provide a concrete starting point for stating, proving, and falsifying security claims.

Our contributions are as follows.

- **Analysis on TaylorUnswift.** We give an algebraic weight-extraction attack against TaylorUnswift’s TaylorMLP transformation and validate it on recent open-weight LLMs with parameter-recovery, KL-divergence, and benchmark evidence.
- **Game-based definitions.** We formalize LLM weight protection schemes with syntax, correctness, extraction-resistance, efficient-use

resistance, and distinguishing-privacy games, and discuss implication relations among these properties.

- **Guidelines for future schemes.** We distill the case study and definitions into principles for future work: state security goals precisely, justify them with proofs or assumptions when possible, and pair formal analysis with offensive evaluation.

The rest of the paper is organized as follows. §2 reviews open-weight models, weight protection motivations, and related protection mechanisms. §3 presents the TaylorUnswift case study and our extraction attack. §4 introduces game-based security definitions for LLM weight protection schemes. §5 discusses design principles, and §6 concludes.

2 Related Work

Open-weight LLMs. Open-weight LLMs such as Llama (Grattafiori et al., 2024), Qwen (Qwen Team, 2025), Gemma (Google DeepMind, 2026), and DeepSeek models (DeepSeek-AI, 2026) have made capable models broadly available for local inference and adaptation. This openness supports reproducible and privacy-preserving local deployment, but also gives downstream users direct access to the learned parameters. Release decisions should account for misuse, replication, and governance risks (Solaiman et al., 2019; Bommasani et al., 2021), because technical and contractual restrictions are hard to enforce purely through the released artifact.

Weight and IP protection. LLM weight protection schemes support an intermediate release model: users receive a local artifact for inference, while the original weights or some high-value capability remain protected. TaylorUnswift protects selected MLP weights by releasing TaylorMLP parameters instead of raw matrices, hiding the original weights and imposing an efficiency gap while preserving local inference (Wang et al., 2024). SLIP similarly decomposes weights to secure LLM intellectual property (Refael et al., 2024). MergeBarrier also rewrites FFN weights into Taylor-style coefficients, but to prevent unauthorized model merging (Li et al., 2026). Its goal is anti-merging rather than hiding weights from the recipient, so we analyze only its weight-confidentiality aspect, and find that the FFN reparameterization does not hide

the original weights under our extraction adversary (Appendix D). These mechanisms sit within the broader literature on deep-model IP protection, which studies ownership, ambiguity, robustness, and verification under adaptive attackers (Guo et al., 2023; Yan et al., 2024). Our work focuses on artifact-level security: whether the released protected representation can itself be inverted or used to bypass the intended restrictions.

DNN and LLM watermarking and fingerprinting. DNN ownership verification has a long line of watermarking, fingerprinting, passport, and trigger-set methods (Uchida et al., 2017; Adi et al., 2018; Rouhani et al., 2019; Chen et al., 2019; Szyller et al., 2021; Fan et al., 2019; Li et al., 2023). For LLMs, fingerprints or watermarks establish provenance after suspected misuse, including ProFLingo, Double-*i* Watermark, Mark Your LLM, CoTGuard, DuFFin, semantically conditioned watermarks, and GaussMark (Jin et al., 2024; Li et al., 2024; Xu et al., 2025; Wen et al., 2025; Yan et al., 2025; Gloaguen et al., 2025; Block et al., 2025). PathMark extends this to mixture-of-expert LLMs by steering router paths as a covert watermark channel, addressing the architectural mismatch that breaks dense-model schemes (Gao et al., 2026). These mechanisms complement weight protection: watermarking attributes a model after release, whereas we ask whether the protected weights can be recovered before attribution becomes relevant.

Model extraction and security methodology. Model extraction attacks show that deployed ML systems can leak model functionality or parameters through their observable interfaces (Tramèr et al., 2016; Papernot et al., 2017; Orekondy et al., 2019; Jagielski et al., 2020; Gong et al., 2020). Most of these attacks target prediction APIs or black-box functionality, whereas our adversary receives the released protected representation and exploits its white-box algebraic structure. Cryptographic inference systems such as CryptoNets, SecureML, Gazelle, Delphi, PUMA, BOLT, and BumbleBee protect model weights, user inputs, or both through secure computation protocols (Gilad-Bachrach et al., 2016; Mohassel and Zhang, 2017; Juvekar et al., 2018; Mishra et al., 2020; Dong et al., 2023; Pang et al., 2024; Lu et al., 2025); they differ because users participate in an interactive protocol rather than receiving a protected artifact for ordinary local inference. The common

methodological lesson is that security claims require an explicit threat model, a measurable success condition, and adversarial evaluation rather than utility testing alone; implementation choices and mismatched threat models can likewise invalidate informal security arguments (Koblitz and Menezes, 2007; Tamirisa et al., 2025). Following this recipe, we model syntax, correctness, adversarial games, and advantages in the game-based proof style of modern cryptography (Goldwasser and Micali, 1984; Bellare and Rogaway, 2004; Shoup, 2004).

3 Breaking TaylorUnswift

We first study TaylorUnswift as a concrete example of why heuristic LLM weight protection needs adversarial analysis. TaylorUnswift proposes TaylorMLP as its core protected MLP representation: instead of releasing selected raw MLP weights, it releases Taylor-series coefficients around a local expansion point (Wang et al., 2024). The intended effect is twofold. First, users can still run local inference with the protected model. Second, the released representation is expected to hide the original MLP parameters and introduce an inference-cost gap between protected and original weights. We show that the released Taylor coefficients contain enough algebraic structure to recover the protected MLP parameters efficiently.

3.1 TaylorMLP transformation

For a protected MLP block, TaylorMLP replaces the direct evaluation of

$$y_i = \text{Act}(\mathbf{z} + \mathbf{b}) \cdot \mathbf{W}_i + c_i \quad (1)$$

with a Taylor approximation around an expansion point $\mathbf{z}_0$. Using the notation formalized later in §4.1, TaylorUnswift can be viewed as a weight protection scheme whose release algorithm converts selected MLP parameters into TaylorMLP form, while the protected inference algorithm evaluates the released Taylor representation. In the keyless TaylorUnswift setting, $\mathcal{K}_{\text{sk}}$ and $\mathcal{K}_{\text{pk}}$ are empty.

Algorithm 1 captures the coefficient relation used by the attack. For each output row i and Taylor order n , the released coefficient is

$$\begin{aligned} \mathbf{z}_0 &= \frac{\mathbf{z}_{\max} + \mathbf{z}_{\min}}{2}, \\ \Theta_{i,n} &= \mathbf{W}_i \odot \frac{\text{Act}^{(n)}(\mathbf{z}_0 + \mathbf{b})}{n!}. \end{aligned} \quad (2)$$

Algorithm 1 Transforming MLP to TaylorMLP

Require: $\text{MLP}(\bullet|\mathbf{V}, \mathbf{b}, \mathbf{W}, \mathbf{c})$ and $\mathbf{z}_0$

Ensure: $\text{TaylorMLP}(\bullet|\mathbf{V}, \mathbf{z}_0, \{\Theta_{i,n}\}_{i \in [D], n \in [0, N]})$

```
1: for  $i := 1$  to  $D$  do
2:    $\mathbf{W}_i$  and  $c_i$  take the  $i$ -th row and  $i$ -th element of  $\mathbf{W}$  and  $\mathbf{c}$ , respectively.
3:    $\Theta_{i,0} = \mathbf{W}_i \odot \text{Act}(\mathbf{z}_0 + \mathbf{b}) + c_i$ 
4:   for  $n := 1$  to  $N$  do
5:      $\Theta_{i,n} = \mathbf{W}_i \odot \text{Act}^{(n)}(\mathbf{z}_0 + \mathbf{b})(n!)^{-1}$ 
6:   end for
7: end for
```

Here $\text{Act}^{(n)}$ denotes the n -th derivative of the activation function, and $\mathbf{z}_{\max}$ and $\mathbf{z}_{\min}$ are collected from calibration inputs. The TaylorMLP approximation follows the original three-step derivation:

$$\begin{aligned} y_i &= \text{Act}(\mathbf{z} + \mathbf{b}) \cdot \mathbf{W}_i + c_i \\ &\approx \left\langle \mathbf{W}_i, \sum_{n=0}^N \frac{\text{Act}^{(n)}(\mathbf{z}_0 + \mathbf{b})}{n!} \odot (\mathbf{z} - \mathbf{z}_0)^n \right\rangle + c_i \\ &= \sum_{n=0}^N \left\langle \mathbf{W}_i \odot \text{Act}^{(n)}(\mathbf{z}_0 + \mathbf{b})(n!)^{-1}, \right. \\ &\quad \left. (\mathbf{z} - \mathbf{z}_0)^n \right\rangle \\ &= \sum_{n=0}^N \langle \Theta_{i,n}, (\mathbf{z} - \mathbf{z}_0)^n \rangle. \end{aligned}$$

This form preserves inference functionality, but it also releases multiple equations involving the same hidden parameters.

3.2 Threat model and attack goal

The adversary receives the protected TaylorUnswift weights, the public transformation recipe and public metadata needed to run the released model. The adversary does not receive the original raw MLP weights, private training data, or a secret key. The attack goal is extraction: recover the original protected MLP parameters closely enough that the recovered model behaves like the original unprotected model. In our evaluation, we count a scalar weight value as recovered when its relative error is below the implementation threshold 0.01 (1%).

3.3 Algebraic weight extraction

The core observation is that TaylorMLP adds more released coefficients than hidden variables, and the coefficients are tied together elementwise. Rearranging Equation 2 gives

$$\mathbf{W}_i = (n!) \Theta_{i,n} \odot \frac{1}{\text{Act}^{(n)}(\mathbf{z}_0 + \mathbf{b})}. \quad (3)$$

Thus, once $\mathbf{z}_0 + \mathbf{b}$ is known, the original row $\mathbf{W}_i$ can be recovered from any stable Taylor order. To

solve for $\mathbf{z}_0 + \mathbf{b}$, consider two orders (a, b) satisfying $0 \leq a < b \leq N$. For every coordinate j ,

$$\frac{(a!) \Theta_{i,a}[j]}{(b!) \Theta_{i,b}[j]} = \frac{\text{Act}^{(a)}((\mathbf{z}_0 + \mathbf{b})[j])}{\text{Act}^{(b)}((\mathbf{z}_0 + \mathbf{b})[j])}. \quad (4)$$

Let

$$f_{a,b}(x) = \frac{\text{Act}^{(a)}(x)}{\text{Act}^{(b)}(x)}. \quad (5)$$

When $f_{a,b}$ is locally invertible and numerically stable on the relevant range, the released ratio determines each coordinate of $\mathbf{z}_0 + \mathbf{b}$. The adversary then subtracts the released expansion point $\mathbf{z}_0$ to obtain $\mathbf{b}$ and recovers $\mathbf{W}_i$ from the coefficient equation. This ratio argument captures the algebraic reason why TaylorMLP leaks the original weights. The attack runs in time and memory linear in the number of protected TaylorMLP parameters up to the Taylor order factor, i.e., $O(N \cdot \#\text{Params})$.

Numerical stability. The attack is algebraic, but finite-precision evaluation matters in practice. Derivative ratios such as $f_{a,b}(x)$ can be unstable near zeros of the denominator or in outlier activation regions, especially when model weights are stored in low precision. We therefore solve the recovery equations in double precision and primarily use the residual of the least-squares fit rather than a single raw ratio to select the recovered sign and weights. The remaining failures are concentrated in numerically degenerate coordinates; this explains the small unrecovered fraction observed for GeLU-based Gemma models. Appendix A visualizes the derivative-ratio stability patterns for SiLU and GeLU. Our experimental implementation uses a more stable overdetermined version of the same attack, detailed as pseudocode in Appendix B.

3.4 Experiment

Setup. We evaluate the attack on five recent open-weight instruction or reasoning checkpoints: Llama-3-8B-Instruct (Grattafiori et al., 2024), Gemma-4-E2B-it, Gemma-4-E4B-it (Google DeepMind, 2026), Qwen3.5-9B (Qwen Team, 2026a), and Qwen3.6-27B (Qwen Team, 2026b). For each model, we transform the MLP layers in every transformer block into TaylorMLP form and then run the algebraic recovery attack using only the released TaylorMLP parameters. Protected dimensions are selected by the activation-range

Model	Gemma-4-E2B-it	Gemma-4-E4B-it	Llama-3-8B-Instruct	Qwen3.5-9B	Qwen3.6-27B
# TaylorMLP Parameters	1,557,135,360	3,303,014,400	5,637,144,576	4,831,838,208	17,112,760,320
# Recovered Parameters	1,557,103,104	3,301,570,560	5,637,144,576	4,831,838,208	17,112,760,320
Recovered Ratio	99.9979%	99.9563%	100.0000%	100.0000%	100.0000%
Running Time	29.19 s	55.41 s	106.08 s	92.69 s	421.07 s
Attack Cost in USD	0.019	0.035	0.067	0.059	0.268

Table 1: Attack performance on the protected MLP parameters of recent open-weight LLM families. “# TaylorMLP Parameters” is the number of MLP parameters transformed into TaylorMLP form, and “# Recovered Parameters” is the number recovered by our attack. “Recovered Ratio” counts a weight value as recovered when its relative error is below the implementation threshold 0.01 (1%).

Model	Result	MATH-500	AIME 2026	GPQA-Diamond	LiveCodeBench v6
Gemma-4-E2B-it	Original	61.6	36.7	44.1	42.3
	Recovered	61.2	36.8	43.4	40.9
	Delta	−0.4	+0.1	−0.7	−1.4
Gemma-4-E4B-it	Original	84.4	43.3	56.8	53.3
	Recovered	84.7	42.7	55.7	53.8
	Delta	+0.3	−0.6	−1.1	+0.5
Qwen3.5-9B	Original	91.2	91.7	81.1	63.4
	Recovered	92.0	91.6	82.7	62.7
	Delta	+0.8	−0.1	+1.6	+0.7
Qwen3.6-27B	Original	96.4	94.4	86.9	84.5
	Recovered	96.2	94.0	86.3	83.2
	Delta	−0.2	−0.4	−0.6	−1.3

Table 2: End-task benchmark accuracy for original and recovered models. Delta rows show the difference (Recovered − Original).

Model	Tokens	KL mean	KL std	KL median
Gemma-4-E2B-it	111,930	0.000684	0.005759	0.000004
Gemma-4-E4B-it	111,938	0.000772	0.009950	0.000006
Llama-3-8B-Instruct	109,838	0	0	0
Qwen3.5-9B	120,150	0.000030	0.000382	0
Qwen3.6-27B	120,148	0.003996	0.155695	0

Table 3: Teacher-forcing KL divergence on MATH-500 between the original model and the model reconstructed from recovered weights. All rows use Top-K = 32; the reported KL is $KL(\text{original}||\text{recovered})$ over the top-32 token distribution collected by SGLang.

rule used by TaylorMLP, following the original TaylorUnswift release configuration. For SiLU-based Llama and Qwen MLPs and GeLU-based Gemma MLPs, the recovery implementation uses the corresponding activation derivatives and solves the released recovery equations in double precision. Our artifact is available at <https://github.com/vegetable-yx/TaylorAttack>.

Results. Table 1 shows that the attack recovers essentially all protected MLP parameters across all five models. The Llama and Qwen models are recovered exactly under the 0.01 (1%) relative-error criterion. The Gemma models have a small num-

ber of unrecovered columns caused by numerically degenerate GeLU derivative ratios, but the recovered ratios remain 99.9979% for Gemma-4-E2B-it and 99.9563% for Gemma-4-E4B-it. The largest model, Qwen3.6-27B, is recovered in about seven minutes on a single GH200 instance.

Table 3 reports the MATH-500 KL results, and Appendix C provides both MATH-500 and GPQA-Diamond. The recovered Llama and Qwen models have near-zero KL in most settings, consistent with almost exact parameter recovery. Gemma shows small but nonzero KL because a small number of GeLU-protected columns are not algebraically recoverable with stable ratios, yet the average divergence remains low. The benchmark results in Table 2 further show that recovered models stay close to the original models on end tasks.

3.5 Takeaways

TaylorUnswift’s TaylorMLP-based release transformation is vulnerable to weight extraction, and therefore fails the most basic recovery-style security goal considered in this paper. The attack also undermines the intended efficiency gap: after recovering the original weights, an adversary can run

the ordinary unprotected inference path instead of the slower protected one.

More broadly, the failure illustrates why LLM weight protection schemes need explicit adversarial definitions. TaylorUnswift evaluates utility and slowdown, but without a formal game it does not rule out the algebraic adversary above. The next section turns this lesson into a general game-based framework for weight protection schemes.

4 Security of Weight Protection Schemes

Informal claims such as “the protected representation hides the original weights” as in the original TaylorUnswift scheme are not precise enough for security analysis. This section gives a game-based language for LLM weight protection schemes. While the goal of formal security definitions is not to cover every possible deployment requirements, the motivation is to make common claims explicit enough that they can be proved, compared, or refuted by an adversary.

4.1 Syntax and correctness

Consider a learning task where training on a dataset $D \in \mathcal{D}$ produces model weights $W \in \mathcal{W}$. The original inference procedure takes an input $x \in \mathcal{X}$ and returns an output $y \in \mathcal{Y}$.

Syntax. A weight protection scheme is a tuple of probabilistic algorithms

$$\Sigma = (\text{Train}, \text{Run}, \text{KGen}, \text{Protect}, \text{RunP}), \quad (6)$$

where

- **Train** : $\mathcal{D} \rightarrow \mathcal{W}$ abstracts the original training process and outputs weights W .
- **Run** : $\mathcal{W} \times \mathcal{X} \rightarrow \mathcal{Y}$ abstracts original inference with unprotected weights.
- **KGen** : $\perp \rightarrow \mathcal{K}_{\text{sk}} \times \mathcal{K}_{\text{pk}}$ generates private and public keys. For keyless schemes such as TaylorUnswift, both key spaces may be empty.
- **Protect** : $\mathcal{K}_{\text{sk}} \times \mathcal{W} \rightarrow \mathcal{W}_p$ transforms original weights into protected weights.
- **RunP** : $\mathcal{K}_{\text{pk}} \times \mathcal{W}_p \times \mathcal{X} \rightarrow \mathcal{Y}$ runs inference with protected weights.

This syntax intentionally abstracts over many implementation choices. For example, $\mathcal{W}_p$ may be a different tensor format from $\mathcal{W}$, a decomposed representation, a watermarked representation, or a TaylorMLP-style coefficient representation.

Property	Adversary goal	Typical failure witness
X-Res	Recover original weights	Extracted weights close to W
F-Inf	Bypass protected inference cost	Same output with lower cost
D-Priv	Distinguish protected source	Guess which W_b was protected

Table 4: Security properties for LLM weight protection schemes. X-Res is a recovery-style property, F-Inf is an efficient-use property, and D-Priv is a distinguishing-style property.

Game-based definitions. We use games to specify security properties, following standard cryptographic practice (Goldwasser and Micali, 1984; Bellare and Rogaway, 2004; Shoup, 2004). A game is a probabilistic experiment that samples the scheme state, gives an adversary a specified view, and returns whether the adversary succeeds. The adversary’s success probability or distinguishing advantage then becomes the object of analysis.

Correctness. A protected model must remain useful. Let $\text{Dist}(a, b)$ be a task-appropriate distance function, and let $\text{Same}_\tau(a, b)$ denote $\text{Dist}(a, b) \leq \tau$ for a threshold τ . For logits, Dist may be KL divergence or norm distance; for weights, it may be relative error or a matrix norm. Correctness requires protected inference to match original inference with high probability.

Definition 4.1 (Correctness of a weight protection scheme). We define the following game.

Game_{Correct}(D)	
1 :	$(\text{pk}, \text{sk}) \leftarrow \text{KGen}()$
2 :	$W \leftarrow \text{Train}(D)$
3 :	$W_p \leftarrow \text{Protect}(\text{sk}, W)$
4 :	$x \leftarrow \mathcal{X}$
5 :	$y \leftarrow \text{Run}(W, x)$
6 :	$y_p \leftarrow \text{RunP}(\text{pk}, W_p, x)$
7 :	return $\text{Same}_{\tau_y}(y_p, y)$

A scheme Σ is (D, τ_y, ϵ) -correct if

$$\Pr[\text{Game}_{\text{Correct}}(D) \Rightarrow 1] \geq 1 - \epsilon. \quad (7)$$

4.2 Security properties

Table 4 summarizes the three adversarial goals considered in this paper. They are deliberately phrased as families of games rather than as one universal definition, because different protection schemes may target different security goals.

Extraction resistance. The most direct security goal is to prevent an adversary from recovering the original weights from the protected weights. This captures the failure mode in §3.

Game_{X-Res}($D, \mathcal{A}$)

```

1 :  $W \leftarrow \$ \text{Train}(D)$ 
2 :  $(\text{pk}, \text{sk}) \leftarrow \$ \text{KGen}()$ 
3 :  $W_p \leftarrow \$ \text{Protect}(\text{sk}, W)$ 
4 :  $W^* \leftarrow \$ \mathcal{A}(\text{pk}, W_p)$ 
5 : return  $\text{Same}_{\tau_w}(W^*, W)$ 

```

The adversary's advantage is

$$\text{Adv}_{\Sigma}^{\text{X-Res}}(D, \mathcal{A}) = \Pr[\mathbf{Game}_{\text{X-Res}}(D, \mathcal{A}) \Rightarrow 1]. \quad (8)$$

Definition 4.2 (Extraction resistance). A scheme Σ is (D, τ_w, ϵ) -extraction-resistant, or X-Res-secure, if for all efficient adversaries $\mathcal{A}$,

$$\text{Adv}_{\Sigma}^{\text{X-Res}}(D, \mathcal{A}) \leq \epsilon. \quad (9)$$

X-Res is a weak but useful baseline. It does not rule out every practical abuse, because an adversary may bypass a scheme without exactly reconstructing the original weights. However, a successful X-Res adversary is a strong proof of insecurity: if the original weights can be extracted from the protected artifact, then the protection mechanism has failed at a basic confidentiality goal.

Efficient-use resistance. Many weight protection schemes intend to preserve local inference while making unauthorized high-throughput inference less attractive. This goal is not captured by exact weight recovery alone. Let $\text{Cost}(F, u)$ measure the cost of running a procedure F on input u , instantiated by running time, FLOPs, memory, or a concrete deployment metric.

Game_{F-Inf}($D, \mathcal{A}, t$)

```

1 :  $W \leftarrow \$ \text{Train}(D)$ 
2 :  $(\text{pk}, \text{sk}) \leftarrow \$ \text{KGen}()$ 
3 :  $W_p \leftarrow \$ \text{Protect}(\text{sk}, W)$ 
4 :  $x \leftarrow \$ \mathcal{X}$ 
5 :  $y_p \leftarrow \$ \text{RunP}(\text{pk}, W_p, x)$ 
6 :  $y^* \leftarrow \$ \mathcal{A}(\text{pk}, W_p, x)$ 
7 : return  $\text{Same}_{\tau_y}(y^*, y_p)$ 

```

The adversary is valid only when $\text{Cost}(\mathcal{A}, (\text{pk}, W_p, x)) \leq t$, where t is below the intended protected-inference cost. The advantage is

$$\text{Adv}_{\Sigma}^{\text{F-Inf}}(D, \mathcal{A}, t) = \Pr[\mathbf{Game}_{\text{F-Inf}}(D, \mathcal{A}, t) \Rightarrow 1]. \quad (10)$$

Definition 4.3 (Efficient-use resistance). A scheme Σ is (D, τ_y, t, ϵ) -efficient-use-resistant, or F-Inf-secure, if for all efficient adversaries $\mathcal{A}$ with cost at most t ,

$$\text{Adv}_{\Sigma}^{\text{F-Inf}}(D, \mathcal{A}, t) \leq \epsilon. \quad (11)$$

This definition models attacks that produce the same outputs as protected inference while using less cost than the protected inference path. Exact extraction is one way to win this game when original inference is cheaper than protected inference, but it is not the only way: quantization, compilation, distillation, or format conversion may also be relevant efficient-use attacks.

Distinguishing privacy. The distinguishing-style family asks whether protected weights leak information about which original weights generated them. It is inspired by indistinguishability definitions in cryptography, but adapted to weight artifacts.

Game_{D-Priv}($D, \mathcal{A}$)

```

1 :  $b \leftarrow \$ \{0, 1\}$ 
2 :  $(\text{pk}, \text{sk}) \leftarrow \$ \text{KGen}()$ 
3 :  $W_0 \leftarrow \$ \text{Train}(D)$ 
4 :  $W_1 \leftarrow \$ \text{Train}(D)$ 
5 : if  $\text{Same}_{\tau_w}(W_0, W_1)$  then return 0
6 :  $W_p \leftarrow \$ \text{Protect}(\text{sk}, W_b)$ 
7 :  $b' \leftarrow \$ \mathcal{A}(\text{pk}, W_0, W_1, W_p)$ 
8 : return  $b = b'$ 

```

The distinguishing advantage is

$$\text{Adv}_{\Sigma}^{\text{D-Priv}}(D, \mathcal{A}) = 2 \left| \Pr[\mathbf{Game}_{\text{D-Priv}}(D, \mathcal{A}) \Rightarrow 1] - \frac{1}{2} \right|. \quad (12)$$

Definition 4.4 (Distinguishing privacy). A scheme Σ is (D, τ_w, ϵ) -distinguishing-private, or D-Priv-secure, if for all efficient adversaries $\mathcal{A}$,

$$\text{Adv}_{\Sigma}^{\text{D-Priv}}(D, \mathcal{A}) \leq \epsilon. \quad (13)$$

D-Priv captures a stronger secrecy intuition than X-Res. Instead of asking only whether the adversary can reconstruct W , it asks whether the protected artifact leaks enough information to link it to one of two candidate original weights. The exact sampling condition can be specialized for a concrete scheme, for example by requiring two independently trained weights, two fine-tuned variants, or two weights sampled from a deployment-relevant distribution.

4.3 Relations among properties

The properties above are not independent. Under natural cost and correctness assumptions, attacks against weaker goals can be converted into attacks against stronger ones. We state the relations here and defer proofs to Appendix J.

F-Inf security implies X-Res security. Suppose protected inference is intended to be more expensive than original inference, and suppose extraction is cheap enough that extraction followed by original inference costs less than protected inference. Then an X-Res attack immediately yields an F-Inf attack: recover W from W_p , run $\text{Run}(W, x)$, and output the result. By contraposition, any scheme secure against efficient-use attacks is also secure against extraction in that cost regime.

D-Priv security implies X-Res security. An extraction adversary yields a detection adversary: given (W_0, W_1, W_p) , recover W^* from W_p and output the index of the candidate closest to W^* . If extraction succeeds and the candidates are separated by the Same_{τ_w} predicate, detection succeeds. Thus D-Priv is at least as strong as X-Res for the corresponding weight distribution and threshold.

Implication for TaylorUnswift. The attack in §3 constructs an efficient X-Res adversary for the TaylorMLP representation. Because recovered weights can be used in the ordinary inference path, the same attack also threatens the efficient-use barrier whenever original inference is cheaper than protected TaylorMLP inference. This is why the attack is more than a numerical reconstruction result: it is a formal witness that the scheme fails a basic game-based security goal.

5 Towards Future Schemes

The TaylorUnswift case study and the games in §4 suggest several principles for future LLM weight protection schemes.

State the security goal before designing the transformation. A protection scheme should identify which adversarial goal it targets: extraction resistance, efficient-use resistance, distinguishing privacy, ownership attribution, or tamper resistance. These goals are not interchangeable: preventing exact parameter recovery does not preclude a faster functionally equivalent model, and a watermark may support attribution without hiding weights.

Designs should begin with a precise game, adversary view, success predicate, and cost model.

Justify security with assumptions or reductions. A claim that a transformation “looks hard to invert” is not a security argument; one should explain why the representation does not leak enough structure to win the relevant game, ideally via a reduction to a well-studied primitive. If no proof is available, state the missing assumption rather than presenting utility measurements as security evidence.

Run offensive evaluations against the stated game. Formal definitions make attack experiments sharper by specifying what counts as a break. Offensive evaluation should include algebraic recovery, numerical inversion, format conversion, quantization, distillation, and any attack suggested by the public release representation. Successful attacks should be treated as proofs of insecurity for the corresponding game.

Preserve deployability without hiding the threat model. Practical schemes must integrate with common inference frameworks and model formats, but an easily runnable representation is equally easy to inspect, transform, and optimize. Future work should make this tradeoff explicit by reporting both utility and adversarial security metrics.

6 Conclusion

We revisited LLM weight protection schemes through the lens of adversarial games. Our case study shows that TaylorUnswift’s protection can be efficiently inverted: the released Taylor coefficients enable recovery of nearly all protected MLP parameters across recent open-weight LLMs, and the recovered models closely match the originals. This break illustrates a broader lesson: detailed utility evaluation does not establish security unless the adversary, success condition, and assumptions are made explicit. To support more rigorous analysis, we formulated weight protection syntax, correctness, and game-based notions for extraction resistance, efficient-use resistance, and distinguishing privacy. These definitions provide a vocabulary for future schemes to state what they protect, for reviewers to evaluate whether evidence matches the claim, and for attackers to produce clear proofs of insecurity. We hope this work encourages future research to combine formal security goals, proof-oriented reasoning, and offensive evaluation before protected weights are released.

Limitations

Our definitions cover several core security goals for LLM weight protection, but they do not model every deployment risk. They abstract away side channels, implementation bugs, hardware-specific leakage, license enforcement, and operational controls around model distribution. Other protection goals, such as robust watermark attribution or post-release tamper resistance, may require additional games.

Our empirical case study focuses on TaylorUnswift’s TaylorMLP transformation and on extraction through released Taylor coefficients. The same game-based methodology can guide analysis of other schemes, but each mechanism may require its own assumptions and attack surface analysis. The recovery rates may also depend on activation functions, numerical precision, model architecture, and protected-dimension selection, so broader evaluations are needed before generalizing the empirical findings beyond the TaylorUnswift-style setting.

References

- Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. 2018. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX Security Symposium*, pages 1615–1631.
- Mihir Bellare and Phillip Rogaway. 2004. [Code-based game-playing proofs and the security of triple encryption](#). *IACR Cryptol. ePrint Arch.*, 2004:331.
- Adam Block, Ayush Sekhari, and Alexander Rakhlin. 2025. [GaussMark: a practical approach for structural watermarking of language models](#). *CoRR*, abs/2501.13941.
- Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, and 95 others. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.
- Huili Chen, Bitu Darvish Rouhani, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. 2019. [DeepMarks: A secure fingerprinting framework for digital rights management of deep learning models](#). In *Proceedings of the 2019 ACM International Conference on Multimedia Retrieval*, pages 105–113.
- Xuxin Cheng, Ke Zeng, Zhiquan Cao, Linyi Dai, Wenxuan Gao, Fei Han, Ai Jian, Feng Hong, Wenxing Hu, Ziheng Huang, Dejian Kong, Jia Leng, Zhuoyuan Liao, Pei Liu, Jiaye Lin, Xing Ma, Jingqing Ruan, Jiaxing Song, Xiaoyu Tan, and 49 others. 2025. [Higher satisfaction, lower cost: A technical report on how llms revolutionize meituan’s intelligent interaction systems](#). *CoRR*, abs/2510.13291.
- DeepSeek-AI. 2026. [DeepSeek-V4: Towards highly efficient million-token context intelligence](#). *Preprint*, arXiv:2606.19348.
- Ye Dong, Wen-jie Lu, Yancheng Zheng, Haoqi Wu, Derun Zhao, Jin Tan, Zhicong Huang, Cheng Hong, Tao Wei, and Wenguang Chen. 2023. [PUMA: secure inference of llama-7b in five minutes](#). *CoRR*, abs/2307.12533.
- Lixin Fan, Kam Woh Ng, Chee Seng Chan, and Qiang Yang. 2019. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. In *Advances in Neural Information Processing Systems*, volume 32.
- Yudong Gao, Qingyue Wang, Yuanyuan Yuan, Ruixuan Huang, Linghan Chen, Zimo Ji, and Shuai Wang. 2026. [Pathmark: Protecting intellectual property of mixture-of-expert LLMs via path watermarks](#). In *Proceedings of the 2026 ACM SIGSAC Conference on Computer and Communications Security*.
- Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. 2016. CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 201–210.
- Thibaud Gloaguen, Robin Staab, Nikola Jovanović, and Martin Vechev. 2025. [LLM fingerprinting via semantically conditioned watermarks](#). *CoRR*, abs/2505.16723.
- Shafi Goldwasser and Silvio Micali. 1984. [Probabilistic encryption](#). *J. Comput. Syst. Sci.*, 28(2):270–299.
- Xueluan Gong, Yanjiao Wang, Huayang Li, Yang Liu, and Xiaosong Qiu. 2020. [Model extraction attacks and defenses on cloud-based machine learning models](#). *IEEE Communications Magazine*, 58(12):83–89.
- Google DeepMind. 2026. [Gemma 4: Byte for byte, the most capable open models](#). Google DeepMind Technical Report.
- 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, and 51 others. 2024. [The llama 3 herd of models](#). *CoRR*, abs/2407.21783.
- Fuchun Guo, Willy Susilo, Xiaofeng Chen, Peng Jiang, Jianchang Lai, and Zhen Zhao. 2023. Sok: Research motivations of public-key cryptography. *Cryptology ePrint Archive*.

- Matthew Jagielski, Nicholas Carlini, David Berthelot, Alexey Kurakin, and Nicolas Papernot. 2020. High accuracy and high fidelity extraction of neural networks. In *29th USENIX Security Symposium*, pages 1345–1362.
- Heng Jin, Chaoyu Zhang, Shanghao Shi, Wenjing Lou, and Y Thomas Hou. 2024. Proflingo: A fingerprinting-based intellectual property protection scheme for large language models. In *2024 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9. IEEE.
- Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. 2018. GAZELLE: A low latency framework for secure neural network inference. In *27th USENIX Security Symposium*, pages 1651–1669.
- Neal Koblitz and Alfred Menezes. 2007. Another look at "provable security". *J. Cryptol.*, 20(1):3–37.
- Bowen Li, Lixin Fan, Hanlin Gu, Jie Li, and Qiang Yang. 2023. FedIPR: Ownership verification for federated deep neural network models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4521–4536.
- Qinfeng Li, Miao Pan, Jintao Chen, Fu Teng, Zhiqiang Shen, Ge Su, Hao Peng, and Xuhong Zhang. 2026. Do not merge my model! safeguarding open-source LLMs against unauthorized model merging. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40.
- Shen Li, Liuyi Yao, Jinyang Gao, Lan Zhang, and Yaliang Li. 2024. Double-i watermark: Protecting model copyright for LLM fine-tuning. *CoRR*, abs/2402.14883.
- Wen-jie Lu, Zhicong Huang, Zhen Gu, Jingyu Li, Jian Liu, Cheng Hong, Kui Ren, Tao Wei, and Wenguang Chen. 2025. Bumblebee: Secure two-party inference framework for large transformers. In *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025*. The Internet Society.
- Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. 2020. Delphi: A cryptographic inference service for neural networks. In *29th USENIX Security Symposium*, pages 2505–2522.
- Payman Mohassel and Yupeng Zhang. 2017. SecureML: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy*, pages 19–38.
- Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. 2019. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4954–4963.
- Qi Pang, Jinhao Zhu, Helen Möllering, Wenting Zheng, and Thomas Schneider. 2024. BOLT: privacy-preserving, accurate and efficient inference for transformers. In *IEEE Symposium on Security and Privacy, SP 2024, San Francisco, CA, USA, May 19-23, 2024*, pages 4753–4771. IEEE.
- Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519.
- Qwen Team. 2025. Qwen3 technical report. Preprint, arXiv:2505.09388.
- Qwen Team. 2026a. Qwen3.5: Towards native multi-modal agents.
- Qwen Team. 2026b. Qwen3.6-27B: Flagship-level coding in a 27B dense model.
- Yehonathan Refael, Adam Hakim, Lev Greenberg, Tal Aviv, Satya Lokam, Ben Fishman, and Shachar Seidman. 2024. SLIP: securing llms IP using weights decomposition. *CoRR*, abs/2407.10886.
- Bitva Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. 2019. DeepSigns: An end-to-end watermarking framework for ownership protection of deep neural networks. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 485–497.
- Victor Shoup. 2004. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptol. ePrint Arch.*, 2004:332.
- Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, Miles McCain, Alex Newhouse, Jason Blazakis, Kris McGuffie, and Jasmine Wang. 2019. Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*.
- Sebastian Szyller, Buse Gul Atli, Samuel Marchal, and N. Asokan. 2021. DAWN: Dynamic adversarial watermarking of neural networks. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 4417–4425.
- Rishub Tamirisa, Bhurugu Bharathi, Long Phan, Andy Zhou, Alice Gatti, Tarun Suresh, Maxwell Lin, Justin Wang, Rowan Wang, Ron Arel, Andy Zou, Dawn Song, Bo Li, Dan Hendrycks, and Mantas Mazeika. 2025. Tamper-resistant safeguards for open-weight llms. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.
- Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction apis. In *25th USENIX Security Symposium*, pages 601–618.

Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. 2017. [Embedding watermarks into deep neural networks](#). In *Proceedings of the 2017 ACM International Conference on Multimedia Retrieval*, pages 269–277.

Guanchu Wang, Yu-Neng Chuang, Ruixiang Tang, Shaochen Zhong, Jiayi Yuan, Hongye Jin, Zirui Liu, Vipin Chaudhary, Shuai Xu, James Caverlee, and Xia Ben Hu. 2024. [Taylor unswift: Secured weight release for large language models via taylor expansion](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 6928–6941. Association for Computational Linguistics.

Yan Wen, Junfeng Guo, and Heng Huang. 2025. [Cot-guard: Using chain-of-thought triggering for copyright protection in multi-agent LLM systems](#). *CoRR*, abs/2505.19405.

Yijie Xu, Aiwei Liu, Xuming Hu, Lijie Wen, and Hui Xiong. 2025. [Mark your LLM: detecting the misuse of open-source large language models via watermarking](#). *CoRR*, abs/2503.04636.

Biwei Yan, Kun Li, Minghui Xu, Yueyan Dong, Yue Zhang, Zhaochun Ren, and Xiuzhen Cheng. 2024. [On protecting the data privacy of large language models \(llms\): A survey](#). *CoRR*, abs/2403.05156.

Yuliang Yan, Haochun Tang, Shuo Yan, and Enyan Dai. 2025. [DuFFin: a dual-level fingerprinting framework for LLMs IP protection](#). *CoRR*, abs/2505.16530.

Xin Yang, Letian Li, Abudukelimu Wuerkaixi, Xuxin Cheng, Cao Liu, Ke Zeng, Xunliang Cai, and Wenyuan Jiang. 2026. [Towards self-robust llms: Intrinsic prompt noise resistance via coipo](#). *CoRR*, abs/2603.03314.

A Analysis of numerical stability of SiLU/GeLU derivative

The numerical stability of higher-order derivatives of the SiLU and GeLU activation function exhibits significant sensitivity when evaluating ratios of derivatives. As demonstrated in Fig.A1 and Fig.A2, ratios of SiLU derivatives within the interval $[-10, 10]$ frequently encounter numerical instability, often resulting in large fluctuations or undefined values at various points away from zero. While stability improves in the vicinity of zero due to more balanced numerical magnitudes, even slight deviations or outliers in the input distributions can cause pronounced instabilities. Importantly, since different network architectures and model sizes inherently produce distinct internal numerical distributions, the resulting numerical instabilities manifest differently across these models, leading to varying degrees of reconstruction errors and inaccuracies. Therefore, the numerical instability induced by varying distributions across different model architectures and scales inevitably introduces a certain degree of error into our reconstruction method, underscoring the necessity for careful numerical considerations when employing higher-order SiLU derivatives in analysis and optimization tasks.

B Numerically Stable Recovery Implementation

Algorithm 2 summarizes the recovery implementation used in our experiments. It is the same algebraic attack described in §3, but it uses all released Taylor orders and all output rows to form an overdetermined least-squares problem for each protected dimension. The direct derivative-ratio inversion is retained only as a fallback for degenerate cases.

C KL Results

Table A1 reports the complete teacher-forcing KL divergence results for MATH-500 and GPQA-Diamond. All entries use SGLang, top-32 log probabilities, and an 8k context window.

D MergeBarrier and the LWE Claim

MergeBarrier (Li et al., 2026) targets unauthorized model merging, while our X-Res adversary targets weight extraction from publicly released artifacts. This threat-model difference is important context:

Model	MATH-500				GPQA-Diamond			
	Tokens	KL mean	KL std	KL median	Tokens	KL mean	KL std	KL median
Gemma-4-E2B-it	111,930	0.000684	0.005759	0.000004	54,257	0.001737	0.008644	0.000056
Gemma-4-E4B-it	111,938	0.000772	0.009950	0.000006	54,127	0.002138	0.036509	0.000103
Llama-3-8B-Instruct	109,838	0	0	0	55,053	0	0	0
Qwen3.5-9B	120,150	0.000030	0.000382	0	57,591	0.000011	0.000208	0
Qwen3.6-27B	120,148	0.003996	0.155695	0	57,589	0	0	0

Table A1: Complete KL divergence results with benchmarks shown as column groups. KL is computed as $\text{KL}(\text{original} \parallel \text{recovered})$ on the top-32 token distribution.

Algorithm 2 Numerically stable TaylorMLP recovery for one protected MLP

Require: Public TaylorMLP tensors $L, F, \tilde{G}, p, U$ and activation Act

Ensure: Recovered sampled weights (G^*, U^*, D^*) and signs s^*

- 1: Let $L = \text{local_approx_output}$, $F = \text{fuse_weight}$, $\tilde{G} = \text{gate_proj_weight}$, $p = \text{local_point}$, and $U = \text{up_proj_weight}$.
- 2: Precompute coefficient vectors $c_s[j]$ for $s \in \{-1, +1\}$ and each protected dimension j .
- 3: **for** each protected dimension j **do**
- 4: Form O_j by concatenating $L[:, j]$ and all released Taylor-order columns $F[:, j, n]$.
- 5: Discard non-finite rows and set $\text{best} \leftarrow \perp$.
- 6: **for** $s \in \{-1, +1\}$ **do**
- 7: Build $c_s[j]$ from $\text{Act}^{(n)}(sp_j)$ and the sign factor s^n for all orders n .
- 8: Fit the down-projection column by projection: $d_s \leftarrow O_j c_s[j]^\top / \|c_s[j]\|_2^2$.
- 9: Compute the normalized residual r_s between O_j and $d_s c_s[j]$.
- 10: Update best if r_s is smaller.
- 11: **end for**
- 12: **if** no finite least-squares candidate is available **then**
- 13: Estimate a stable ratio from $F[:, j, 2]/L[:, j]$.
- 14: Choose s by ratio inversion over $\{+|p_j|, -|p_j|\}$ and refit d_s .
- 15: **end if**
- 16: Set $s_j^* \leftarrow s$, $D^*[:, j] \leftarrow d_s$, and $G^*[j, :] \leftarrow s \tilde{G}[j, :]$.
- 17: **end for**
- 18: Copy $U^* \leftarrow U$ and copy unprotected FFN weights from the released representation.
- 19: **return** (G^*, U^*, D^*, s^*) .

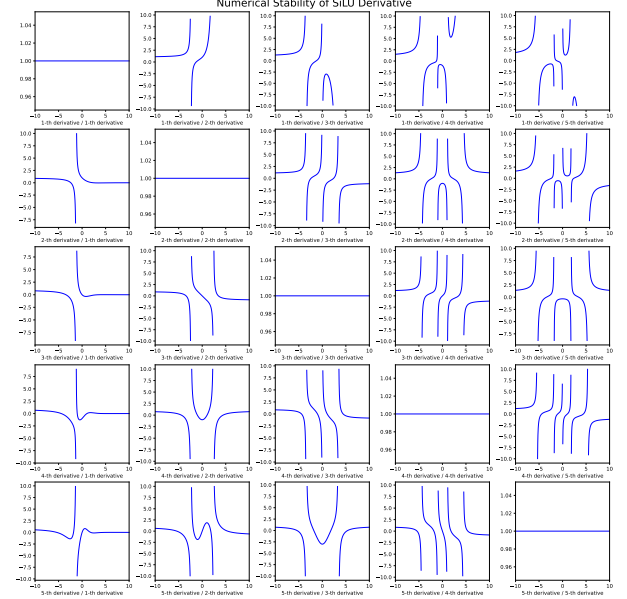


Figure A1: Numerical stability analysis of SiLU derivative ratios. Each subplot shows the ratio between SiLU derivatives from first to fifth order, evaluated over the interval. Diagonal entries represent ratios of derivatives with themselves and therefore equal 1. Many off-diagonal entries exhibit significant numerical instability, characterized by large fluctuations or extreme values, particularly away from zero. These instabilities highlight potential sources of errors and inaccuracies in practical computations involving higher-order SiLU derivatives.

our findings are limited to MergeBarrier’s weight-confidentiality aspect and do not evaluate its anti-merging objective.

That said, MergeBarrier’s FFN reparameterization releases Taylor coefficient tensors with the same structure attacked in this paper. Given those released coefficients, our adversary solves the overdetermined equation system and back-substitutes to recover the original parameters. We added results for LLaMA-2-13B, the same model family used in MergeBarrier’s experiments:

Regarding the LWE hardness argument, we

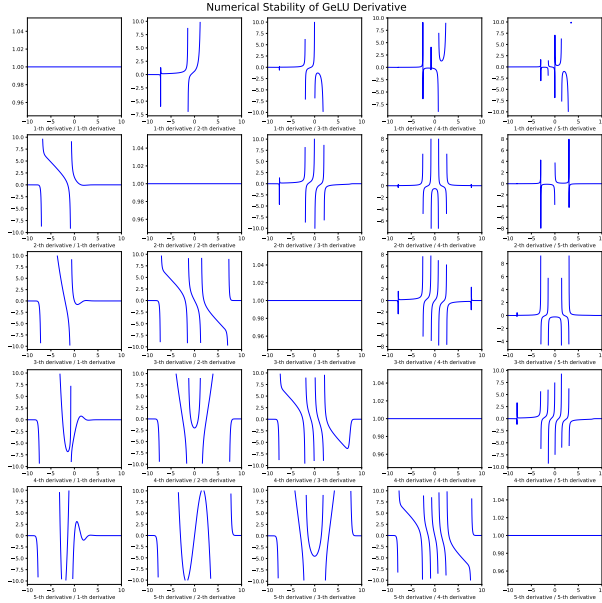


Figure A2: Numerical stability analysis of GeLU derivative ratios. Similar to SiLU, ratios between GeLU derivatives from first to fifth order show numerical instabilities characterized by large fluctuations and extreme values, particularly away from zero. Although minor differences exist, overall stability patterns closely resemble those observed for SiLU.

briefly note two issues: establishing hardness would require a reduction from LWE to weight inversion, whereas MergeBarrier argues in the reverse direction; and the floating-point parameter setting does not correspond to any widely used standard LWE parameterization, so the algebraic resemblance alone does not transfer to LWE hardness. Our attack also does not assume that the Taylor expansion has zero remainder: the remainder is a runtime approximation residual rather than an independently sampled noise term attached to each coefficient equation.

Model	Ratio@0.1%	Ratio@0.01%	Mean rel. Fro
LLaMA-2-13B	100.00%	100.00%	1.29×10^{-8}
Llama-3-8B-Instruct	100.00%	100.00%	1.317×10^{-8}
Gemma-4-E2B-it	100.00%	100.00%	0.0005087
Gemma-4-E4B-it	99.96%	99.96%	0.002925
Qwen3.5-9B	100.00%	100.00%	1.329×10^{-8}
Qwen3.6-27B	100.00%	100.00%	1.344×10^{-8}

Table A2: Weight recovery on the FFN parameters of MergeBarrier’s released coefficient format, including LLaMA-2-13B (the same model family used in MergeBarrier’s experiments). Under the default release setting (fp32 coefficients), recovery is near-perfect for all models, and the stricter 0.1%/0.01% thresholds are not decisive for the extraction conclusion.

Model	Setting	Ratio@0.1%	Failed dims
Llama-3-8B-Instruct	order6_float32_noise0	100.00%	0
Llama-3-8B-Instruct	order6_bfloat16_noise0	88.31%	0
Llama-3-8B-Instruct	order2_bfloat16_noise0	91.55%	0
Llama-3-8B-Instruct	order6_bfloat16_noise1e-3	86.55%	0
Qwen3.5-9B	order6_float32_noise0	100.00%	0
Qwen3.5-9B	order6_bfloat16_noise0	88.53%	0
Qwen3.5-9B	order2_bfloat16_noise0	91.05%	0
Qwen3.5-9B	order6_bfloat16_noise1e-3	86.68%	0
Gemma-4-E4B-it	order6_float32_noise0	99.74%	79
Gemma-4-E4B-it	order6_bfloat16_noise0	87.70%	79
Gemma-4-E4B-it	order2_bfloat16_noise0	88.12%	362
Gemma-4-E4B-it	order6_bfloat16_noise1e-3	80.33%	7558

Table A3: 4-layer ablation over Taylor order, released coefficient precision, and injected coefficient noise. The full-model experiments use the default setting (bf16 model, fp32 released coefficients, no noise).

E Truncation Order, Precision, and Coefficient Noise

To map the effective range of the attack, we ran a 4-layer ablation varying the Taylor order, the precision of the released coefficients, and injected coefficient noise. The full-model results use the default setting (bf16 model, fp32 released coefficients, no noise); the ablation below stresses these settings:

Four findings emerge. (1) Under the release setting used in the main experiment (fp32 coefficients), recovery is near-perfect for all models. (2) bf16 coefficient precision reduces elementwise ratios primarily due to rounding, but the attack remains effective at the dimension level (0 failed dims for Llama/Qwen). (3) Lower Taylor order (order 2) does not significantly degrade recovery beyond the bfloat16 effect, confirming that the attack does not rely on high-order expansion. (4) Explicit coefficient noise at 10^{-3} shows the expected degradation, defining the effective range of the attack.

F Naive Single-Ratio Baseline

We compare the stable least-squares solver with a naive single-ratio method that estimates each sampled dimension using only the ratio between the second-order and zero-order coefficients, without the overdetermined multi-order least-squares solver. Both methods see the same protected artifact.

G Full-Vocabulary KL Divergence

Table A5 reports exact full-vocabulary KL divergence on MATH-500 for all five models. Each row covers 500 examples, with KL computed over the complete vocabulary at every position.

Model	Method	Ratio@0.1%	Failed dims	Mean rel. Fro
Llama-3-8B-Instruct	Stable LS	88.31%	0	0.0002084
Llama-3-8B-Instruct	Naive single-ratio	85.23%	333	0.03573
Qwen3.5-9B	Stable LS	88.53%	0	0.0002054
Qwen3.5-9B	Naive single-ratio	85.11%	334	0.03991
Gemma-4-E4B-it	Stable LS	87.70%	79	0.01739
Gemma-4-E4B-it	Naive single-ratio	85.05%	582	0.04382

Table A4: Comparison between the stable least-squares solver and a naive single-ratio baseline on the same 4-layer protected artifacts. Stable LS uses all released Taylor orders and rows, improving robustness under finite-precision release. The naive baseline’s higher failed-dim and Frobenius error quantify the improvement.

Model	KL mean	KL median	KL p99	KL max
Llama-3-8B-Instruct	0	0	0	0
Gemma-4-E2B-it	0.00117546	3.3676×10^{-6}	0.0143646	1.68734
Gemma-4-E4B-it	0.000995345	8.02027×10^{-6}	0.0121234	3.31804
Qwen3.5-9B	0	0	0	0
Qwen3.6-27B	0	0	0	0

Table A5: Exact full-vocabulary KL divergence on MATH-500 between the original model and the model reconstructed from recovered weights. Llama-3-8B-Instruct, Qwen3.5-9B, and Qwen3.6-27B have zero mean/median/p99/max divergence. Gemma models have small median ($< 10^{-5}$) and p99 (< 0.015) divergence, confirming that most token positions remain close; the rare nonzero maxima are consistent with the localized GeLU recovery failures characterized in §I.

H Inference Cost Comparison

The F-Inf implication requires measured cost numbers. Table A6 compares the recovered (ordinary) MLP path with the protected TaylorMLP path under matched hidden-state shapes.

I Gemma Failure Localization

The Gemma failures are localized rather than broad. Gemma-4-E2B-it failures are concentrated in layer 4 (8 failed dims, max down-column error 1). Gemma-4-E4B-it failures are concentrated in layer 1 (79 failed dims, max down-column error 1), layer 5 (65 failed dims, max down-column error 1), and layer 6 (0 failed dims, max down-column error

Model	Slowdown
Gemma-4-E4B-it	13.25×
Llama-3-8B-Instruct	25.14×
Qwen3.5-9B	22.57×

Table A6: Inference slowdown of the protected TaylorMLP path relative to the recovered (ordinary) MLP path. Protected models are 13.25–25.14× slower than the recovered models, confirming that an adversary who recovers ordinary weights can then use the cheaper ordinary inference path.

0.262). These layers also have larger least-squares residuals, consistent with the GeLU derivative-ratio degeneracy explanation in §A.

J Proofs Sketches

F-Inf security implies X-Res security. We prove the relation by contraposition. Assume there exists an efficient adversary $\mathcal{A}$ that wins **Game**_{X-Res} with non-negligible probability. Construct an adversary $\mathcal{B}_{\mathcal{A}}$ for **Game**_{F-Inf} as follows.

$\mathcal{B}_{\mathcal{A}}(\text{pk}, W_p, x)$

1 : $W^* \leftarrow \mathcal{A}(\text{pk}, W_p)$
2 : **return** $\text{Run}(W^*, x)$

Whenever $\mathcal{A}$ outputs W^* such that $\text{Same}_{\tau_w}(W^*, W)$, correctness of the protection scheme implies that $\text{Run}(W^*, x)$ is the same as $\text{RunP}(\text{pk}, W_p, x)$ up to the output threshold τ_y . Thus $\mathcal{B}_{\mathcal{A}}$ wins whenever $\mathcal{A}$ wins, except for the scheme’s correctness error. If

$$\begin{aligned} & \text{Cost}(\mathcal{A}, (\text{pk}, W_p)) + \text{Cost}(\text{Run}, (W^*, x)) \\ & \leq t \\ & < \text{Cost}(\text{RunP}, (\text{pk}, W_p, x)), \end{aligned} \quad (14)$$

then $\mathcal{B}_{\mathcal{A}}$ is also a valid efficient-use adversary. Therefore,

$$\text{Adv}_{\Sigma}^{\text{X-Res}}(D, \mathcal{A}) \lesssim \text{Adv}_{\Sigma}^{\text{F-Inf}}(D, \mathcal{B}_{\mathcal{A}}, t), \quad (15)$$

up to correctness and threshold slack. By contraposition, F-Inf security in this cost regime implies X-Res security.

D-Priv security implies X-Res security. Again we prove by contraposition. Assume there exists an efficient extraction adversary $\mathcal{A}$ for **Game**_{X-Res}. Construct a distinguishing adversary $\mathcal{B}_{\mathcal{A}}$ as follows.

$\mathcal{B}_{\mathcal{A}}(\text{pk}, W_0, W_1, W_p)$

1 : $W^* \leftarrow \mathcal{A}(\text{pk}, W_p)$
2 : **if** $\text{Same}_{\tau_w}(W^*, W_0)$ **then return** 0
3 : **else return** 1

In **Game**_{D-Priv}, the candidates are required not to satisfy $\text{Same}_{\tau_w}(W_0, W_1)$. If $\mathcal{A}$ recovers the protected source weight, then exactly one candidate is close to W^* under the same threshold, so $\mathcal{B}_{\mathcal{A}}$ outputs the hidden bit correctly. The extra work performed by $\mathcal{B}_{\mathcal{A}}$ is only a constant number of threshold checks, so it is efficient whenever $\mathcal{A}$ is

Name	Value
Instance	Lambda GH200, single GPU
CPU	Neoverse-V2, 64 vCPU
Memory	525 GiB
GPU	NVIDIA GH200
GPU Memory	97,871 MiB
Data type	torch.bfloat16
OS	Ubuntu 24.04.4 LTS
Kernel	Linux 6.8.0-1046-nvidia-64k
Architecture	arm64
Driver/CUDA	580.105.08 / 13.0–13.0.1
Python	3.12.3
torch	2.11.0+cu130
transformers	5.6.0
sglang	0.5.12
numpy	2.3.5
scipy	1.17.1
datasets	4.8.5
huggingface_hub	1.15.0
safetensors	0.7.0
accelerate	1.13.0

Table A7: Configuration of experiment and computing infrastructure.

efficient. Therefore a successful X-Res adversary gives a successful D-Priv adversary, and by contraposition D-Priv security implies X-Res security for the same sampling distribution and threshold.

K Computational Infrastructure

Table A7 provides details on the computational infrastructure and environment information. All full-model experiments were performed on a single NVIDIA GH200 setting as described in Table A7. Attack-only runtimes are reported in Table 1. Including recovery, KL evaluation, and downstream benchmark evaluation, the experiments used at least 24 single-GH200 GPU-hours.

L Artifacts & Open Source License

The research artifacts associated with this submission are prepared for anonymous review and, where applicable, are distributed under a permissive open-source license. We rely on commonly used open-source libraries for model serving, data processing, numerical computation, and evaluation, and use them in accordance with their respective licenses. Pretrained language models are accessed through their official APIs or repositories and are subject

to the licenses and usage terms specified by their providers.

Intended use and consistency. All artifacts used in this work are employed in a manner consistent with their stated or implied intended use. The evaluation materials associated with this work are intended solely for research evaluation of LLM weight protection schemes. Pretrained language models are used for inference-only experiments under their respective research or community terms, and the benchmark data are used only for offline academic evaluation rather than for deployment in real-world decision-making systems.

M LLM Usage

We used large language models as writing aids for grammar, clarity, and consistency checks. The authors manually verified the technical claims, mathematical derivations, citations, code, and experimental results. No experimental data, attack metrics, benchmark scores, or security definitions were generated or accepted without author inspection.

N Potential Risk

The weight-extraction attack presented in this work operates solely on publicly released protected representations and requires no privileged access, training data, or secret keys. While publishing an efficient attack against an existing protection scheme could in principle lower the barrier for adversarial misuse of open-weight models, we argue that the net effect is beneficial. The vulnerability demonstrated here already exists in any deployed instance of TaylorUnswift-style protection; our contribution is to make it explicit rather than to introduce a new threat. Responsible disclosure of such weaknesses is standard practice in the security community, and concealing known breaks would leave developers and users with a false sense of security. Furthermore, the game-based formalism introduced in §4 is designed to support the construction of provably stronger schemes, and the attack methodology itself serves as a concrete adversarial evaluation tool for future proposals. We therefore believe this work advances the state of LLM weight protection rather than undermining it.