SPARSEMARK: ROUTING-AWARE DIFFERENTIAL PARAMETER WATERMARKING FOR MIXTURE-OF-EXPERTS LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Parameter watermarking for sparse mixture-of-experts (MoE) language models faces a placement challenge: evidence concentrated in a few experts is vulnerable to localized damage, while embedding must preserve model utility. We introduce SparseMark, a post-training multi-bit watermark that distributes evidence across experts and limits the influence of individual parameter pairs during recovery. SparseMark combines balanced expert allocation with response-based carrier selection, encodes a secret codeword through antisymmetric parameter updates, and spreads each bit across experts using keyed block ordering. Verification recovers the codeword from stored baseline contrasts through unit sign voting, without model forward passes. Controlled experiments show that balanced allocation improves recovery under targeted expert damage at comparable utility cost, while unit voting outperforms mean aggregation under pruning at fixed embedding energy. SparseMark achieves exact clean recovery of 256-bit codewords across three MoE backbones with small measured changes in perplexity. Detailed OLMoE evaluations demonstrate watermark retention under the evaluated pruning, quantization, and fine-tuning settings, while identifying limited recovery margin under feedback-guided attacks after quantization. These findings support distributed carrier placement and bounded pairwise influence as design principles for parameter watermarking in sparse MoE models.

1 INTRODUCTION

White-box watermarking embeds secret-key-dependent signals into model parameters to support ownership verification after transformations such as fine-tuning, quantization, or pruning. Existing parameter-space methods are largely designed for dense models, where eligible weights can be treated as a comparatively uniform embedding substrate. Sparse MoE models break this assumption: only a subset of experts is selected for each token, so expert parameters differ substantially in routing frequency and local response (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022; Du et al., 2022; Zoph et al., 2022). Parameter watermarking for MoEs therefore introduces an architectural placement problem: which expert parameters should carry the watermark, and how should they be organized into a stable keyed representation despite this routing heterogeneity?

To address this problem, we introduce **SparseMark**, a routing-aware, post-training, multi-bit parameter watermarking scheme for sparse MoE models. Each registration embeds a distinct multi-bit signature linked to an owner or licensed release; disjoint carrier allocation lets multiple registrations coexist in the same model and remain independently verifiable. SparseMark leverages calibration-time routing statistics to select high-exposure expert parameters as carriers, embeds a secret codeword through keyed antisymmetric updates without gradient optimization, and recovers each bit by majority voting over pairwise hard decisions. Experiments on OLMoE-1B-7B, Qwen1.5-MoE-A2.7B, and GPT-OSS-20B confirm exact recovery of a 256-bit codeword, robust retention under fine-tuning, pruning, and quantization, and rejection of all evaluated unregistered candidates.

This paper makes three contributions:

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.

1. We develop a routing-aware watermarking scheme for sparse MoE models that combines balanced carrier allocation, keyed block spreading, and antisymmetric embedding to distribute a multi-bit signature across experts without gradient optimization.
2. We introduce a parameter-space verifier based on unit sign voting, preventing large pair contrasts from dominating recovery. Acceptance combines a registered recovery threshold with two complementary tests of codeword and spreading-key specificity.
3. We demonstrate exact recovery of 256-bit codewords on clean OLMoE, Qwen1.5-MoE, and GPT-OSS-20B models. Controlled OLMoE experiments show that routing-aware balanced allocation and block spreading limit damage from public-score targeting, while unit voting improves recovery under pruning. All evaluated unregistered candidates are rejected.

2 THREAT MODEL

Adversary The adversary obtains a released watermarked MoE model M_k with full white-box access to its architecture and parameters, and knows the watermarking algorithm and verification protocol, both public by assumption. The exact enrollment calibration sample is private and unavailable to the adversary. The adversary may use publicly available proxy corpora to estimate routing frequencies and response scores and guide targeted perturbations, but has no access to the secret key or private enrollment assets and receives no verifier feedback. Its goal is to evade verification at minimal utility cost, using any common post-training transformation: fine-tuning, quantization, pruning, numerical perturbation, or routing modification. We evaluate attack effectiveness jointly through watermark extraction rate, the registered ownership decision, and utility changes measured by teacher-forced PPL and downstream accuracy. This comparison characterizes the utility cost associated with reducing watermark recovery or causing rejection.

Defender The defender enrolls the watermark using the pre-watermark model M_0 and retains the secret key k and private enrollment assets. All verification statistics, null distributions, thresholds, and test levels are fixed at enrollment, before candidate evaluation. Given a candidate model M, the defender applies the parameter-space verifier to determine whether the enrolled watermark remains detectable.

3 METHOD

As illustrated in Fig. 1, SparseMark proceeds in three steps: routing-aware carrier selection, keyed pairwise embedding, and parameter-space recovery. Enrollment distributes the pair budget across eligible experts in all MoE layers and pairs each carrier with an unused reference in the same expert's down-projection. Throughout, each expert is identified by its layer and local expert index.

3.1 EMBEDDING DESIGN

The secret key material is denoted by $k = (K_\pi, K_c, K_s)$, where K_π controls reference pairing and block ordering under separate domains for pairing and block ordering, K_s generates the spreading signs, and $K_c \in \{-1, +1\}^B$ is a secret codeword vector of B payload bits, sampled independently of both keys and retained for verification.

A carrier is a scalar parameter coordinate used to embed the watermark. SparseMark places carriers according to two complementary principles. Across experts, it distributes the pair budget approximately uniformly over all eligible experts, so no small subset of experts holds a decisive share of the watermark evidence, and localized expert removal reaches only a small fraction of carriers. Within each expert, it selects carriers from the rows with the largest gate-weighted response measured during calibration. This criterion places carriers in rows with high measured gate-weighted response, which low-response pruning preferentially retains. SparseMark identifies eligible experts from their calibration routing counts and distributes a total budget of N carrier–reference pairs approximately evenly across these experts in all MoE layers. Each expert receives the same base allocation; any remaining pairs are assigned one per expert in descending order of routing frequency. This allocation limits carrier concentration while giving frequently routed experts priority for the remaining budget.

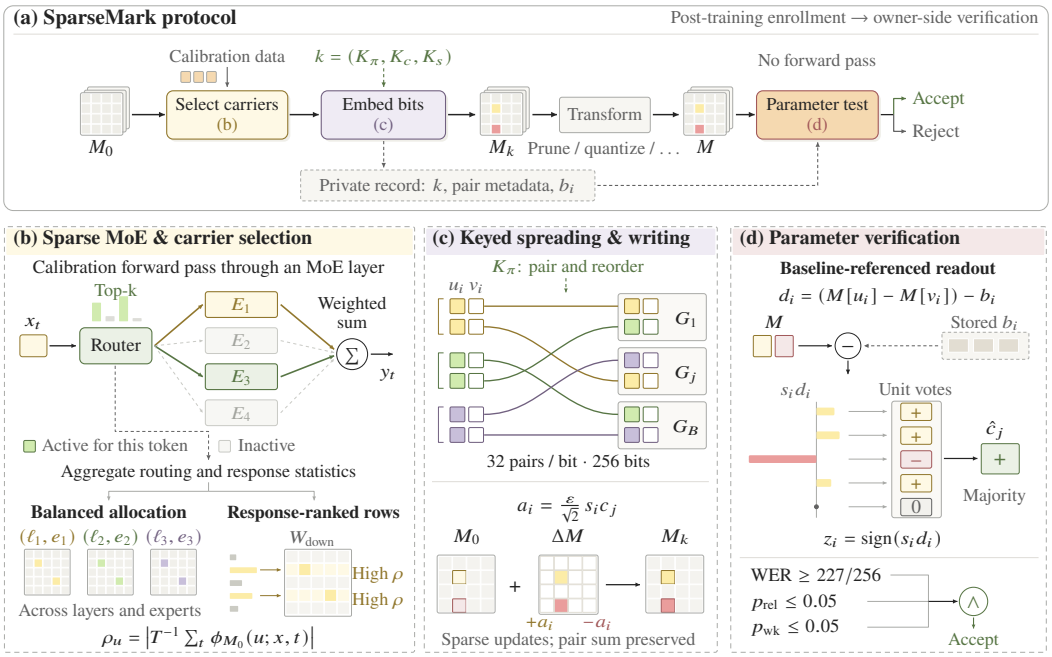

Figure 1: **SparseMark overview. (a)** End-to-end protocol. Enrollment writes a codeword into M_0 without gradient optimization and retains private verification assets. The released M_k may undergo transformations such as fine-tuning, pruning, or quantization before owner-side verification. Dashed arrows carry private assets. **(b)** A calibration token is routed to a sparse subset of MoE experts and their outputs are combined by a weighted sum. Statistics aggregated over calibration tokens guide balanced allocation across eligible experts (ℓ, e) and selection of down-projection rows with high gate-weighted response ρ_u. **(c)** K_π pairs carriers (filled cells) with unused references (open cells) within experts and mixes pairs into blocks across experts and layers. Colors track expert provenance. Code symbols c_j and spreading signs s_i control equal and opposite writes that preserve each pair sum. The sparse increment ΔM changes only the selected carrier–reference coordinates; blank cells in ΔM denote zero updates. The default uses 256 bits and 8192 pairs, or 32 pairs per bit. **(d)** Given a candidate in the registered coordinate order, stored contrasts $b_i = M_0[u_i] - M_0[v_i]$ yield d_i. Despreading and unit sign voting bound every pair's influence; block majorities recover the codeword. Zero votes abstain and tied blocks are unrecovered. Acceptance requires the recovery threshold and both preregistered specificity tests to pass. Verification needs no candidate forward pass.

Within each expert, SparseMark selects carrier coordinates from the down-projection in descending order of their corresponding rows' response scores ρ_u. To compute ρ_u, we multiply each row's output in M_0 by the router weight assigned to its expert, average over all calibration tokens with unrouted tokens contributing zero, and take the absolute value.

For pair $i \in \{1, \ldots, N\}$, let u_i denote the selected carrier coordinate and v_i its reference coordinate. The pairing key K_π selects v_i from the unused coordinates in the same expert's down-projection parameters, subject to $v_i \neq u_i$.A separate keyed ordering assigns the resulting pairs to codeword blocks.

Let B denote the payload length, i.e., the number of codeword bits. Each payload bit is spread across a block of parameter pairs: the codeword bit determines the common block direction, while independent spreading signs distribute that bit across individual pairs. After pairing, we order all pairs by HMAC-SHA256 ranks under K_π, using a separate block-order domain. Consecutive groups of $K = N/B$ pairs form the B blocks. Let G_j denote the pair indices assigned to block j. For each block $j \in \{1, \ldots, B\}$, the target block symbol is the j-th bit of the registered codeword, denoted by $c_j = K_c[j] \in \{-1, +1\}$. The key K_s generates the spreading signs $s_i \in \{-1, +1\}$. Under the PRF model, these outputs are treated as independent uniform signs. The secret signs s_i mask this shared direction through the products $s_i c_j$, while verification removes the mask by multiplying each measured contrast by s_i. Keyed ordering mixes pairs across experts and layers before block

partitioning, reducing the risk that a localized perturbation affects many votes for the same bit. It also makes the block assignment depend on the secret key rather than the public coordinate order.

Let $M[u]$ denote the scalar parameter at coordinate u, and let ε denote the L_2 magnitude of the antisymmetric update applied to an embedding pair. For each pair $i \in G_j$, SparseMark writes

$$M_k[u_i] = M_0[u_i] + \frac{\varepsilon}{\sqrt{2}} s_i c_j \tag{1}$$

$$M_k[v_i] = M_0[v_i] - \frac{\varepsilon}{\sqrt{2}} s_i c_j \tag{2}$$

For a fixed pairwise L_2 budget ε, equal and opposite updates maximize the change in pair contrast while preserving the pair sum. The resulting contrast shift is $\sqrt{2}\varepsilon s_i c_j$, providing a direct signal for parameter-space verification. At the same budget, pairing improves mean INT4 WER from 90.7% to 92.3% on OLMoE at $\varepsilon = 10^{-3}$, owing to lower quantization noise at the reference coordinates (Appendix A.4).

3.2 Parameter Verification

The verifier reconstructs the pairing π from K_π and loads the registered codeword K_c and stored contrasts b_i. Given a candidate model M, the verifier extracts a candidate codeword $\hat{c} = (\hat{c}_1, \ldots, \hat{c}_B)^\top \in \{-1, 0, +1\}^B$ and evaluates its agreement with the registered codeword K_c using the recovery threshold and key-specificity tests defined below. At enrollment, we store the original contrast of each registered pair,

$$b_i = M_0[u_i] - M_0[v_i] \tag{3}$$

Given a candidate model M in the registered coordinate order, the verifier computes

$$d_i = \big(M[u_i] - M[v_i]\big) - b_i \tag{4}$$

Thus, d_i is the change in pair contrast between the candidate model M and the enrolled model M_0. Verification therefore relies only on 64 KiB of baseline contrasts, in addition to keys and reconstruction metadata, requiring no forward pass through the candidate.

For each candidate model, the expected direction of d_i is determined by the spreading sign and the codeword symbol assigned to the pair. Each pair first makes a local hard decision, giving the pair-level vote

$$z_i = \operatorname{sign}(s_i d_i) \in \{-1, 0, +1\}, \tag{5}$$

we use $\operatorname{sign}(0) = 0$, so a pair abstains when $d_i = 0$. The votes aggregate into the block-level margin

$$V_j = \frac{1}{|G_j|} \sum_{i \in G_j} z_i \tag{6}$$

We decode each block by majority vote,

$$\hat{c}_j = \begin{cases} +1, & V_j > 0, \\ -1, & V_j < 0, \\ 0, & V_j = 0. \end{cases} \tag{7}$$

Every non-abstaining pair contributes one unit vote regardless of contrast magnitude. This bounds each pair's contribution to the block score, preventing a large contrast from outweighing multiple opposing votes. Section 4.5 evaluates this design against mean aggregation under pruning. Conditional on enrollment, we measure alignment with the registered codeword using $T_{\mathrm{Param}} = B^{-1} \sum_{j=1}^{B} c_j V_j$. Larger values indicate stronger agreement between the block voting margins and the expected codeword.

The codeword relabeling test holds the block margins V fixed and evaluates permutations of the registered codeword K_c. It tests whether the observed responses align unusually well with the specific assignment of bits in K_c to carrier blocks.

The wrong-code key test holds the carrier coordinates and pair contrasts fixed, replacing K_c with independent null codewords $K_c' \sim \text{Uniform}(\{-1,+1\}^B)$ alongside independently redrawn spreading keys K_s'. It tests whether the observed alignment is specific to the enrolled pair (K_c, K_s).

We require both tests to pass because they assess complementary aspects of specificity: the assignment of code bits to block responses and the joint choice of codeword and spreading signs. For each test, we recompute T_{Param} under the corresponding randomization and calculate p_{rel} or p_{wk} as the fraction of null statistics at least as large as the observed value, with a plus-one correction (Appendix D.3). Both must be at most 0.05.

Watermark extraction rate is the bit-recovery rate $WER = B^{-1}\sum_{j=1}^{B}\mathbf{1}[\hat{c}_j = c_j]$, counting recovered blocks out of $B = 256$.

The parameter-level detection criterion is given by

$$D = \mathbf{1}\big[WER \geq 227/256\big] \cdot \mathbf{1}\big[p_{\text{rel}} \leq 0.05\big] \cdot \mathbf{1}\big[p_{\text{wk}} \leq 0.05\big] \tag{8}$$

3.3 Wrong-key bound under code independence

The recovery gate $\text{WER} \geq 227/256$ controls false acceptance under the independent wrong-code null at a target probability of at most 2^{-128}. Formally, condition on the candidate model, the pairing and spreading keys, and all retained verification assets. This conditioning fixes the extracted candidate bits $\hat{\mathbf{c}} = (\hat{c}_1, \ldots, \hat{c}_B)$. Under the independent wrong-code null, the alternative codeword $\mathbf{c}' = (c_1', \ldots, c_B')$ consists of mutually independent uniform signs drawn independently of this conditioning.

$$t^\star = \min\{t \in \mathbb{N} : \Pr[\text{Bin}(256, 1/2) \geq t] \leq 2^{-128}\} = 227, \tag{9}$$

The resulting tail probability is approximately $2^{-128.99}$. Since acceptance also requires passing the recovery gate, the full decision $\mathcal{D}$ inherits the same bound under the independent wrong-code null (Appendix D.3).

4 Experiments

This section evaluates SparseMark along four dimensions, organized around the following research questions:

RQ1 (Effectiveness): Can SparseMark reliably embed and recover a multi-bit ownership codeword across MoE backbones, while rejecting unregistered candidates, compared with existing parameter watermarking methods?

RQ2 (Fidelity & Stealthiness): Does routing-aware embedding preserve language-modeling and downstream utility, and do the parameter-level modifications remain undetectable?

RQ3 (Robustness): Does the watermark survive fine-tuning, pruning, quantization, parameter noise, and removal attacks that target routing-exposed experts?

RQ4 (Design Choices): Which components of SparseMark, namely carrier selection, structural dispersion, embedding strength, and block size, account for its effectiveness and robustness?

4.1 Experimental Setup

Models and configuration Our primary model is OLMoE-1B-7B-0125 (Muennighoff et al., 2024). Cross-model enrollments use Qwen1.5 Sparse MoE and GPT-OSS-20B. Unless stated otherwise, each enrollment uses $B = 256$ codeword bits and $N = 8192$ disjoint embedding pairs, so each block contains $N/B = 32$ pairs; we denote this default configuration by B256/N8192. The embedding strength is $\varepsilon = 1 \times 10^{-3}$. Section 4.5 reports the corresponding strength and block-size scans. References are selected by a keyed permutation within each expert's down-projection (Section 3.1), and codeword bits are independent per block. For OLMoE, enrollment uses 128 WikiText-2 raw test texts totaling 21,980 tokens, tokenized with special tokens and without truncation; these data are used to estimate expert routing frequencies and local responses.

Baselines We compare with ELLMark (Yuan et al., 2025), GaussMark (Block et al., 2025), NeuralMark (Yao et al., 2026), and PathMark (Gao et al., 2026). Baselines are enrolled and evaluated on OLMoE under shared transformations where their protocols define them; extraction follows each method's original protocol using its native hyperparameters, with implementation details in Appendix E and numeric hyperparameters in the released configuration manifest.

Metrics We use teacher-forced perplexity (PPL) on a held-out WikiText-2 evaluation set as the primary language-model utility measure, complemented by downstream accuracy benchmarks (Section 4.3). Watermark extraction rate (WER) measures recovery under each method's native verification protocol. For SparseMark, ELLMark and NeuralMark, it is the fraction of payload bits recovered; for PathMark and GaussMark, it is the fraction of evaluated watermark instances successfully detected. False accept rate (FAR) is the fraction of unregistered candidates incorrectly accepted.

Statistical units and reporting convention. We evaluate ten independently keyed enrollment records per condition and report means and standard deviations across records. Appendix E.1 specifies the statistical units and reporting conventions.

4.2 RQ1: Watermark Effectiveness

RQ1 asks whether SparseMark reliably embeds and recovers ownership information in the absence of any attack.

Clean recovery Across all ten B256/N8192 registrations, clean watermarked models achieve 100% WER and pass both specificity tests ($p_{\mathrm{rel}} = 1/1{,}000$ and $p_{\mathrm{wk}} = 1/10{,}001$), yielding 100% ownership acceptances.

Table 1: Cross-method, cross-model clean recovery without attack. ΔPPL denotes the paired difference relative to the unwatermarked baseline.

	OLMoE		Qwen1.5-MoE		GPT-OSS-20B	
Method	Δ PPL	WER (%)	Δ PPL	WER (%)	Δ PPL	WER (%)
ELLMark	-0.0001 ± 0.0044	88.28	0.0066 ± 0.0043	100.00	0.0649 ± 0.0315	69.69
GaussMark	0.0001 ± 0.0043	6.88	-0.0003 ± 0.0013	7.00	-0.0009 ± 0.0019	4.00
NeuralMark	7699.8950 ± 2312.5838	98.91	2647.5789 ± 723.7230	99.73	-0.9568 ± 0.3864	95.59
PathMark	-0.0357 ± 0.0452	83.20	0.1848 ± 0.2247	100.00	27.8247 ± 3.9794	100.00
SparseMark	-0.0004 ± 0.0048	100.00	0.0093 ± 0.0109	100.00	-0.0266 ± 0.0348	100.00

NeuralMark exceeds the utility threshold on both OLMoE and Qwen1.5-MoE across the tested LoRA strengths, $\alpha \in \{1, 2, 4, 8, 16, 32\}$, including its default setting of $\alpha = 32$.

Cross-method clean comparison SparseMark achieves exact clean recovery while preserving utility on the main model, as shown in Table 1. This advantage is maintained across the additional models, confirming that SparseMark is the only evaluated method that simultaneously provides exact clean recovery and clean-utility preservation throughout the benchmark.

Specificity Across all three models and all evaluated transformations (clean, noise, 10% pruning, and INT4 quantization), the verifier accepts none of the 800 independently keyed unregistered candidates per condition (false acceptance rate: 0%; 95% exact binomial CI: $[0, 0.46\%]$ for each condition). The verifier also accepts all 16 coexisting registrations per seed and rejects all $2,400$ tested registration mismatches.

Coexisting registrations SparseMark supports 16 coexisting registrations in the evaluated setting, totaling $4,096$ embedded bits per model: all 160 registrations across ten seeds recover exactly before quantization and remain accepted after the shared INT4 transformation.

Calibration transfer Replacing WikiText-2 enrollment data with FineWeb-Edu preserves 100.00% WER under clean and pruning conditions and achieves $91.88 \pm 2.04\%$ under per-output-channel INT4 fake quantization, with all registrations accepted in each condition.

4.3 RQ2: FIDELITY

SparseMark preserves teacher-forced language-model utility at the default embedding strength. Across ten independently keyed OLMoE enrollments at $\varepsilon = 10^{-3}$, the paired WikiText-2 PPL change is -0.0004 ± 0.0048. Downstream benchmark evaluations exhibit the same pattern of preservation: accuracy changes by $+0.01\% \pm 0.12\%$ on the full MMLU test set (14,042 questions), by $-0.07\% \pm 0.72\%$ on the ARC-Challenge validation set (299 questions), and by $+0.66\% \pm 0.45\%$ on GSM8K (1,319 questions; 42.77% to 43.42%). (Appendix E.1).

Two learned parameter-space detectors perform near chance on held-out watermark keyson held-out watermark keys (AUC 0.492 and 0.503; Appendix C.5).

4.4 RQ3: ROBUSTNESS

Every attack in this subsection targets the released watermarked checkpoint under the key-unaware adversary of Section 2; the adaptive attack additionally grants verifier feedback.

Shared transformations across methods Table 2 compares all methods on OLMoE under the four transformations every baseline protocol defines: shape-preserving pruning, INT8 and INT4 quantization, and Gaussian parameter noise. All four are weight-only, applied to the released checkpoint; INT8 and INT4 use symmetric per-output- channel fake quantization with dequantized write-back.SparseMark recovers its full codeword under pruning, INT8, and noise. INT4 is the most aggressive of the four yet still yields 92.30% WER and all seeds clear the registered detection gate.

Table 2: Cross-method recovery on OLMoE under shared transformations.

Method	Prune (%)	INT8 (%)	INT4 (%)	Noise (%)
ELLMark	85.43	58.20	54.30	62.77
GaussMark	4.69	4.38	3.75	4.69
NeuralMark	97.66	100.00	98.91	98.91
PathMark	83.20	81.80	79.80	81.80
SparseMark	100.00	100.00	92.30	100.00

Calibrated packed INT4 Under a separate calibrated group-wise INT4 protocol, SparseMark attains $99.61 \pm 0.37\%$ WER after integer packing and reloading with FP16 scales; all ten watermarked records are accepted and all unwatermarked controls are rejected (Appendix C.1).

Extended transformations Over 1,000 steps of full-model fine-tuning at a learning rate of 10^{-5}, SparseMark retains 100.00% WER and 100% acceptance, while teacher-forced PPL improves from 10.3006 to 8.6863. Mean WER falls below the 88.67% gate only at 10^{-3}, where the model itself collapses (PPL 2309.9). Under merged LoRA, and under the weight-space transformations of weight fusion, route editing and million-scalar random overwrite, SparseMark keeps $\geq 99.92\%$ WER. Full results across learning rates and merged LoRA are in Table 14, and the remaining transformations are in Table 13.

Utility attribution Under INT4 quantization and 10% pruning, the watermark–transformation interaction in teacher-forced PPL is -0.0010 ± 0.0078 and $+0.0014 \pm 0.0101$, respectively, compared with transformation-only increases of $+3.1049$ and $+2.7980$. These near-zero interactions indicate that watermarking has little effect on the PPL increase induced by either transformation.

Extended adaptive attacks and routing interventions To stress-test SparseMark beyond the baseline threat model, we additionally evaluate an extended 200-query search with feedback from the scalar verifier (Appendix C.4). From the clean initialization, final WER remains 100.00%, with all registrations accepted. From per-output-channel fake int4 quantization, mean WER decreases from 92.30% to 91.68%, and one registration moves from 227 correct blocks, exactly at the recovery threshold, to 224.

Routing interventions preserve at least 99.92% WER. Router noise, route unlearning, and 50% expert removal raise teacher-forced PPL to 263.80, 22.70, and 77.97, respectively.

Table 3: Robustness and utility trade-off under feedback-guided and routing-targeted attacks on OLMoE.

Attack	WER (%)	ΔPPL
Adaptive (Clean init)	100.00 ± 0.00	$+0.0054$
Adaptive (INT4 init)	91.68 ± 2.40	$+3.1031$
Router logit noise (σ=1.0)	100.00 ± 0.00	$+253.5028$
Route unlearning (epoch 10)	100.00 ± 0.00	$+12.4105$
Expert removal 10%	100.00 ± 0.00	$+1.9099$
Expert removal 50%	99.92 ± 0.16	$+67.6759$

Targeted attacks using public statistics SparseMark maintains high watermark recovery under both targeted attacks. Pair decorrelation preserves 100% WER at $L_2 \in \{0.02, 0.05\}$, $99.84 \pm 0.19\%$ at $L_2 = 0.10$, and $90.82 \pm 1.45\%$ at $L_2 = 0.20$, with $\Delta\text{PPL} = +1.1724$ at the largest budget. Noise targeted by routing frequency preserves 100% WER across all tested budgets, with $\Delta\text{PPL} \leq +0.0026$ (Table 16).

4.5 RQ4: Which design choices matter

The following analyses isolate four design choices in SparseMark.

Carrier selection SparseMark improves recovery over global random carrier selection under low-response pruning. When the lowest-response 50% and 70% of rows are pruned, SparseMark retains $100.00 \pm 0.00\%$ and $96.88 \pm 0.55\%$ WER, respectively, compared with $90.10 \pm 1.61\%$ and $62.11 \pm 2.30\%$ for global random selection. The corresponding ΔPPL values are $+29.3$ and $+208.8$. These results support the combined placement strategy of balanced expert allocation and response-based carrier selection.

Structural dispersion Balanced expert allocation and keyed block ordering limit the damage within each block, preserving correct majorities under the evaluated public-score attack. The public-score attack evaluates robustness against key-unaware adversaries: the attacker aims to erase watermark evidence by zeroing a limited number of expert down-projections selected using public routing/response scores. The attack budget counts targeted experts, and paired ΔPPL measures the resulting utility degradation. At a fixed budget of 32 experts targeted by the same public ρ_u ranking, the attack hits 86.82% of carrier pairs under global response-score selection (ρ_u), yielding 71.56% WER and rejection of all ten registrations (ΔPPL = 6.33). Under the identical attack, SparseMark has only 3.12% of its pairs targeted. Each block retains at least 26 correct votes, preserving 100% WER and acceptance of all ten registrations ($\Delta PPL = 6.34$). The sharply different recovery outcomes at nearly identical utility cost support the benefit of distributed carrier allocation (Table 6).

Keyed block ordering spreads the allocated pairs across blocks. Across the ten enrollment records, each 32-pair block spans 31.6 distinct experts and 14.0 MoE layers on average. In these records, no expert contributes more than three pairs to a block, and no layer contributes more than nine. Starting from correct pair votes, damage confined to a single expert or a single layer therefore leaves a correct majority even if every affected vote flips. This dispersion limits the influence of localized parameter damage under unit voting.

Block voting Sign voting achieves substantially higher pruning recovery than mean aggregation at the same embedding-energy budget. With $B = 256$ fixed and $\varepsilon_K = 10^{-3}\sqrt{32/K}$, increasing K from 1 to 32 raises sign-vote WER from 94.57% to 100.00%, while mean-aggregation WER falls to 75.51% (Table 10).

Under INT4, an empirical majority-vote model approximates within-block votes as i.i.d., using the measured probabilities of correct, incorrect, and zero votes. It predicts 92.33% WER at the default strength, compared with 92.30% observed, with a mean absolute prediction error of 0.12 percentage points across five embedding strengths.

Operating point The INT4 sweep identifies a low-distortion operating point and shows how stronger perturbations provide additional recovery margin. INT4 recovery increases monotonically from $60.94\% \pm 4.57\%$ at $\varepsilon = 2.5 \times 10^{-4}$ to $74.65\% \pm 3.47\%$ at 5×10^{-4}, $92.30\% \pm 2.23\%$ at 10^{-3}, $99.88\% \pm 0.19\%$ at 2×10^{-3}, and $100.00\% \pm 0.00\%$ at 4×10^{-3}, while clean ΔPPL remains within $+0.011$ throughout the scan. We use $\varepsilon = 10^{-3}$ as the default because it is the smallest tested strength that exceeds the WER threshold, thereby minimizing the parameter intervention while retaining sufficient INT4 robustness.

We use a payload of $B = 256$ bits, following prior white-box watermarking evaluations (Wang & Kerschbaum, 2021; Li et al., 2024), and choose $K = 32$ carrier pairs per bit ($N = 8192$).

At $\varepsilon = 10^{-3}$, this configuration achieves 92.30% sign-vote recovery under INT4 and is the only tested block size at fixed strength to achieve formal acceptance across all records. Appendix B.2 provides the supporting block-size and payload-length scans.

5 Related Work

Weight-based methods encode ownership evidence directly in model parameters (Uchida et al., 2017; Rouhani et al., 2019; Fan et al., 2019; Zhang et al., 2026; Block et al., 2025). Existing approaches use learned embedding losses, secret parameter selection, or aggregation to improve recovery and resist attacks, but their carrier choices do not account for the uneven use of experts in sparse MoE models. SparseMark addresses sparse MoE architectures via an optimization-free, post-training formulation: it uses expert routing statistics solely to guide offline carrier selection, embedding the watermark through keyed antisymmetric differential contrasts. Keyed spreading and unit sign voting distribute each bit across pairs and limit the influence of any one pair.

In contrast, activation-space watermarking schemes embed evidence within latent representations. For instance, Bellafqira & Coatrieux (2025) align internal feature distributions to encode a signature, which requires gradient-based optimization during embedding and forward-pass execution with intermediate feature hooking during verification. Black-box verification schemes (Adi et al., 2018; Zhang et al., 2018) instead construct evidence at the prediction boundary through specially crafted trigger sets, requiring only input–output query access for verification but depending on surrogate fine-tuning or adversarial sample synthesis to plant the trigger behavior.

Finally, emerging MoE-specific techniques derive evidence from dynamic routing paths, expert utilization profiles, or execution traces (Gao et al., 2026; He et al., 2025; Lv et al., 2026; Sun et al., 2026). These methods tie the watermark to the dynamic behavior of the router under specific query distributions. SparseMark shifts the verification surface entirely from runtime behavioral traces to static parameter geometry: the ownership decision is deterministically evaluated on the parameter tensor itself, providing a registered ownership test without instantiating the model or reproducing conditioned execution paths.

6 Conclusion

SparseMark provides a practical watermark for sparse MoE models that encodes multiple bits in model parameters. Routing-aware balanced allocation disperses the carriers, antisymmetric paired embedding improves recovery under per-output-channel INT4 fake quantization at a fixed embedding budget, and unit sign voting gives each pair equal influence in parameter-space verification. Clean recovery is exact on OLMoE, Qwen1.5-MoE, and GPT-OSS-20B. Within the stated threat model, SparseMark gives owners of sparse MoE models a concrete ownership test for released sparse MoE checkpoints.

AI use statement

Large language models (LLMs) were employed exclusively to improve the quality of written English, with attentive supervision by the authors. No experimental design, data analysis, or any intellectual contribution were generated through LLMs.

Ethics statement

SparseMark supports model provenance verification through parameter-space watermarking. Its responsible use requires protecting secret registration assets and interpreting verification results within the stated threat model and statistical assumptions. A positive verification result provides evidence of a registered watermark; determining legal ownership additionally requires the relevant licensing and provenance records.

Reproducibility statement

Section 3 specifies the watermark embedding procedure and verification rule. Section 4 and appendices document the evaluated models, calibration and evaluation data, embedding settings, transformation protocols, and statistical analyses. Additional implementation details and results across independently keyed registrations support reproduction of the reported findings.

References

Yossi Adi, Carsten Baum, Moustapha Cissé, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *Proceedings of the 27th USENIX Security Symposium*, pp. 1615–1631. USENIX Association, 2018. URL https://www.usenix.org/conference/usenixsecurity18/presentation/adi.

Reda Bellafqira and Gouenou Coatrieux. DICTION: Dynamic robust white box watermarking scheme for deep neural networks. *Applied Sciences*, 15(13):7511, 2025. doi: 10.3390/app15137511. URL https://doi.org/10.3390/app15137511.

Adam Block, Alexander Rakhlin, and Ayush Sekhari. GaussMark: A practical approach for structural watermarking of language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 4507–4569. PMLR, 2025. URL https://proceedings.mlr.press/v267/block25a.html.

Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten P. Bosma, Zongwei Zhou, Tao Wang, Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. GLaM: Efficient scaling of language models with mixture-of-experts. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 5547–5569. PMLR, 2022. URL https://proceedings.mlr.press/v162/du22c.html.

Lixin Fan, Kam Woh Ng, and Chee Seng Chan. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, pp. 4714–4723. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/hash/75455e062929d32a333868084286bb68-Abstract.html.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. URL https://jmlr.org/papers/v23/21-0998.html.

Yudong Gao, Qingyue Wang, Yuanyuan Yuan, Ruixuan Huang, Linghan Chen, Zimo Ji, and Shuai Wang. PathMark: Protecting intellectual property of mixture-of-expert LLMs via path watermarks, 2026. URL https://arxiv.org/abs/2607.03688.

Xin He, Junxi Shen, Zhenheng Tang, Xiaowen Chu, Bo Li, Ivor W. Tsang, and Yew-Soon Ong. RouteMark: A fingerprint for intellectual property attribution in routing-based model merging, 2025. URL https://arxiv.org/abs/2508.01784.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. In *The Ninth International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=qrwe7XHTmYb.

Fangqi Li, Haodong Zhao, Wei Du, and Shilin Wang. Revisiting the information capacity of neural network watermarks: Upper bound estimation and beyond. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(19):21331–21339, 2024. doi: 10.1609/aaai.v38i19.30128.

Bo Lv, Zhiheng Xu, KeDong Xiu, Ruyi Ding, Tianhang Zheng, Zhibo Wang, and Kui Ren. RouteScan: A non-intrusive approach to auditing MoE LLMs safety via expert routing telemetry, 2026. URL https://arxiv.org/abs/2605.24817.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. OLMoE: Open mixture-of-experts language models, 2024. URL https://arxiv.org/abs/2409.02060.

Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. 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*, pp. 485–497. ACM, 2019. doi: 10.1145/3297858.3304051. URL https://doi.org/10.1145/3297858.3304051.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *The Fifth International Conference on Learning Representations (ICLR)*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Zewen Sun, Qian Jiang, Siyuan Sheng, and Liyao Xiang. WaterMoE: Expert-routing-based watermarking for high fidelity and efficiency, 2026. URL https://arxiv.org/abs/2607.13099. arXiv preprint arXiv:2607.13099.

Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, pp. 269–277. ACM, 2017. doi: 10.1145/3078971.3078974. URL https://doi.org/10.1145/3078971.3078974.

Tianhao Wang and Florian Kerschbaum. Riga: Covert and robust white-box watermarking of deep neural networks. In *Proceedings of The Web Conference 2021*, pp. 993–1004. ACM, 2021. doi: 10.1145/3442381.3450000.

Yuan Yao, Jin Song, and Jian Jin. Hashed watermark as a filter: Defeating forging and overwriting attacks in weight-based neural network watermarking. arXiv preprint arXiv:2507.11137, 2026. URL https://arxiv.org/abs/2507.11137. Accepted to AAAI 2026.

Shuguang Yuan, Xingyu Su, Peizhuo Lv, Weiji Xue, Jing Yu, Xiaojie Zhu, and Chi Chen. An efficient white-box LLM watermarking for IP protection on online market platforms. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pp. 3680–3691, 2025. doi: 10.1145/3711896.3736841. URL https://doi.org/10.1145/3711896.3736841.

Bingxue Zhang, Xiaofeng Xu, and Feida Zhu. LineageMark: Multi-user white-box watermarking for contribution tracing in model derivation chains, 2026. URL https://arxiv.org/abs/2606.17123.

Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph. Stoecklin, Heqing Huang, and Ian Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security (ASIACCS)*, pp. 159–172, 2018.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. ST-MoE: Designing stable and transferable sparse expert models, 2022. URL https://arxiv.org/abs/2202.08906.

This supplement is organized around the questions the evidence answers: the mechanism evidence behind the design choices (Appendix A), the operating configuration and calibration sensitivity (Appendix B), the extended robustness evaluation (Appendix C), the exact enrollment, registration, and verification protocol (Appendix D), the evaluation protocols and baselines (Appendix E), and the reproducibility assets (Appendix F).

A Evidence for the Design Choices

This appendix gathers the evidence behind SparseMark's mechanism choices: routing-aware balanced carrier allocation with keyed block spreading (Appendix A.1), unit sign voting (Appendix A.2), response-based selection within fixed expert quotas (Appendix A.3), and the pair-vote model that predicts recovery from embedding strength (Appendix A.4).

A.1 Balanced allocation and localized damage

The central evidence for balanced carrier allocation is the matched comparison in Table 4: under the same public ρ_u ranking, the same attack operation, and the same budget of 32 zeroed experts, global response-score selection and SparseMark incur nearly identical utility damage, yet the watermark survives only under balanced allocation.

Table 4: Matched public-score attack at a budget of 32 zeroed experts on OLMoE (ten independently keyed registrations per arm). Both arms face the same public ρ_u ranking and attack operation; utility damage is nearly identical while watermark outcomes differ sharply.

Carrier allocation	Pairs hit	WER (%)	Accepted rate (%)	Paired ΔPPL
Global response-score selection	86.82%	71.56 ± 2.46	0	$+6.33 \pm 0.02$
SparseMark	3.12%	$\mathbf{100.00 \pm 0.00}$	100	$+6.34 \pm 0.01$

Mechanism: allocation, spreading, and sign voting This contrast follows from routing-aware balanced allocation, keyed block ordering, and sign voting. Routing-aware balanced allocation distributes carrier pairs across $1{,}021$ experts, limiting the attack to 3.12% of SparseMark's pairs, compared with 86.82% under global response-score selection. Keyed block ordering spreads the affected pairs across blocks, with no block containing more than six affected pairs in any evaluated registration. Each 32-vote block therefore retains at least 26 correct votes. Since sign voting limits each pair's influence to one vote, every bit remains recoverable even if all six affected votes are wrong.

Structural dispersion and the conditional recovery bound. Table 5 quantifies how the keyed permutation distributes carrier pairs across OLMoE layers and experts ($B = 256$, $N = 8192$, $K = 32$), showing broad coverage and limited concentration within each block.

Table 5: Carrier dispersion across 32-pair blocks. Shares are percentages of carrier pairs within each block; means and maxima summarize the corresponding per-block statistics.

Dispersion metric	**Mean across blocks**	**Observed maximum**
Distinct experts per block	31.6	32
Distinct layers per block	14.0	—
Largest expert share per block	4.2%	9.4%
Largest layer share per block	15.1%	28.1%
Largest share from one local expert index across layers	8.5%	18.8%

This dispersion yields a conditional recovery guarantee for the evaluated registrations. If all K pair votes are initially correct and an attack changes at most m pairs while leaving the remaining votes unchanged, then even arbitrary changes to the affected votes give

$$c_j V_j \geq \frac{K - 2m}{K}. \tag{10}$$

Thus, every block remains correctly decoded when $m < K/2$. For $K = 32$, a tie requires at least 16 vote flips, and an incorrect bit requires at least 17. The observed maxima correspond to $m \leq 3$ for one expert, $m \leq 6$ for one local expert index across all layers, and $m \leq 9$ for one layer. Each intervention therefore preserves correct decoding under the stated conditions; the guarantee concerns the evaluated allocations.

These bounds explain resistance to localized parameter damage. Router-only edits that leave the registered expert parameters unchanged preserve the verifier readout directly. Together, these mechanisms account for the corresponding results in Tables 3 and 15.

Public-score zeroing protocol. We evaluate four carrier-selection arms on OLMoE under identical conditions: the same public candidate pool, utility evaluation bank, and embedding configuration. All arms use antisymmetric embedding, keyed reference selection within the same expert's down-projection, keyed block ordering, and the same unit-vote verifier; the arms compare carrier allocation rules under the same embedding and verification procedure (the sign-generation implementations differ and are specified in Appendix D.1). Pair and block assignments are regenerated for each arm.

An expert is identified by its layer and expert index. The public pool contains 1,021 eligible experts; three experts receiving fewer than 32 calibration tokens are excluded. For each expert, the public attack score is the arithmetic mean of the specified row score across all of its eligible down-projection candidate rows, using the routing frequency f, response score ρ_u, and conditional response r_{cond} of Eq. 17 (Appendix D.1), with both response scores retaining the router gate weight. This yields expert-level rankings by f, ρ_u, and r_{cond}, frozen in the public candidate pool; experts are ranked in descending score order, with ties resolved by canonical pool order.

At each budget, the attacker zeros the entire down-projection weight matrix of every selected expert, leaving all other projection weights and router parameters unchanged. The operation preserves tensor shapes and requires no secret keys, registered carrier coordinates, or private pre-watermark parameters; it is therefore fully executable by a public adversary. Candidate selection for the three global baseline arms ranks the public candidate pool globally by $f_{e(u)}$ (Frequency-only), ρ_u (Response-only), and r_u^{cond} (Conditional-response), each capped at 1,024 rows per expert, whereas SparseMark distributes the pair budget approximately evenly across eligible experts and ranks carriers within each expert by ρ_u (Appendix D.1). During evaluation, the key-unaware adversary targets Frequency-only, Response-only, and Conditional-response by zeroing down-projections ranked by their respective expert-level scores f, ρ_u, and r_{cond}. SparseMark is targeted by the exact same public ρ_u ranking used against Response-only selection, ensuring that SparseMark and Response-only face an identical adversary and damage budget.

Table 6: Public-score targeting by zeroing selected experts' down-projection matrices on OLMoE. Every global carrier-selection control is defeated at a low budget, whereas SparseMark retains full watermark extraction and acceptance at every tested budget.

Selection	Score	Experts	WER (%)	Accepted rate (%)	ΔPPL
Frequency-only	f	4	99.77 ± 0.20	100%	$+0.2682 \pm 0.0040$
Frequency-only	f	8	43.05 ± 3.24	0%	0.7858 ± 0.0042
Frequency-only	f	16	43.05 ± 3.24	0%	$+1.8203 \pm 0.0042$
Response-only	h_u	16	95.27 ± 1.33	100%	$+2.3358 \pm 0.0092$
Response-only	ρ_u	32	71.56 ± 2.46	0%	$+6.3335 \pm 0.0228$
Conditional-response	r_{cond}	32	90.27 ± 1.10	100%	$+5.1932 \pm 0.0089$
Conditional-response	r_{cond}	64	71.48 ± 3.06	0%	$+12.5223 \pm 0.0123$
SparseMark	f	8	$\mathbf{100.00 \pm 0.00}$	100%	$+0.7881 \pm 0.0058$
SparseMark	ρ_u	32	$\mathbf{100.00 \pm 0.00}$	100%	$+6.3377 \pm 0.0125$
SparseMark	r_{cond}	64	$\mathbf{100.00 \pm 0.00}$	100%	$+12.5277 \pm 0.0123$

Budget-sweep results. Table 6 reports results over ten independently keyed registrations per arm. Each global control fails abruptly once the attack budget reaches its concentrated carrier set. Frequency-only selection survives the zeroing of four experts (WER 99.77%) but collapses at eight: WER falls to 43.05% and all ten registrations are rejected, at a paired PPL cost of only 0.79; doubling

the budget to 16 yields no further reduction, confirming that the eight highest-frequency experts already contain the carriers on which this arm depends. Response-only selection is likewise accepted on all ten registrations at a budget of 16 but rejected on all ten at 32. Conditional-response selection raises the required budget but does not remove the vulnerability: it is rejected on all ten registrations at 64 experts, at a larger PPL cost of 12.52. SparseMark retains 100% WER and acceptance of all ten registrations at every tested budget. Together with the matched comparison of Table 4, these results support the benefit of balanced carrier allocation: every tested global carrier-selection rule concentrates the watermark within a small, publicly identifiable subset of experts, enabling a key-free adversary to remove it at low utility cost, whereas balanced allocation eliminates this failure mode within the evaluated budgets.

Coverage statistics. The coverage mechanism behind Table 4 is visible in per-block vote counts. A pair is counted as hit when either endpoint belongs to a targeted expert; zeroing a down-projection matrix sets both endpoints of every pair it hosts to zero, approximately half of the hit pairs vote incorrectly, and untouched pairs remain correct. At the shared budget of 32 experts, Response-only selection exposes 86.82% of its pairs and retains approximately 4.2 untouched correct votes per block, producing an average of 43.4 incorrectly decoded blocks and 29.4 ties; SparseMark exposes 3.12% and, as stated above, every block retains at least 26 correct votes. The same coverage-dependent pattern governs the other arms. Under Frequency-only selection, targeting four experts hits exactly half of its pairs, and only one block is decoded incorrectly (five tied) among the 2,560 evaluated blocks, while eight experts cover every pair and reduce WER to 43.05%. Under Conditional-response selection, pair coverage rises from 74.78% to 87.24% between budgets 32 and 64 as WER falls from 90.27% to 71.48%; SparseMark exposes only 6.25% of its pairs at the latter budget, with at most nine hit pairs in any block and at least 23 correct votes preserved.

All four arms assign pairs to blocks through keyed ordering followed by partitioning into groups of 32; none explicitly balances expert occupancy within individual blocks. Keyed ordering therefore does not by itself prevent concentrated carrier sets from being broadly affected by public-score targeting. Routing-aware balanced allocation across experts limits overall pair coverage, and the resulting block distribution preserves correct majorities under the evaluated interventions.

A.2 Sign voting

Mean aggregation decodes each block as

$$\hat{c}_j^{\text{mean}} = \text{sign}\left(|G_j|^{-1} \sum_{i \in G_j} s_i d_i \right). \tag{11}$$

Its WER is the percentage of blocks for which $\hat{c}_j^{\text{mean}} = c_j$. Mean aggregation averages signed contrasts before taking the sign; sign-vote decoding averages their signs (Section 3.2). Both use the same candidate model and registration.

Comparison evidence. Under the fixed-total-energy pruning sweep (Table 10), mean aggregation falls to 75.51% WER at $K = 32$ while sign voting retains 100.00%, a 24.49-point gap at equal embedding energy. Under targeted expert removal (Table 15), mean aggregation drops to 65.27% across the three selection rules at 10% removal, whereas sign voting remains at 100.00%. The two block-size scans of Appendix B.2 answer different questions and are not interchangeable: increasing K at fixed strength also increases total embedding energy, whereas increasing K at fixed total energy weakens each pair's perturbation (their $K = 1$ pruning results differ slightly for this reason).

Sign voting improves recovery under pruning and targeted expert removal by limiting each pair to one vote. At equal total embedding energy, it achieves 100.00% pruning WER at $K = 32$, compared with 75.51% for mean aggregation, a gain of 24.49 percentage points (Table 10). Mean aggregation achieves higher recovery under the evaluated pair-decorrelation perturbations: Table 16 reports 100.00% versus 99.84% WER at a relative perturbation budget of 0.10 and 97.42% versus 90.82% at 0.20. These comparisons characterize the different responses of magnitude-weighted and unit-vote decoding to the tested perturbations.

A.3 Response selection within fixed quotas

For each independently keyed registration, we compute the mean response score over selected carrier coordinates and subtract the average of the corresponding means from 999 keyed random control sets. Each control samples uniformly from the remaining coordinates within each expert and matches the selected set's per-expert counts.

Table 7: Carrier selection under fixed expert quotas.

Selection rule	Response gap
Random	$-4.56 \times 10^{-7} \pm 4.02 \times 10^{-6}$
Weight-magnitude	$+1.47 \times 10^{-3} \pm 7.56 \times 10^{-8}$
Routing-frequency-only	$-1.65 \times 10^{-5} \pm 1.84 \times 10^{-7}$
Routing+response (ours)	$+1.67 \times 10^{-3} \pm 7.89 \times 10^{-8}$

With expert quotas fixed, all four rules have the same routing exposure of 12.53%. Response-based selection produces the largest observed response gap, 1.67×10^{-3},followed by weight-magnitude selection 1.47×10^{-3}. These descriptive results characterize the selected carriers under the enrollment score

A.4 Predicting recovery from pair votes

Antisymmetric paired embedding improves INT4 recovery over single-coordinate embedding at equal energy, increasing the mean recovery margin above the threshold by approximately $1.8\times$. At $\varepsilon = 10^{-3}$, mean WER increases from 90.7% to 92.3%, expanding the margin above 88.67% from approximately 2.0 to 3.6 percentage points. Both configurations use the same squared update budget ε^2: the single-coordinate update has magnitude ε, while the paired update applies opposite perturbations of magnitude $\varepsilon/\sqrt{2}$. The measured SNR analysis below attributes this advantage to lower quantization noise at the reference coordinates.

To explain this improvement, we compare the embedded signal amplitude with the standard deviation of the quantization error. Placing the full perturbation at the carrier coordinate produces a signal of magnitude ε. Paired embedding instead applies opposite updates of magnitude $\varepsilon/\sqrt{2}$, preserving the same total squared update ε^2 while producing a contrast signal of magnitude $\sqrt{2}\varepsilon$.

Let σ_u and σ_v denote the quantization-error standard deviations at the carrier and reference coordinates, and let r_q denote their error correlation. Because verification subtracts the reference coordinate from the carrier coordinate, the corresponding errors also subtract. The variance of this contrast error is therefore

$$\sigma_u^2 + \sigma_v^2 - 2r_q\sigma_u\sigma_v. \tag{12}$$

Using these error statistics for the comparison, the single-coordinate and paired signal-to-noise ratios are

$$\mathrm{SNR}_{\mathrm{single}} = \frac{\varepsilon}{\sigma_u}, \qquad \mathrm{SNR}_{\mathrm{pair}} = \frac{\sqrt{2}\varepsilon}{\sqrt{\sigma_u^2 + \sigma_v^2 - 2r_q\sigma_u\sigma_v}}. \tag{13}$$

Their ratio gives the predicted gain:

$$G = \frac{\mathrm{SNR}_{\mathrm{pair}}}{\mathrm{SNR}_{\mathrm{single}}} = \frac{\sqrt{2}\sigma_u}{\sqrt{\sigma_u^2 + \sigma_v^2 - 2r_q\sigma_u\sigma_v}} \tag{14}$$

Lower quantization noise at the reference coordinates explains the measured pairing advantage. At the default strength, $\sigma_u/\sigma_v = 2.134$ and the error correlation is 0.013 ± 0.014, giving a predicted SNR gain of 1.287 that closely matches the measured 1.288 ± 0.028. Pairing increases the signal amplitude by approximately 41% and the contrast-noise standard deviation by approximately 10%, yielding an

SNR gain of about 28%. Equation 14 identifies the governing condition: with uncorrelated errors, pairing improves SNR when the reference noise variance is lower than the carrier noise variance; equal variances yield $G = 1$.

B Operating Configuration and Calibration Sensitivity

This appendix documents how the operating point (ε, block size K, payload B) was selected and how sensitive enrollment is to the calibration corpus.

B.1 Embedding strength

The strength sweep identifies $\varepsilon = 10^{-3}$ as the smallest tested perturbation that satisfies the registered ownership criterion across all ten registrations under per-output-channel INT4 fake quantization. Increasing ε to 2×10^{-3} raises mean WER from 92.30% to 99.88%, providing additional recovery margin. We select $\varepsilon = 10^{-3}$ to minimize perturbation magnitude among the tested settings that meet this criterion.

Table 8: INT4 recovery crosses the formal gate at $\varepsilon = 10^{-3}$ and saturates at larger strengths.

ε	INT4 sign-vote WER (%)	Interpretation
2.5×10^{-4}	60.94 ± 4.57	Below recovery gate
5×10^{-4}	74.65 ± 3.47	Below recovery gate
10^{-3}	92.30 ± 2.23	Lowest evaluated point above gate
2×10^{-3}	99.88 ± 0.19	Additional robustness margin
4×10^{-3}	100.00 ± 0.00	Saturation

B.2 Redundancy and payload

Three complementary scans examine the trade-off between payload capacity and redundancy: varying block size at fixed embedding strength, varying block size at fixed total embedding energy, and varying payload length at fixed pair budget.

Block size at fixed strength. We fix $B = 256$ and $\varepsilon = 10^{-3}$ and increase K, so the number of carrier pairs grows as $N = BK$. Table 9 shows that INT4 recovery initially decreases from $K = 1$ to $K = 4$, then rises to 92.30% at $K = 32$. Pruning recovery increases to 100.00% at $K \geq 16$, while clean PPL remains between 10.29 and 10.30. Larger blocks thus provide an overall redundancy benefit, although INT4 recovery is not monotonic across the full sweep. This scan increases both redundancy and total embedding energy; the next scan controls for the latter.

Table 9: Block size sweep at fixed $B = 256$ and $\varepsilon = 10^{-3}$.

	INT4 WER (%)		Pruning WER (%)		
K	sign-vote	mean-agg.	sign-vote	mean-agg.	PPL
1	64.53 ± 3.10	64.53 ± 3.10	94.53 ± 1.63	94.53 ± 1.63	10.29
4	55.08 ± 3.14	74.53 ± 2.46	98.32 ± 1.29	84.26 ± 1.56	10.30
8	69.88 ± 2.27	79.80 ± 2.29	99.92 ± 0.25	77.38 ± 2.55	10.30
16	82.46 ± 2.59	87.46 ± 2.46	100.00 ± 0.00	73.16 ± 2.78	10.29
32	92.30 ± 2.23	92.66 ± 1.59	100.00 ± 0.00	75.51 ± 2.72	10.29

Block size at fixed total energy. Sign voting converts additional redundancy into higher pruning recovery without increasing total embedding energy. We fix $B = 256$ and set $\varepsilon_K = 10^{-3}\sqrt{32/K}$, keeping $BK\varepsilon_K^2$ constant across Table 10. Pruning WER rises from 94.57% at $K = 1$ to 100.00% at $K \geq 16$, whereas mean aggregation declines to 75.51% at $K = 32$. This controlled comparison demonstrates the benefit of equal pair weighting under pruning. INT4 recovery is nonmonotonic, reflecting the trade-off between additional votes and weaker per-pair signals.

Table 10: Block size sweep on OLMoE at fixed total embedding energy and $B = 256$.

K	1	4	8	16	32
ε_K ($\times 10^{-3}$)	5.66	2.83	2.00	1.41	1.00
Pruning sign-vote WER (%)	94.57 ± 1.48	98.36 ± 1.33	99.92 ± 0.25	100.00 ± 0.00	100.00 ± 0.00
Pruning mean-agg. WER (%)	94.57 ± 1.48	86.95 ± 1.39	82.42 ± 1.74	78.91 ± 2.71	75.51 ± 2.72
INT4 sign-vote WER (%)	96.45 ± 0.79	88.01 ± 2.42	89.38 ± 1.81	93.52 ± 1.42	92.30 ± 2.23
INT4 mean-agg. WER (%)	96.45 ± 0.79	95.35 ± 1.40	94.53 ± 1.62	95.78 ± 1.45	92.66 ± 1.59

Payload length at fixed pair budget. We fix $N = 8192$ and $\varepsilon = 10^{-3}$ and vary B, using the same carrier pairs across settings (Table 11). Clean WER remains 100.00%, and pruning WER remains at 99.96–100.00%, with all records accepted. As B increases from 256 to 1024, INT4 WER decreases from 92.30% to 68.68%, while pair sign agreement stays near 63.8%. This pattern reflects the reduction in voting redundancy as K falls from 32 to 8. With the corresponding gates, INT4 acceptance falls from 100% to 90% and then 30%. The default $B = 256$, $N = 8192$ setting therefore provides the widest INT4 margin among the evaluated payloads.

Table 11: Payload length sweep at fixed $N = 8192$ and $\varepsilon = 10^{-3}$.

B	K	Gate $t(B)$	$t(B)/B$ (%)	Condition	WER (%)
256	32	227	88.67	Clean	100.00 ± 0.00
				Pruning 10%	100.00 ± 0.00
				INT4	92.30 ± 2.23
512	16	400	78.13	Clean	100.00 ± 0.00
				Pruning 10%	100.00 ± 0.00
				INT4	81.46 ± 1.75
1024	8	719	70.21	Clean	100.00 ± 0.00
				Pruning 10%	99.96 ± 0.08
				INT4	68.68 ± 1.86

B.3 Calibration corpus sensitivity

SparseMark preserves watermark recovery when enrollment calibration changes from WikiText-2 to FineWeb-Edu. FineWeb-Edu enrollment achieves 100.00% WER on clean models and under 10% pruning, and $91.88 \pm 2.04\%$ under per-output-channel INT4 fake quantization; every registration passes the full ownership criterion. All registrations satisfy the full ownership criterion under these three conditions. The corresponding paired teacher-forced PPL differences relative to matched unwatermarked controls are -0.0017,$+0.0055$, and -0.0256, respectively.

C Extended Robustness Evaluation

This appendix collects robustness results beyond the main-text transformations: calibrated packed INT4 (Appendix C.1), gradient-based adaptation and other parameter transformations (Appendix C.2), routing and parameter interventions (Appendix C.3), feedback-guided search (Appendix C.4), and parameter detectability (Appendix C.5).

C.1 Calibrated packed INT4

Calibrated group-wise INT4 quantization with packed storage preserves the watermark on OLMoE: across ten independently keyed registrations at the default configuration ($B = 256$, $N = 8192$, $\varepsilon = 10^{-3}$), watermarked WER is $99.61 \pm 0.37\%$, all ten records satisfy the ownership criterion, and all unwatermarked controls are rejected, with a paired teacher-forced PPL difference of -0.0002 ± 0.0033 on the disjoint FineWeb-Edu evaluation set (Table 12). We compare packed storage, which serializes integer codes and FP16 scales before reloading, with fake quantization, which applies quantization and dequantization in memory using FP32 scales. Both modes use the same calibration procedure

and quantization settings. The 92.30% WER reported in Table 2 and Appendix B uses symmetric per-output-channel INT4 fake quantization. The 99.61% WER reported here uses a separate calibrated group-wise INT4 protocol and is identical across its packed and fake implementations. The difference between these recovery rates does not isolate the effect of packed storage.

Calibration and quantization protocol. Quantizer calibration uses 128 sequences from the WikiText-2 raw test split, truncated to at most 128 tokens each (14,476 tokens in total). The unwatermarked model M_0 and watermarked model M_k are calibrated separately using the same sequences and quantization procedure. Teacher-forced PPL is measured on a FineWeb-Edu evaluation set disjoint from quantizer calibration and scale selection, with identical texts, tokenization, and scoring for both models. The calibration objective is activation-aware scale selection: one forward pass over the calibration sequences records, for each covered matrix, the mean squared input activation a_j of every input channel j (normalized to unit mean within the matrix), and each group's scale is chosen to minimize the activation-weighted reconstruction error $\sum_{j \in \mathrm{grp}} a_j \, (\hat{w}_j - w_j)^2$ of the weights themselves; no layer-output reconstruction or weight update is performed. Quantization covers every two-dimensional weight matrix except the token embedding, the output head, and the normalization weights: the attention projections Q, K, V, O of all 16 layers, the 16 router gate matrices, and all 3,072 expert projections (`gate_proj`, `up_proj`, `down_proj` of 64 experts $\times$ 16 layers), i.e. 3,152 matrices and 6.71×10^9 weights (97.0% of all parameters); the remaining 2.06×10^8 parameters stay in bfloat16. Each covered matrix is quantized to 4 bits with symmetric codes $q \in \{-7, \ldots, 7\}$ over groups of $g = 128$ consecutive entries along the input dimension of each output row (all covered input dimensions of OLMoE are multiples of 128, so no padding occurs). The scale of each group is $s = \alpha \max_{w \in \mathrm{grp}} |w|/7$ with the clipping coefficient $\alpha \in \{0.5, 0.625, 0.75, 0.875, 1.0\}$ selected per group to minimize the activation-weighted error above; codes are rounded to the nearest integer and clamped to $[-7, 7]$, so entries beyond $\pm 7s$ are clipped. No zero point is used. In the packed mode the codes are stored two per byte as unsigned 4-bit nibbles (offset $+8$, low nibble first) and the scales in float16 in a safetensors checkpoint (3.46 GB, 4.125 bits per covered weight including scales); the resident tensors are zeroed before the checkpoint is reloaded. In the fake mode the scales remain in float32 and the dequantized values are written back to the bfloat16 tensors in memory; the two modes therefore differ in the float16 rounding of the scales and in the serialization round trip, and in nothing else.

Verification and utility path after packed storage. After a packed checkpoint is reloaded, the verifier reconstructs each registered coordinate as $\hat{M}[u] = s_{\mathrm{grp}(u)} \, q_u$ (symmetric codes, no zero point), where q_u is the stored integer code and $s_{\mathrm{grp}(u)}$ the float16 scale of its group, cast to bfloat16, and then evaluates the baseline-referenced contrast of Eq. 4 on $\hat{M}$ exactly as in the floating-point setting; the pair votes, block majority, recovery gate, and specificity ranks of Section 3.2 are unchanged. Teacher-forced PPL is computed by dequantizing every packed tensor to bfloat16 and running the standard floating-point forward pass.

Results Table 12 reports the full recovery and utility breakdown. Recovery is identical for every paired record across the two storage modes, establishing watermark recovery after integer packing and reloading. The transformed watermark PPL difference $\Delta_T = \mathrm{PPL}(T(M_k)) - \mathrm{PPL}(T(M_0))$ on FineWeb-Edu is -0.0002 ± 0.0033 for packed storage and $+0.0035 \pm 0.0043$ for fake quantization, indicating little additional utility cost from watermarking under this pipeline.

C.2 Gradient adaptation and other transformations

Extended transformations The remaining transformations require no knowledge of the secret key and preserve recovery in parameter space at or near 100%, with negligible utility change (Table 13). These include fine-tuning and merged LoRA with a matched budget of 200 steps, weight fusion at a ratio of 0.25, random overwriting of one million scalar parameters, and route editing.

Long-horizon adaptation SparseMark preserves exact recovery under utility-improving adaptation, and every observed verification failure is accompanied by a substantial increase in test PPL. After 1000 steps of full-model fine-tuning at learning rate 10^{-5}, all registrations retain 100.00% WER while mean test PPL improves from 10.3006 to 8.6863. At 10^{-4}, mean WER remains 99.80% and all registrations pass verification, despite mean PPL increasing to 70.2129. At 10^{-3}, mean WER

Table 12: Calibrated group-wise INT4 quantization on OLMoE. Calibration uses WikiText-2; teacher-forced PPL is measured on disjoint FineWeb-Edu texts. Absolute PPL values are not directly comparable across evaluation corpora.

Metric	Calibrated INT4: packed	Calibrated INT4: fake
Watermarked WER (%)	99.61 ± 0.37	99.61 ± 0.37
Ownership acceptances ($D = 1$)	100%	100%
Unwatermarked control WER (%)	44.8 ± 2.8	44.8 ± 2.9
Unwatermarked control acceptances	0%	0%
Watermarked teacher-forced PPL	8.6491 ± 0.0033	8.6508 ± 0.0043
Unwatermarked control teacher-forced PPL	8.6493	8.6473
$\Delta_T = \mathrm{PPL}(T(M_k)) - \mathrm{PPL}(T(M_0))$	-0.0002 ± 0.0033	$+0.0035 \pm 0.0043$

Table 13: Fine-tuning, LoRA merging, fusion, overwrite, and routing edits preserve at least 99.92% recovery at the default strength.

Transformation	PPL	WER (%)
Fine-tuning 200	10.4240 ± 0.0148	100.00 ± 0.00
Merged LoRA 200	10.9076 ± 0.0461	100.00 ± 0.00
Fusion 0.25	10.2884 ± 0.0043	100.00 ± 0.00
Overwrite 10^6	10.3201 ± 0.0151	99.92 ± 0.16
Route edit	10.5075 ± 0.0777	100.00 ± 0.00

decreases to 87.66% and four of ten registrations pass verification; every run exhibits severe utility degradation, with PPL between 1632 and 3028.

Last-layer fine-tuning preserves exact recovery at both tested learning rates, as does LoRA at 10^{-4}. LoRA at 10^{-3} causes all registrations to fail verification, with WER ranging from 57.42% to 79.30%. However, even the lowest PPL in this setting is 27.68, or 2.69 times the original value, and PPL reaches 493.58. Across the evaluated configurations, no run causes verification failure while maintaining or improving the original test PPL.

Table 14: Long-horizon adaptation across full-model, last-layer, and LoRA fine-tuning for 1000 steps on WikiText-2.Values are mean $\pm$ SD across ten registrations; original test PPL is 10.3006

Condition	WER (%)	PPL
Full-model fine-tuning,lr 10^{-5}	100.00 ± 0.00	8.6863 ± 0.0293
Full-model fine-tuning,lr 10^{-4}	99.80 ± 0.21	70.2129 ± 10.6639
Full-model fine-tuning,lr 10^{-3}	87.66 ± 2.93	2309.8907 ± 597.8578
Last-layer fine-tuning, lr 10^{-5}	100.00 ± 0.00	10.6107 ± 0.0098
Last-layer fine-tuning, lr 10^{-4}	100.00 ± 0.00	12.0891 ± 0.3047
LoRA, lr 10^{-4}	100.00 ± 0.00	10.9200 ± 0.0470
LoRA, lr 10^{-3}	70.51 ± 5.65	83.7961 ± 144.8713

C.3 Key-unaware parameter attacks

Following the robustness requirements for adaptive routing established in PathMark, we evaluate three routing attacks on OLMoE: router-logit noise, targeted expert removal, and proxy-trigger route unlearning, with ten records per cell. Route unlearning runs for 10 epochs with 20 optimizer steps per epoch, giving 200 optimizer steps in total; all reported unlearning results use the final epoch. As demonstrated in Table 15, sign-vote recovery consistently remains at or above the predefined threshold across various selection rules, attack intensities, and block sizes. Router-logit noise and route unlearning alter routing behavior while leaving the registered expert weights unchanged, so they preserve the parameter readout directly; targeted expert removal modifies registered weights and therefore tests the voting mechanism itself.

Table 15: Attack-intensity sweep under routing-aware selection. Sign-vote WER remains at least 99.92% across stronger routing attacks; the only sub-perfect case occurs under 50% expert removal after severe utility degradation (PPL ≈ 77.97).

Attack	Mean-aggregation WER (%)	Sign-vote WER (%)	ΔPPL
Logit noise $\sigma = 0.5$	100.00 ± 0.00	100.00 ± 0.00	6.5058
Logit noise $\sigma = 1.0$	100.00 ± 0.00	100.00 ± 0.00	253.7971
Logit noise $\sigma = 1.5$	100.00 ± 0.00	100.00 ± 0.00	5429.5421
Logit noise $\sigma = 2.0$	100.00 ± 0.00	100.00 ± 0.00	15714.8467
Removal 10%	65.27 ± 3.26	100.00 ± 0.00	1.9042
Removal 30%	55.82 ± 3.02	100.00 ± 0.00	12.2902
Removal 50%	52.70 ± 2.90	99.92 ± 0.16	67.6702
Unlearning (200 steps)	100.00 ± 0.00	100.00 ± 0.00	12.4048

Under proxy-trigger route unlearning (epoch 10), sign-vote recovery is 100.00 ± 0.00 at every block size $K \in \{1, 4, 8, 16, 32\}$. Across all three attack families, no evaluated configuration removes the codeword under sign-vote decoding at acceptable model quality; the single departure from perfect WER (50% expert removal, 99.92 ± 0.16) occurs after utility degradation (PPL ≈ 77.97).

Routing-side interventions and removal attempts We additionally evaluate routing-frequency noise and stratum-pair decorrelation at relative budgets $s \in \{0.02, 0.05, 0.10, 0.20\}$, using ten independently keyed records per budget. Here s is a single global relative Frobenius budget over the attacked weight subset,

$$\|\Delta W\|_F = s\|W_{\text{ref}}\|_F \tag{15}$$

rather than a per-row or per-stratum budget. Routing-frequency noise ranks expert down-projections by public routing frequency, perturbs the top 1,024 selected rows with Gaussian noise, and jointly rescales the perturbation to the prescribed budget; W_{ref} is the selected weight subset. For stratum-pair decorrelation, a fixed-seed random matching pairs candidate rows within each layer, expert, and each pair receives equal-and-opposite perturbations along a shared random unit direction. The joint perturbation is scaled to the same global budget, with W_{ref} given by the complete down-projection matrices of the touched strata.

Table 16: Mean-aggregation and sign-vote WER under key-unaware routing-frequency noise and stratum-pair decorrelation.

Family	L_2	Mean-aggregation WER (%)	Sign-vote WER (%)	ΔPPL
routefreq (top-1024)	0.02	100.00 ± 0.00	100.00 ± 0.00	$+0.0023$
routefreq (top-1024)	0.05	100.00 ± 0.00	100.00 ± 0.00	$+0.0015$
routefreq (top-1024)	0.10	100.00 ± 0.00	100.00 ± 0.00	$+0.0026$
routefreq (top-1024)	0.20	100.00 ± 0.00	100.00 ± 0.00	$+0.0020$
pairdecorr	0.02	100.00 ± 0.00	100.00 ± 0.00	$+0.0186$
pairdecorr	0.05	100.00 ± 0.00	100.00 ± 0.00	$+0.1189$
pairdecorr	0.10	100.00 ± 0.00	99.84 ± 0.19	$+0.4119$
pairdecorr	0.20	97.42 ± 0.94	90.82 ± 1.45	$+1.1724$

C.4 FEEDBACK-GUIDED SEARCH

Adaptive parameter-space attack The adversary receives extended scalar T_{Param} feedback. The 200-query budget comprises one initial evaluation and 199 sequential proposals. Each proposal uniformly samples an eligible expert down-projection group: one expert slice of a floating-point 3D parameter tensor, or an entire floating-point 2D parameter tensor. Starting from the current accepted model, the attack adds independent zero-mean Gaussian noise with standard deviation $0.05 \cdot \text{RMS}(W_g)$, where W_g is the current group. The proposal is retained only if T_{Param} strictly decreases; otherwise, the group is restored. Accepted perturbations accumulate without a cumulative distortion constraint. No utility feedback is provided.

Across ten registrations, clean and INT4 initializations accept 91–118 and 91–116 proposals, respectively (means 101.0 and 101.2). The corresponding mean absolute decreases in the queried score are approximately 0.0208 and 0.0210. Thus, unchanged final clean WER coexists with a reduced search objective. All final specificity tests pass, so the ownership outcomes in Table 3 are determined by the recovery gate. The ledger records score trajectories and proposal decisions for each query.

C.5 Parameter detectability

This subsection measures the parameter-space footprint of SparseMark embedding and evaluates learned detectors that operate without the watermark key.

Distributional footprint We enroll independent keys on OLMoE-1B-7B at $\varepsilon{=}10^{-3}$ with $N{=}8{,}192$ paired writes under the routing-aware protocol. For each seed we compute the two-sample Kolmogorov-Smirnov statistic between every expert down_proj weight block of M_0 and the corresponding block of M_k (one flattened block vector per sample), and measure random within-stratum pair contrasts to probe antisymmetric write signatures.

Median block-level KS is 2.6×10^{-5} across all seeds; the maximum is 1.15×10^{-4} . Pair-contrast KS is 2.0×10^{-3} uniformly.

Learned detectors We train logistic regression and a two-layer MLP with hidden widths $(64, 32)$ on 29 predefined statistics of expert down-projection blocks, comprising moments, quantiles, absolute-weight statistics, and pair-contrast summaries. The dataset contains 1,024 clean blocks from one unwatermarked model and $10 \times 1,024$ watermarked blocks from ten independently keyed registrations.

Five-fold GroupKFold uses ten watermarked groups, one per registration, and five clean groups assigned by fixed block enumeration index modulo five, independently of seed and layer. Each test fold contains 2,048 watermarked blocks and 204 or 205 clean blocks; the corresponding training fold contains 8,192 watermarked blocks and 820 or 819 clean blocks, respectively. Watermark keys are disjoint between training and testing, but versions of every underlying expert block occur on both sides of each split. The evaluation therefore holds out keys while retaining shared expert-block identities within a single base model.

Standardization is fitted separately on each training fold and applied unchanged to its test fold. All 29 features are fixed in advance, with no data-driven feature selection. MLP early stopping uses a validation subset drawn only from the training fold. Balanced accuracy is evaluated at the default decision threshold of 0.5, without threshold tuning.

The two evaluated detectors show near-chance discrimination on held-out watermark keys within a single base model. Using the 29 predefined parameter statistics, logistic regression achieves an AUC of 0.492 ± 0.010 and the MLP achieves 0.503 ± 0.020; both have balanced accuracy of 0.500 in every fold. The evaluation measures discrimination across watermark keys with expert-block identities shared between training and testing.

Table 17: AUC is threshold-free; balanced accuracy is reported at the default threshold of 0.5.

Classifier	AUC	Per-fold range	Balanced Acc.
Logistic Regression	0.492 ± 0.010	0.478–0.508	0.500
MLP (64,32)	0.503 ± 0.020	0.468–0.529	0.500

D Enrollment, Registration, and Verification

This appendix states the exact enrollment contract (Appendix D.1), the registration and integrity checks (Appendix D.2), the specificity tests and wrong-code bound (Appendix D.3), and the coexistence guarantees for multiple registrations (Appendix D.4).

Algorithm 1 SparseMark Enrollment and Embedding

Require: Base model M_0, calibration corpus $\mathcal{D}$, secret material $k = (K_\pi, K_c, K_s)$, payload length B, pair budget N, strength ε.
Ensure: Watermarked model M_k and registration record $\mathcal{R}$.
1: Estimate expert routing frequencies f and row-response scores ρ on $\mathcal{D}$.
2: Distribute N pairs evenly across eligible experts; assign remaining pairs by descending f.
3: Within each expert, select down-projection carrier coordinates u_i by descending ρ_u.
4: Use K_π to pair each u_i with an unused reference v_i in the same expert's down-projection.
5: Apply a separate keyed ordering and divide the pairs into B blocks G_j, each containing N/B pairs.
6: Set $c \leftarrow K_c$ and generate spreading signs s_i from K_s.
7: $M_k \leftarrow \mathrm{copy}(M_0)$.
8: **for** each block j and pair $i \in G_j$ **do**
9: $\quad b_i \leftarrow M_0[u_i] - M_0[v_i]$.
10: $\quad M_k[u_i] \leftarrow M_0[u_i] + (\varepsilon/\sqrt{2}) s_i c_j$.
11: $\quad M_k[v_i] \leftarrow M_0[v_i] - (\varepsilon/\sqrt{2}) s_i c_j$.
12: **end for**
13: Before release, register the reconstruction assets, baseline contrasts $\{b_i\}$, and verification settings in $\mathcal{R}$ (Appendix D.2).
14: **return** $(M_k, \mathcal{R})$.

D.1 Carrier construction

Candidate pool and reference pairing The candidate pool U enumerates expert projection rows across all layers, excluding coordinates occupied by previous enrollments. Although the pool includes gate, up, and down projections, embedding uses only down-projection coordinates. For each selected carrier u_i, the pairing key selects an unused reference $v_i \neq u_i$ from the same expert's down-projection. The eligible matrices and pairing rule are fixed in the enrollment manifest.

Enrollment calibration yields 21,980 tokens per layer, with sequence lengths ranging from 27 to 429 tokens (mean 171.7). Across all 1,024 experts, indexed by layer and expert, the median routed token count is 2,519, with an interquartile range of 1,671–3,584. Only three experts receive fewer than 32 tokens and are excluded, leaving 1,021 eligible experts. These counts characterize routing coverage; Appendix B.3 separately evaluates the sensitivity of watermark recovery to the calibration corpus.

Pair budget allocation The global embedding budget of N carrier pairs (default $N = 8192$) is distributed approximately evenly across eligible experts to prevent carrier concentration. Let $\mathcal{E}_{\text{eligible}}$ denote the set of experts across all layers whose total routed calibration tokens meet the minimum threshold ($|T_e| \geq 32$), yielding $|\mathcal{E}_{\text{eligible}}| = 1{,}021$ out of $1{,}024$ total experts on OLMoE. Each eligible expert $(\ell, e) \in \mathcal{E}_{\text{eligible}}$ is assigned a uniform base quota of

$$q = \left\lfloor \frac{N}{|\mathcal{E}_{\text{eligible}}|} \right\rfloor \tag{16}$$

carrier pairs ($q = 8$ at the default operating point). The residual remainder $r = N - q \cdot |\mathcal{E}_{\text{eligible}}|$ (with $r = 24$ on OLMoE) is distributed by allocating one additional pair ($q + 1$) to the top-r experts ordered by routing frequency.

Within each expert, carrier candidates are ranked by descending ρ_u, with ties resolved by canonical coordinate order. This ordering rule is fixed in the enrollment manifest.

Ranking statistic Index all calibration token occurrences by $n = 1, \ldots, T$, where n uniquely identifies an input sequence and a position within that sequence. Let $e = (\ell, a)$ identify expert a in layer ℓ, and let $e(u)$ denote the expert containing coordinate u. Define $\phi_{M_0}(u; n)$ as the output of the expert row containing u at token occurrence n, multiplied by the router gate weight assigned to $e(u)$; it is zero when that expert is not selected. Let $\mathcal{T}_e$ be the set of calibration token indices routed

to expert e. We define

$$f_e = \frac{|\mathcal{T}_e|}{T}, \qquad \rho_u = \left| \frac{1}{T} \sum_{n=1}^{T} \phi_{M_0}(u; n) \right|, \qquad r_u^{\text{cond}} = \frac{\rho_u}{f_{e(u)}} = \left| \frac{1}{|\mathcal{T}_{e(u)}|} \sum_{n \in \mathcal{T}_{e(u)}} \phi_{M_0}(u; n) \right|. \tag{17}$$

Here, f_e measures expert routing frequency, ρ_u measures the aggregate gate-weighted row response, and r_u^{cond} measures its magnitude conditional on routing to $e(u)$. The conditional response is defined for $f_{e(u)} > 0$; eligible experts satisfy $|\mathcal{T}_e| \geq 32$.

Keyed pseudorandom permutation For each expert, a keyed Fisher–Yates permutation orders the unused down-projection reference coordinates, using the PRF counter stream derived from K_π in the pair domain. Occupancy bookkeeping excludes coordinates drawn by earlier enrollments; given the key and the enrollment history, the shuffle is deterministic. Each block symbol reads one bit from K_c, which is sampled independently at enrollment and retained for verification. Each spreading sign uses the least significant bit of the first byte of $\mathrm{HMAC\text{-}SHA256}(K_s, \mathrm{canonical_json}(\text{pair metadata}))$. The code bits are sampled independently of the pairing and spreading keys. In the public-score targeting study of Appendix A.1, the control arms instead derive their code and spreading signs from a shared secret using SHA256 with distinct domain labels; all other embedding and verification components are identical across arms.

Reconstruction consistency Across all registrations, reconstruction from the retained keys and candidate pool exactly reproduces the registered pair assignments, block indices, code symbols, spreading signs, and write signs. All pairing hashes match their commitments, and every pair satisfies the registered same-expert, down-projection constraint.

Stored-reference equivalence. We checked verification using stored pair contrasts against verification using M_0 directly on clean and adaptive row-noise candidates, with $N = 8192$. Computing and storing the contrasts in the verifier's FP64 precision reproduced all 8192 pair displacements bit for bit in each case. Pair votes, block scores, recovered codewords, T_{Param}, recovery rates, ownership decisions, and both specificity p-values were also bitwise identical. Stored block assignments matched those reconstructed from the key. An FP32 representation reduced contrast storage to 32 KiB, with maximum absolute rounding error at most 7.45×10^{-9}. Although 3979 pair displacements differed bitwise in each case under FP32 storage, the pair votes, block scores, recovered codewords, T_{Param}, recovery rates, and decisions remained unchanged in these checks.

D.2 Registration and integrity checks

To prevent ambiguity in multi-watermark coexistence and coordinate collisions, SparseMark defines a registration contract summarized in Table 18. The contract distinguishes the keys and retained enrollment data used to reconstruct the coordinate structure and signs, the stored baseline contrasts used for parameter verification, and any additional reference data required for structural alignment. Registration binds these assets through cryptographic commitments and integrity hashes.

Before candidate weights are loaded into GPU memory, the verifier validates the retained assets against their registered digests, checks the key commitments, and reconstructs and validates the pairing. Any mismatch aborts verification. All 160 sequential registrations pass pairing hash verification and carrier disjointness checks (Appendix D.4). In separate integrity tests, the verifier rejects all four tested forms of tampering before model loading: missing occupancy data, modified occupancy data, substituted candidate pools, and altered pairing hashes. These results establish consistency checks for the tested allocation assets.

Pair reconstruction requires K_π together with the retained candidate pool, registered configuration, and occupancy data applicable at enrollment. The relevant key components and configuration also determine the block assignment, spreading signs, and expected codeword. The baseline contrasts b_i, however, are computed from M_0 at enrollment and stored separately; they cannot generally be reconstructed from the keys. Structural alignment, when needed, additionally requires the reference data specified by the alignment procedure.

Table 18: Registration assets and integrity requirements for SparseMark. The 64 KiB storage figure covers baseline contrasts only.

Component	Specification
Registration record	Protocol version, selection rule, allocation scheme, configuration digest $H(\mathcal{M}_{cfg})$, candidate pool reference and digest $H(U)$, pairing digest $H(\pi)$, and commitments to K_π, K_c, and K_s.
Coexistence extension	Reference and digest $H(O)$ for the occupancy data applicable at enrollment, together with the occupied coordinate count. These data are retained for pairing reconstruction.
Baseline contrasts	Retained values $b_i = M_0[u_i] - M_0[v_i]$, indexed by the registered pair order, with an asset reference, numerical format, and digest. For $N = 8192$, FP64 contrast storage requires 64 KiB, excluding coordinate metadata and alignment references.
Alignment references	When structural alignment is used, retain the reference data required by the selected procedure, together with asset references and digests. Their storage is additional to the baseline contrasts.
Integrity checks	Validate the registered asset digests, including those of the baseline contrasts and any required alignment references, and check the key commitments. Reconstruct and validate the pairing. Any mismatch aborts verification before model loading.
Statistical scope	This bound controls independent wrong-code acceptance for fixed registration assets; it does not establish security against adaptively constructed registration records.

Algorithm 2 SparseMark Verification and Extraction

Require: Candidate model M in the registered coordinate order, secret material k, registration record $\mathcal{R}$ and its retained assets.
Ensure: Ownership decision D and recovery rate R.
1: Validate the registration assets; reject on mismatch.
2: Reconstruct pairs (u_i, v_i), blocks G_j, codeword c, and spreading signs s_i from k and the retained assets.
3: Load baseline contrasts $\{b_i\}$ and the registered verification settings.
4: **for** each pair i **do**
5: $\quad d_i \leftarrow (M[u_i] - M[v_i]) - b_i$.
6: $\quad z_i \leftarrow \mathrm{sign}(s_i d_i)$.
7: **end for**
8: **for** each block j **do**
9: $\quad V_j \leftarrow |G_j|^{-1} \sum_{i \in G_j} z_i$; $\widehat{c}_j \leftarrow \mathrm{sign}(V_j)$.
10: **end for**
11: $R \leftarrow B^{-1} \sum_{j=1}^{B} \mathbf{1}[\widehat{c}_j = c_j]$.
12: $T_{\mathrm{Param}} \leftarrow B^{-1} \sum_{j=1}^{B} c_j V_j$.
13: Compute p_{rel} and p_{wk} using the registered relabeling and wrong-code-key tests.
14: $D \leftarrow \mathbf{1}[R \geq t^\star/B]\, \mathbf{1}[p_{\mathrm{rel}} \leq \alpha]\, \mathbf{1}[p_{\mathrm{wk}} \leq \alpha]$.
15: **return** (D, R).

For a fixed registered configuration, deterministic reconstruction produces the same coordinate structure and signs as an explicit manifest. Given identical baseline contrasts, aligned candidate coordinates, and numerical precision, the two representations therefore yield identical verification outputs. This removes the need to retain an explicit per-pair coordinate list, while preserving the requirement to retain the reconstruction inputs, baseline contrasts, and any alignment references. Their registered commitments and digests allow the verifier to detect inconsistencies with the enrollment record.

D.3 SPECIFICITY TESTS AND WRONG-CODE BOUND

Algorithm 2 summarizes parameter-space verification; the specificity tests and statistical bound are detailed below.

For $h \in \{\mathrm{rel}, \mathrm{wk}\}$, let $\{T_h^{(m)}\}_{m=1}^{L_h}$ denote statistics generated from the null family h. For the same enrolled candidate, the verifier computes enrollment-conditional upper-tail ranks

$$p_h = \frac{1 + \sum_{m=1}^{L_h} \mathbf{1}[T_h^{(m)} \geq T_{\mathrm{Param}}]}{L_h + 1} \qquad L_{\mathrm{rel}} = 999,\ L_{\mathrm{wk}} = 10{,}000. \tag{18}$$

The minimum attainable ranks are $1/(L_{\mathrm{wk}} + 1) = 1/10{,}001 \approx 9.999 \times 10^{-5}$ and $1/(L_{\mathrm{rel}} + 1) = 1/1{,}000$, respectively, attained when no null replicate reaches or exceeds the observed statistic. The relabeling null ($h = \mathrm{rel}$) changes only the codeword labels while holding the carrier structure fixed. The independent wrong-code-key null ($h = \mathrm{wk}$) retains that carrier structure but replaces the secret code signs and within-block spreading signs under separate key domains.

For the relabeling test, the observed code labels must be exchangeable with the null labels conditional on the fixed carrier responses. For the wrong-code-key test, the observed code and spreading signs must be exchangeable with their null redraws conditional on the fixed carrier structure and pair contrasts. Under these respective null conditions, the upper-tail ranks are valid, with ties handled conservatively.

With the bit-recovery rate WER as defined in the main text (Section 3.2), the ownership decision in Eq. 8 in the main text combines $WER \geq 227/256$ with $p_{\mathrm{rel}} \leq 0.05$ and $p_{\mathrm{wk}} \leq 0.05$. Both null families and the registered statistic are fixed before any candidate is observed.

Wrong-code bound under independence. Consider a candidate model M and a wrong key with components K'_π, K'_c, and K'_s. Condition on M, K'_π, K'_s, and all fixed verification assets, including the baseline values and block construction. This conditioning fixes the pairing, the spreading signs, and the baseline-referenced pair contrasts $\{d_i\}$, thereby determining the decoded candidate word $\hat{\mathbf{c}}$ under the wrong-key extraction. Let $S = \{j : \hat{c}_j \neq 0\}$ denote the non-abstaining blocks with $n = |S| \leq B$.

Under the independent wrong-code null, the target bits $\{c'_j\}_{j=1}^{B}$ are mutually independent uniform signs, independent of the conditioned quantities. For code bits generated by a PRF, this is the idealized random-function model; domain separation does not by itself establish information-theoretic independence. For $j \in S$, the decoded block $\hat{c}_j$ is a deterministic function of the conditioning, so the event $\{\hat{c}_j = c'_j\}$ occurs with probability $1/2$ independently across non-abstaining blocks. Blocks with $\hat{c}_j = 0$ ($j \notin S$) contribute zero to the recovery count. Hence

$$H \mid (M, K'_\pi, K'_s) \sim \mathrm{Bin}(n, 1/2), \qquad \Pr[H \geq t] \leq \Pr[\mathrm{Bin}(256, 1/2) \geq t] = \sum_{r=t}^{256} \binom{256}{r} 2^{-256}. \tag{19}$$

To enforce an independent wrong-code false acceptance probability target of at most 2^{-128}, the recovery threshold $t^\star$ is chosen as the minimal integer satisfying

$$t^\star = \min\left\{t \in \{1, \ldots, 256\} : \Pr[\mathrm{Bin}(256, 1/2) \geq t] \leq 2^{-128}\right\} = 227. \tag{20}$$

Minimality is established directly at the threshold boundary: for $t = 226$, the binomial tail probability is $\Pr[\mathrm{Bin}(256, 1/2) \geq 226] \approx 1.1298 \times 10^{-38} \approx 2^{-126.06} > 2^{-128}$, whereas for $t = 227$, it drops to $\Pr[\mathrm{Bin}(256, 1/2) \geq 227] \approx 1.4847 \times 10^{-39} \approx 2^{-128.99} \leq 2^{-128}$. Thus, 227 is the smallest integer threshold meeting this target, giving the registered gate $R \geq 227/256$.

The bound is attained when all blocks are non-abstaining ($n = 256$); tied blocks can only reduce the tail probability. The full decision implies passage of the recovery gate and therefore satisfies the same bound. Candidate votes and decoded blocks may be arbitrarily dependent, provided the target code bits remain independent of the conditioned readout. The bound does not cover candidates selected using feedback that depends on the target code bits.

D.4 COEXISTING REGISTRATIONS

All coexistence results use the registered sign-vote verifier. Across ten seeds, all 160 watermarks recover all 256 bits after 16 sequential registrations and satisfy both specificity tests, with

$p_{\mathrm{wk}} = 1/10001$ and $p_{\mathrm{rel}} = 0.001$. Subsequent registrations preserve exact recovery of every earlier watermark. On the final model M_{16}, we test all ordered combinations of registration j's carrier coordinates and block map with registration i's codeword and spreading signs, for $i \neq j$. All 2,400 combinations (ten seeds $\times$ 16 registrations $\times$ 15 nonmatching registrations) are rejected after extraction, null testing, and the full ownership decision. All 160 correctly matched registrations are also rejected on the unwatermarked model.

After the shared INT4 transformation, all 160 watermarks remain accepted. WER is $96.02\% \pm 1.53\%$, with a minimum of $227/256$, exactly the registered recovery threshold.

E Evaluation Protocols and Baselines

This appendix states the statistical units and reporting conventions, the downstream-task evaluation protocols, and the shared baseline contract with its cross-backbone results.

E.1 Statistical units and utility evaluation

Unless stated otherwise, results are reported as mean $\pm$ sample SD across ten independently keyed enrollment records per condition. Transformed instances and null draws are within-record repetitions. Results are summarized separately for each backbone, and false-accept counts identify the evaluated candidate cohort.

For SparseMark, we additionally report 95% bootstrap confidence intervals for mean WER by resampling the enrollment records. Paired utility intervals are computed by resampling the ten within-record differences between watermarked and matched unwatermarked models. These intervals quantify variation across enrollments on the fixed evaluation data. When all record-level values are identical, the bootstrap interval reduces to a single point.

Per-record acceptance decisions are provided in the machine-readable ledger (Appendix F).

Table 19: SparseMark WER on OLMoE across independent enrollment records per condition.

Condition	WER (%)	95% bootstrap CI (%)
Clean	100.00	[100.00, 100.00]
Pruning 10%	100.00	[100.00, 100.00]
INT8	100.00	[100.00, 100.00]
INT4	92.30	[91.05, 93.67]
Gaussian noise	100.00	[100.00, 100.00]

Downstream-task evaluation protocol. All downstream evaluations use zero-shot prompts. MMLU is evaluated on all $14{,}042$ official test questions from 57 subjects; each multiple-choice item is scored by the mean token log-likelihood of its listed options. The main-text MMLU result is micro-averaged over questions, while subject-level accuracies and their unweighted mean are retained as the macro diagnostic. ARC-Challenge uses the same zero-shot scoring on all 299 validation questions. GSM8K shows a mean paired accuracy increase of 0.660 percentage points on the evaluated test set. Resampling registration-level differences gives a 95% CI of $[+0.394, +0.917]$ percentage points, quantifying variation across watermark registrations on these fixed questions. Resampling questions while keeping all registration outcomes for each question together gives a 95% CI of $[-0.743, +2.047]$ percentage points, indicating uncertainty in the direction of the change across questions. WikiText-2 PPL is teacher-forced on the fixed 128-text held-out bank.

Utility attribution decomposition. For each seed and transformation T, four paired models are evaluated on one PPL bank: the clean unwatermarked model M_0, the clean watermarked model M_k, the transformed unwatermarked control $T(M_0)$, and the transformed watermarked model $T(M_k)$. The transformation-only PPL increase is $\mathrm{PPL}(T(M_0)) - \mathrm{PPL}(M_0)$. We define the clean watermark PPL difference as $\Delta_{\mathrm{clean}} = \mathrm{PPL}(M_k) - \mathrm{PPL}(M_0)$ and the transformed watermark PPL difference as $\Delta_T = \mathrm{PPL}(T(M_k)) - \mathrm{PPL}(T(M_0))$. The watermark–transformation interaction is

$$I_T = \Delta_T - \Delta_{\mathrm{clean}} = \big[\mathrm{PPL}(T(M_k)) - \mathrm{PPL}(M_k)\big] - \big[\mathrm{PPL}(T(M_0)) - \mathrm{PPL}(M_0)\big] \qquad (21)$$

Table 20: SparseMark preserves downstream utility across all 57 MMLU subjects.

Metric	**Result**
Baseline macro accuracy	34.64
SparseMark macro accuracy	34.63 ± 0.10
Macro change	-0.01 pp
Median subject change	0.00 pp
Mean absolute subject change	0.47 pp
Subjects with $\lvert\Delta\rvert \leq 1$ pp	51/57 (89.5%)
Largest increase	Econometrics: $+1.93$ pp
Largest decrease	High School Physics: -1.52 pp

Table 21: Paired utility changes for SparseMark on OLMoE, computed as watermarked minus matched unwatermarked performance across ten enrollment records. Confidence intervals are computed by percentile bootstrap over the ten paired registration-level differences, conditional on the fixed evaluation set. For GSM8K, an additional paired bootstrap over questions gives a 95% CI of $[-0.743, +2.047]$ percentage points, retaining all ten registration outcomes for each sampled question.

Metric	Mean paired change	95% CI across registrations
WikiText-2 PPL	-0.0004	$[-0.0031, +0.0025]$
MMLU micro (pp)	$+0.008$	$[-0.061, +0.078]$
MMLU macro (pp)	-0.011	$[-0.070, +0.051]$
ARC-Challenge (pp)	-0.067	$[-0.502, +0.368]$
GSM8K (pp)	$+0.660$	$[+0.394, +0.917]$

This interaction measures how the transformation changes the PPL difference associated with watermarking. Under INT4 and 10% pruning, I_T is -0.0010 ± 0.0078 and $+0.0014 \pm 0.0101$, respectively, compared with transformation-only increases of $+3.1049$ and $+2.7980$.

E.2 Baseline configurations

All baselines are enrolled and evaluated on OLMoE under the same ten record units, the same PPL bank, and the shared transformation set, restricted to transformations each protocol defines; extraction follows each method's native protocol with its native thresholds unchanged (Table 22). Numeric hyperparameter constants of each enrollment are enumerated in the configuration manifest accompanying the artifact release. For PathMark, we evaluate ten groups of 100 watermark embedding schemes, each trained for ten epochs following the original protocol; detection success is computed within each group using PathMark's native verification criterion, and the ten group rates are averaged.

Table 22: Baseline watermarking protocols, payload representations, and native extraction configurations evaluated on OLMoE under the shared evaluation contract.

Method	Signal and payload	Extraction on OLMoE
ELLMark	Multi-bit signature modulated on grouped linear layers	Native grouped decoding; enrolled on feed-forward projections
GaussMark	Structural parameter-noise statistic	Native moment test at its threshold
NeuralMark	Hash-filtered signature	Optimization-based native extraction
PathMark	Triggered-path route signature	Restricted white-box route tracing

E.3 Additional baseline results

We evaluate ELLMark, NeuralMark, and GaussMark on Qwen1.5-MoE-A2.7B and GPT-OSS-20B under the clean protocol of Section 4.2, using ten independently keyed records per method and backbone and the same frozen 128-text WikiText-2 PPL bank.

At its original operating point ($\alpha = 32$), NeuralMark yields an OLMoE ΔPPL of 7699.90 ± 2312.58 and a Qwen ΔPPL of $+2647.58 \pm 723.72$. Additional sweeps over $\alpha \in \{16, 8, 4, 2, 1\}$ yield minimum PPLs of 406.00 on OLMoE and 27.29 on Qwen.As α decreases, the WER of both models declines overall, reaching 91.02% for Qwen and 69.14% for OLMoE at $\alpha = 1$.

ELLMark recovers $69.69\% \pm 14.36\%$ on GPT-OSS-20B, with no records meeting its owner-acceptance criterion.

GaussMark's generation-side detection rates remain near the nominal level on both backbones ($7.00\% \pm 8.23\%$ and $4.00\% \pm 9.66\%$); it defines no owner-acceptance gate.

PathMark achieves $100.00\% \pm 0.00\%$ recovery on both additional backbones, with ΔPPL of $+0.18 \pm 0.22$ on Qwen and $+27.82 \pm 3.98$ on GPT-OSS-20B. On clean OLMoE, it detects 832 of $1{,}000$ schemes (83.20% WER).

F Reproducibility Assets

Compute and verification cost For OLMoE with $B = 256$ and $N = 8192$ on a single NVIDIA A100, enrollment requires 128 model forward calls, one per calibration text, all during routing and response statistics collection. Subsequent enrollment stages require no additional forward calls or gradient optimization. The complete enrollment function took 168.5 seconds before parameter-lookup caching was introduced.

With parameter-lookup caching, end-to-end verification took 80.4–88.5 seconds across ten trials for seed 0, with a mean of 83.35 seconds, compared with 107.4–110.5 seconds before caching. The displacement-extraction stage decreased from 27.8–31.9 seconds to 1.96–2.30 seconds. Caching preserved enrollment-payload hashes and produced identical pair votes; recovery and ownership decisions were unchanged in all ten verification trials. Verification uses stored reference contrasts b_i and requires neither access to M_0 nor model forward passes.

Given resolved parameter locations, contrast extraction and pairwise voting require $\mathcal{O}(N)$ work, followed by $\mathcal{O}(B)$ block decoding. Codeword relabeling holds block scores fixed and costs $\mathcal{O}(L_{\mathrm{rel}} B)$. Explicit redraws of pairwise spreading signs and codewords require recomputing block scores and cost $\mathcal{O}(L_{\mathrm{wk}} N)$, where $L_{\mathrm{rel}} = 999$ and $L_{\mathrm{wk}} = 10{,}000$. These arithmetic costs exclude pairing reconstruction, parameter lookup, and model loading. The differential readout requires $\mathcal{O}(N)$ working memory, excluding model storage and null-test workspace. Stored FP64 reference contrasts occupy $8N$ bytes, or 64 KiB at $N = 8192$; pair metadata and permutation-alignment references are additional.

Models, corpora, and licenses. Experiments use OLMoE-1B-7B-0125, Qwen1.5-MoE, and GPT-OSS as publicly released under their respective licenses. Calibration uses a WikiText-2 sample; utility is measured on a held-out WikiText-2 bank and on MMLU, ARC-Challenge, and GSM8K official splits. License identifiers for each artifact are listed in the release.

Machine-readable ledger. Each released condition carries per-seed records keyed by method, backbone, transformation, and seed, including decision counts, recovery components, utility values, and per-cell availability statuses; content hashes are fixed at freeze. The ledger is the single source for every number in this appendix and in the main text.