

SkillMarker: A KGW-Based Watermarking Method for Agent Skill Ownership Verification

Alan Tang¹, Lifeng Wang^{1,2}, Gang Wu¹, Tao Zhang¹, Yantao Jia^{1,*}, and Xiaofeng Zhu¹

¹Huawei Inc. ²Harbin Institute of Technology, Shenzhen

Agent skills encode operational procedures and domain expertise as reusable knowledge assets, yet their distribution in plaintext exposes them to unauthorized copying and leakage. However, practical mechanisms for skill ownership verification and leakage tracing remain underdeveloped. Watermark embedding before distribution offers a potential solution, but structured, compact skill instructions impose stringent requirements for factual and procedural accuracy. To address these challenges, we introduce SkillMarker, a KGW-based watermarking method that jointly targets functionality preservation and strong watermark embedding. SkillMarker unifies quality-constrained rewriting with signal-oriented candidate search. The former separates essential knowledge from flexible expression to preserve instruction fidelity; the latter explores alternative expressions to obtain a stronger watermark signal under preservation constraints. We release an open-source toolkit that integrates watermark embedding, quality auditing, and detection to support the protection of agent skills. SkillMarker targets ownership verification and leakage tracing in distributed skill ecosystems, where content may be copied in whole or in part, or incorporated into other skills.

1. Introduction

Agent skills package domain expertise into reusable instructions and resources that agents load on demand to specialize their behavior without modifying model weights (Li et al., 2026a; Zhang et al., 2025). This encoded knowledge makes skills valuable assets for organizations and developers, while their reusable format allows that expertise to be deployed across tasks. However, distributing skill instructions in plaintext makes them easy to copy, redistribute, or incorporate into other skills.

These risks motivate watermarking skill text before distribution to support ownership verification and leakage tracing, including when only fragments of the original skill remain. We therefore ask: **Can we watermark the skill for ownership verification and leakage tracing without compromising its functionality?**

However, embedding a watermark into structured, compact skill instructions with stringent accuracy requirements poses two coupled challenges. **C1: Preserving skill functionality during watermarking.** Skill instructions encode precise operational requirements, so even minor changes to resource references, decision conditions, or required actions can change agent behavior. Their structured format and stringent accu-

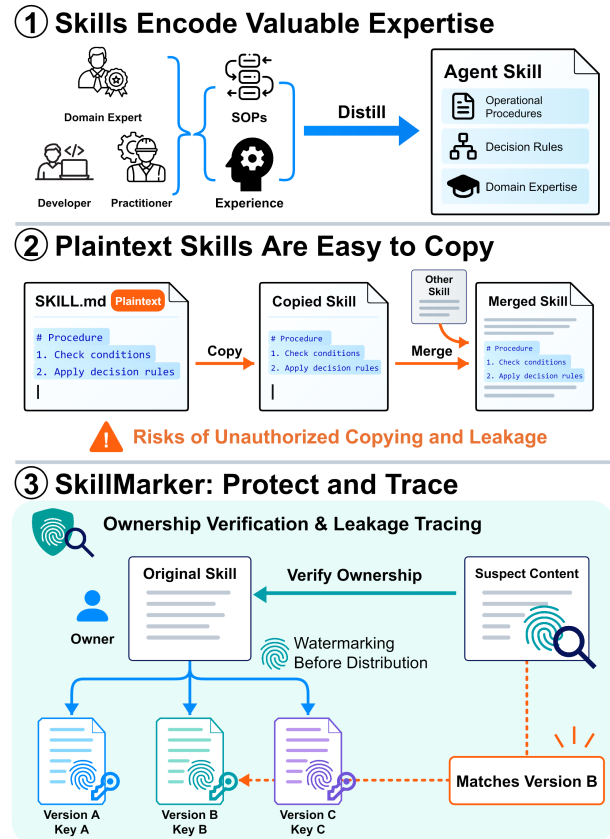


Figure 1 | Plaintext skills expose valuable domain expertise to unauthorized copying and leakage. SkillMarker offers ownership verification and leakage tracing.

*Corresponding author. We provide a watermarking platform: <https://alantang7-skillmark.hf.space/>

racy requirements leave limited room for watermark embedding through rewriting. **C2: Embedding a strong watermark under limited rewriting freedom.** Compact skill instructions offer few opportunities to strengthen the watermark signal. Accuracy constraints further restrict alternative expressions: a weak logit bias may produce little watermark signal, while a stronger bias can change essential instructions.

We introduce **SkillMarker**, a KGW-based watermarking method for agent skill ownership verification (Kirchenbauer et al., 2023). To address C1, we propose *quality-constrained rewriting*, which preserves protected content and constrains editable spans to preserve essential operational requirements. To address C2, we propose *signal-oriented candidate search*, which explores alternative expressions and selects candidates with stronger watermark signals among those that pass preservation checks.

SkillMarker enables developers to **verify skill ownership** by embedding a keyed watermark before publishing a skill and later testing suspect text for a watermark signal. It also enables them to **trace skill leakage** by assigning distinct keys to distributed versions and testing suspect text for a watermark signal that can help identify its source version, including when instructions are copied in whole or incorporated into other skills. To support these applications, we also develop a toolkit that integrates watermark embedding, quality auditing, and detection.

In summary, our contributions are threefold:

- We propose SkillMarker, a KGW-based method that combines functionality preservation and strong watermark embedding for skill ownership verification and leakage tracing.
- We study skill watermark performance under copy-paste and skill-merging scenarios, examining how these operations affect the watermark signal in reused content.
- We develop an integrated toolkit for watermark embedding, quality auditing, and detection, supporting the assessment of both watermark signal strength and instruction fidelity.

2. Method

2.1. Problem Setup and Overview

Let x denote the Markdown body of an original skill. We introduce a watermark key k for watermark embedding and detection. We seek a watermarked skill $\tilde{x}$ that preserves the instructions in x while carrying a watermark signal associated with k . A detector later tests suspect text for this signal, including when only part of the distributed skill is present. Watermark embedding uses a language model with access to its output logits; detection requires the tokenizer and watermark configuration, but no model inference.

For a vocabulary $\mathcal{V}$, write a token sequence as $w_{1:n}$. At generation step t , the input context and preceding generated tokens determine a next-token distribution P_t through a softmax over the model’s output logits. The watermark key k and preceding token w_{t-1} determine a green list G_t and its complement, the red list $R_t = \mathcal{V} \setminus G_t$; $\gamma \in (0, 1)$ controls the nominal green-list fraction, and $\delta > 0$ controls the logit bias. We use Q_t for the resulting watermarked distribution. The set $E \subset \mathcal{V}$ contains tokens exempt from logit bias and scoring. For any scored text, $\mathcal{I}$ denotes its eligible token positions, $N = |\mathcal{I}|$ their count, and G their green-token count.

Figure 2 provides an overview of SkillMarker. We first separate protected content from editable spans and generate candidate rewrites under preservation constraints. We assess watermark signal strength using the green-token rate and linguistic fluency using perplexity (PPL) (Bengio et al., 2003). Among candidates that pass preservation checks, we select those with higher green-token rates, using lower PPL to break ties. We then reconstruct the watermarked skill with protected content unchanged. The toolkit provides independent quality auditing and supports both full-text and window-based watermark detection.

2.2. Quality-Constrained Rewriting

Protecting literal content. To embed a watermark, we apply a logit bias that favors green-list tokens during rewriting. Without preservation constraints, rewriting can change executable code and invalidate a skill. For example, changing `>=`

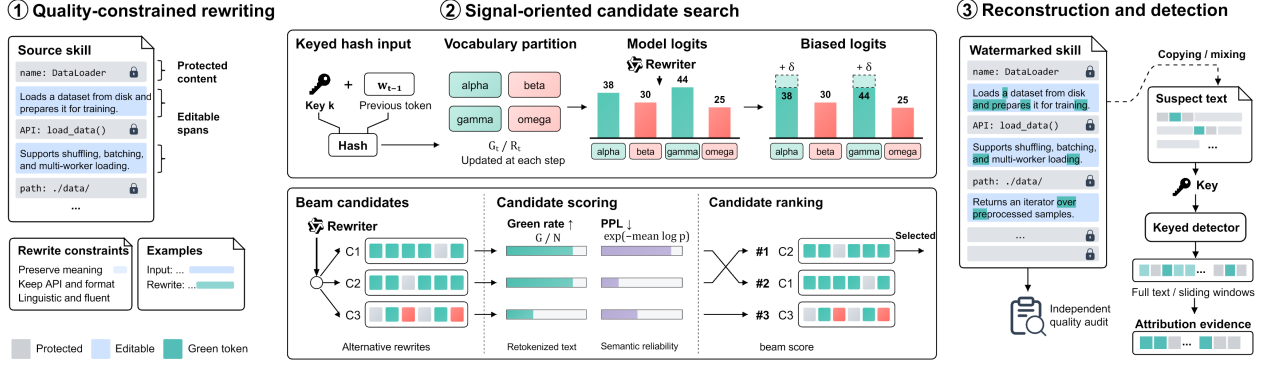


Figure 2 | Overview of SkillMarker. ① Quality-constrained rewriting preserves protected content, such as code and API identifiers, while rewriting editable spans. ② Signal-oriented candidate search ranks candidate rewrites by the green-token rate and uses lower perplexity to break ties. ③ Reconstruction preserves protected content, followed by independent quality auditing and detection in suspect text.

to `>` in the following code block changes when the retry limit is enforced:

```
if attempts >= 3:
    raise RuntimeError
```

Code and numerical thresholds allow little rewriting freedom, whereas surrounding prose often allows alternative expressions. We therefore parse the original skill and designate code, structural delimiters, and recognized literals as protected content that must remain unchanged. Regions with unresolved structure are also protected. We split the remaining prose into editable spans and protect tokens that cross the boundary of protected content. Editable spans below a threshold on the original token count remain unchanged. We rewrite one editable span at a time and then reconstruct the skill with protected content in its original order.

Preserving instruction fidelity. Preserving protected content alone does not preserve a decision rule: changing `all conditions` to `any condition` changes behavior even if every threshold remains unchanged. We use in-context learning to make preservation constraints within editable spans concrete. The rewriting prompt combines explicit preservation instructions with paired input and rewrite examples. These examples show how to change wording while preserving the original language, facts, conditions, negations, numerical values, and technical identifiers. Preservation checks reject incomplete or empty candidates, unauthorized layout or syn-

tax changes, and changes to protected content. Independent quality auditing then assesses language, semantics, completeness, and usability without exposing watermark scores to the reviewers. Downstream task performance is evaluated separately.

Grouping token variants. Watermark embedding can favor token variants that disrupt formatting. For example, a hard watermark that blocks `name` but permits `_name` can produce an unwanted underscore and change Markdown formatting. A logit bias can likewise favor a token variant without excluding the original token. We therefore assign consistent green-list membership to token variants. Our implementation groups tokens that differ only in leading spaces or periods and assigns them a shared representative’s green-list membership; literal-underscore variants remain subject to preservation checks. Specifically, we sample a keyed subset of $|\gamma|V|$ vocabulary IDs and assign each representative’s membership to all tokens in its group. The resulting green-list size need not equal $|\gamma|V|$, so γ remains a nominal parameter.

Exempting formatting tokens. Applying a logit bias to Markdown symbols can favor formatting changes that disrupt skill structure. We therefore exclude designated formatting tokens from the logit bias. Together with special and reserved control tokens, they form E . Following soft KGW (Kirchenbauer et al., 2023), we modify

the remaining logits by

$$Q_t(v) = \frac{P_t(v) \exp(\delta \mathbf{1}\{v \in G_t \setminus E\})}{\sum_{u \in \mathcal{V}} P_t(u) \exp(\delta \mathbf{1}\{u \in G_t \setminus E\})}. \quad (1)$$

Tokens in R_t remain available, allowing the model to select strongly preferred expressions. Exempt tokens receive no logit bias and contribute no watermark signal.

2.3. Signal-Oriented Candidate Search

Exploring alternative expressions. Strict preservation constraints can encourage copying the original text without changes, leaving little room for additional watermark signal. Increasing δ alone may instead favor an inaccurate expression. We therefore use beam search (Sutskever et al., 2014) to explore alternative expressions under the same preservation constraints. It selects likely continuations under Q_t in Eq. (1), producing completed candidates for each editable span.

Selecting on reconstructed text. A candidate’s generated token sequence can change when its text is joined to adjacent content. Scoring that sequence alone may therefore favor a watermark signal absent from the watermarked skill. We insert each candidate into the current skill and tokenize the complete reconstructed text. For the current editable span, $\mathcal{I}$ contains only nonexempt tokens wholly inside its output boundaries and having a preceding token. Each token’s green-list membership is determined using its actual preceding token, even when that token is protected or exempt. The candidate’s green-token rate is

$$r = \frac{G}{N}, \quad G = \sum_{t \in \mathcal{I}} \mathbf{1}\{w_t \in G_t\}. \quad (2)$$

We compute conditional PPL over each candidate’s generated tokens under the rewriting prompt to assess linguistic fluency. PPL is the exponential of the mean token negative log-likelihood under P_t .

For editable span i , let C_i contain candidates that pass the preservation checks and have $N > 0$. We select

$$C_i^* \in \arg \max_{C \in C_i} (r(C), -\text{PPL}(C)), \quad (3)$$

using lexicographic order: higher green-token rates take priority, and lower PPL breaks ties. If C_i is empty, we preserve the original editable span and record a fallback. We process editable spans in their original order and score the saved watermarked skill after reconstruction. This local selection does not guarantee the highest green-token rate for the complete skill.

2.4. Watermark Detection

Suspect text may contain copied instructions mixed with unrelated content, and the positions of protected content may be unavailable. We therefore support both full-text and window-based detection using only the suspect text and watermark configuration. Window-based detection uses sliding windows to detect watermark signals within suspect text. The detector tokenizes the full suspect text once and reconstructs G_t from k and each observed preceding token, using the same token-variant grouping as watermark embedding. It excludes E and the first token. Without the positions of protected content, all remaining positions are eligible, including content that may have been protected during watermark embedding.

For $N > 0$, we report the green-token rate G/N and the KGW statistic

$$z = \frac{G - \gamma N}{\sqrt{N\gamma(1-\gamma)}}. \quad (4)$$

The detector computes the nominal upper-tail probability $\Pr[\text{Binomial}(N, \gamma) \geq G]$. For window-based detection, it scores overlapping windows of the same tokenization, preserving each token’s preceding token in the full suspect text, and applies a Bonferroni correction across the scanned windows. Repeated contexts and shared green-list membership can violate the binomial null assumptions; these probabilities therefore require independent calibration before being interpreted as empirical false-positive guarantees.

3. Experiments

3.1. Experimental Setup

We report updated SkillMarker results on four datasets spanning everyday/general and professional scenarios. For everyday/general scenarios,

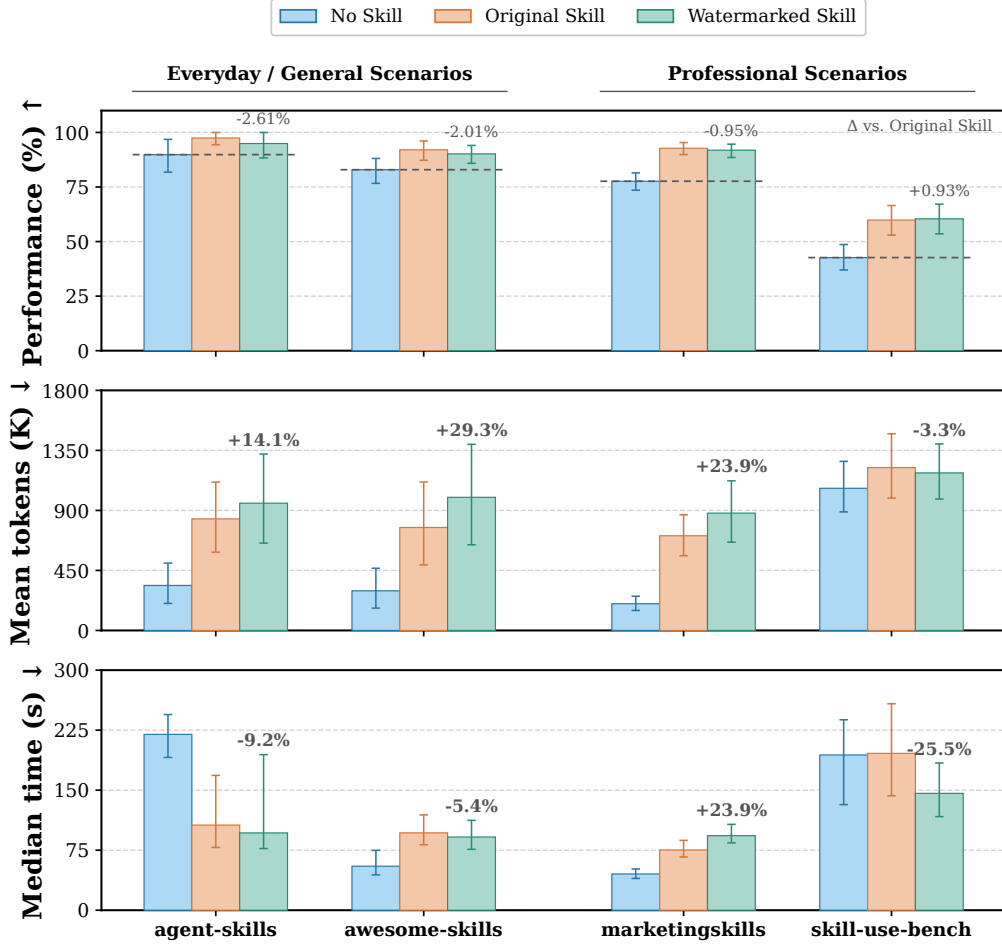


Figure 3 | **Watermarking largely preserves downstream task performance.** Downstream task performance, token usage, and runtime with no skill, original skills, and watermarked skills in everyday/general and professional scenarios.

agent-skills (Osmani, 2026) covers routine software engineering workflows, including planning, code review, testing, and debugging, while awesome-skills (sickn33, 2026) provides a broader collection for writing, document processing, information retrieval, and data analysis. For professional scenarios, marketingskills (Haines, 2026) focuses on marketing workflows such as content strategy, search engine optimization, advertising, and conversion optimization; skill-use-bench (Han et al., 2026) provides executable tasks requiring specialized procedures, including software development, document processing, and scientific computation. We compare three settings: no skill, the original skill, and the watermarked skill. The updated runs use DeepSeek-V4-Flash through Open Code, with one DeepSeek-V4-Flash judgment per case. To reduce evaluation leakage and cross-setting con-

tamination, we remove expected outputs from agent prompts and check each prompt for their absence; expected outputs remain available only to the judge. Grading files are kept off disk during agent execution and restored temporarily for subsequent scoring. We identify copies of the target skills using content fingerprints, remove them and their supporting resources from the evaluation environment, and restore only the skill version required by the current setting. Original and watermarked versions are never exposed simultaneously, and automatic skill discovery is disabled for the no-skill baseline. Each case runs in a separate temporary working directory, which is cleaned afterward to prevent artifacts from carrying over between cases. Execution traces are audited for unintended target-skill access, and contaminated baseline runs are rerun after cleanup.

3.2. Downstream Performance and Resource Usage

Figure 3 reports pooled check pass rates, mean total token usage, and median runtime from the updated three-arm evaluation. Token counts include input, cached input, and output ($K = 1000$). Bars use a common zero baseline. Short dashed lines mark each dataset’s no-skill performance. Labels on watermarked results show changes relative to the original skill before watermarking, expressed as percent changes for all three metrics. All relative-change labels use neutral gray. Error bars show 95% percentile confidence intervals from 10,000 bootstrap replicates that resample skills within each dataset, retaining all their available cases and using the same skill draws across settings. These intervals reflect heterogeneity among the saved skills and cases, rather than variability from repeated agent runs or repeated judgments. For *skill-use-bench*, we use 142 scored tasks per setting, exclude two skipped tasks, and aggregate resource usage over available records.

Watermarking largely preserves downstream task performance: watermarked skills change performance by -2.61% to $+0.93\%$ relative to original skills across four datasets and retain their gains over the no-skill baseline. On *skill-use-bench*, pass rates are 59.85% before watermarking and 60.41% afterward. Original skills improve performance over no-skill runs by 8.49–40.30%. Mean total tokens increase by 14.1–29.3% on the first three datasets but decrease by 3.3% on *skill-use-bench*. These means are sensitive to the long-tailed distribution of run token counts. Median runtime decreases for *agent-skills*, *awesome-skills*, and *skill-use-bench*, and increases for *marketingskills*. Runtime is reported for reference, as machine and service load, execution periods, and concurrency can affect wall-clock measurements. Our assessment of functionality preservation relies on task performance rather than these timing differences. The absolute values are provided in Table 1 in Appendix A.

3.3. Watermark Robustness

We consider two skill-reuse scenarios: copying watermarked content into an unwatermarked text (copy-paste) and combining multiple skills (skill merging). We report detection under both scenarios and localization under copy-paste.

3.3.1. Copy-Paste

Setup. We select three skills, from each source, we copy 256, 512, 1024, 1536, or 2048 tokens into an unwatermarked text containing 1024 or 4096 tokens before insertion. We vary the number of inserted fragments over $\{1, 2, 4, 8\}$. Within each condition, fragmentation splits the same copied content while preserving its total length and order. Three fixed seeds per source control the selection of copied fragments and insertion positions, yielding 360 watermarked documents and 360 paired unwatermarked controls.

Detection and localization. We scan each saved and re-tokenized document without access to the true insertion positions, using 1024-token windows with a stride of 256 tokens and a per-window significance threshold of $p \leq 0.01$. To evaluate localization, we merge all significant windows into the predicted region A and take the union of the inserted fragments as the true region T . Overlapping windows count each token position only once. Localization precision and recall are

$$\text{Precision} = \frac{|A \cap T|}{|A|}, \quad \text{Recall} = \frac{|A \cap T|}{|T|}. \quad (5)$$

When no window is significant, recall is zero and precision is undefined. Each plotted condition includes three seeds for each of the three sources: we first average within each source and then average the three sources with equal weight. Precision uses only runs with valid predictions; if any source has no valid prediction, the corresponding cell is marked N/A.

Results. Figure 4 compares full-text and window-based detection. Longer copied fragments generally increase detection rates, while window-based detection helps detect watermark signals diluted by the unwatermarked text. For

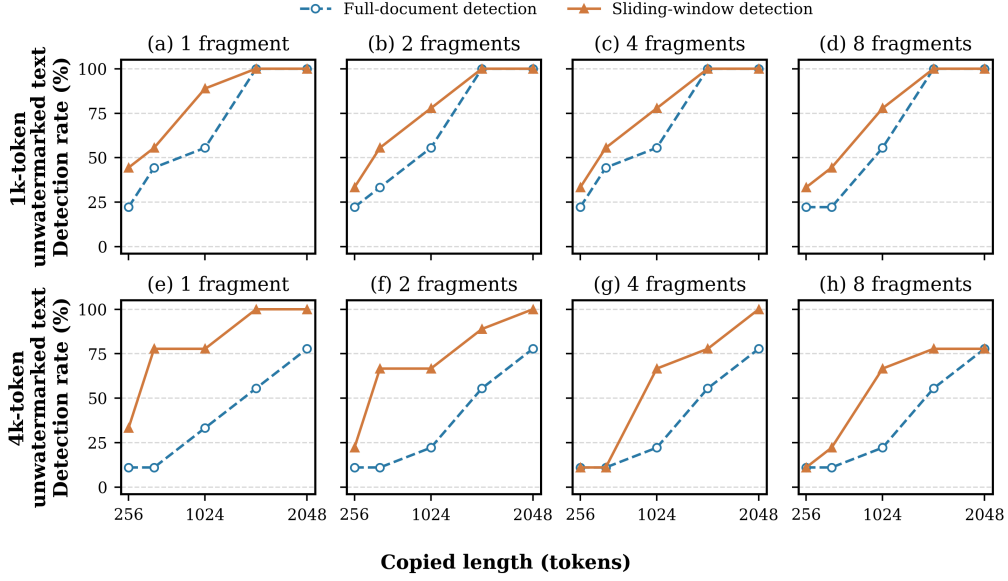


Figure 4 | **Window-based detection improves detection of diluted watermark signals, while fragmentation weakens its advantage.** Full-text and window-based watermark detection after copying skill content into unwatermarked text with varying copied lengths and fragment counts.

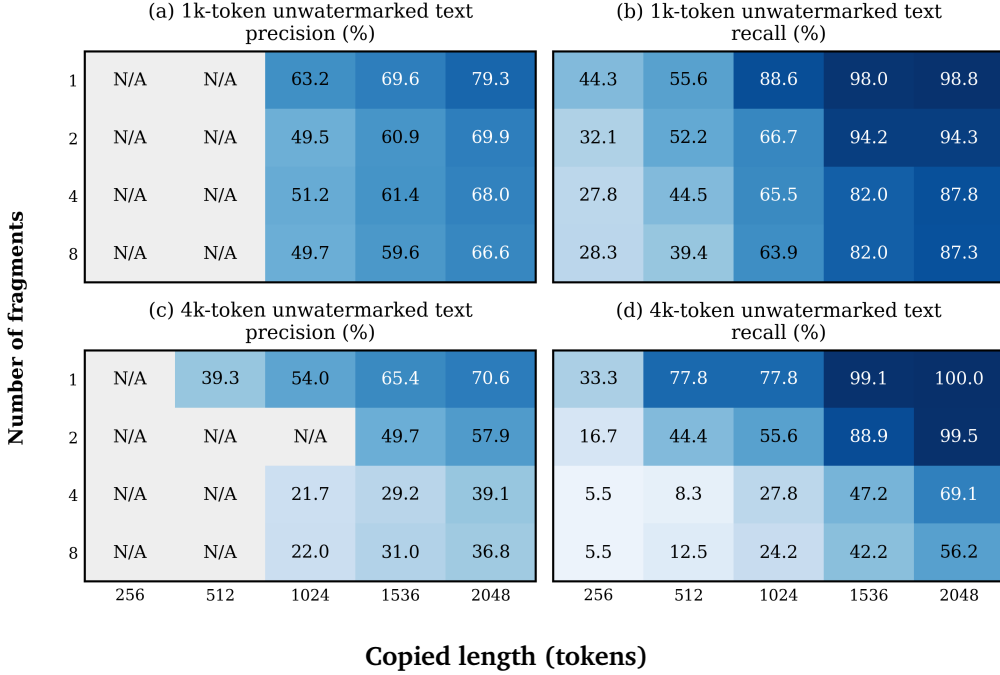


Figure 5 | **Longer copied fragments generally improve localization recall, whereas fragmentation reduces localization quality.** Token-level localization precision and recall after copying skill content into unwatermarked text with varying copied lengths and fragment counts.

example, when 2048 tokens are inserted contiguously into a 4096-token unwatermarked text, window-based detection reaches 100%, compared with 77.8% for full-text detection. Splitting the same content into eight fragments reduces window-based detection to 77.8%. Figure 5 shows that longer copied fragments also generally

improve localization recall, whereas fragmentation reduces the localized fraction of the inserted content. For the same 2048-token copy into a 4096-token unwatermarked text, contiguous insertion yields 70.6% precision and 100% recall; eight fragments yield 36.8% precision and 56.2% recall. These results demonstrate detection and

localization of copied content for the selected skills with strong watermark signals. Dispersed fragments can remain detectable even when they are only partially localized.

3.3.2. Skill Merging

Figure 6 shows that watermark evidence remains detectable after merging a target skill with a limited number of background skills. With one, two, or four total input skills, multi-scale union detection achieves detection rates of 91.7%, 72.2%, and 77.8%, respectively, compared with 58.3%, 47.2%, and 44.4% for single-window detection. Adding background skills reduces detection relative to the single-input setting, consistent with watermark dilution by unwatermarked content. This decline is not strictly monotonic for multi-scale detection, but both detectors retain lower detection rates with background skills than without them. Multi-scale detection helps recover some of the remaining evidence under this dilution. Both methods have the same observed false-positive rates on unwatermarked controls: 0.0%, 11.1%, and 16.7% for one, two, and four input skills. The increasing false-positive rates indicate that reliable detection under greater background noise also requires calibration.

3.4. Structured Skill Case Study

Figure 7 compares selected excerpts from observability-and-instrumentation, watermarked with Qwen3-8B using an English prompt and default protection. Protected spans remain exactly unchanged, and the saved full text yields nominal $p = 4.62 \times 10^{-9}$. Long protected blocks show only their opening lines; omission marks separate non-contiguous excerpts. Original generation defects are retained, so this case illustrates literal preservation and detection, without establishing semantic or functional equivalence.

4. Related Work

4.1. Watermarking

Generation-time watermarking. KGW and Unigram-Watermark bias token selection using context-dependent and fixed vocabulary parti-

tions, respectively (Kirchenbauer et al., 2023; Zhao et al., 2024). Other approaches use tournament sampling (SynthID-Text), sentence-level rejection sampling (SemStamp), or optimal transport (textGrain) (Dathathri et al., 2024; Hou et al., 2024; Li et al., 2026b). SWEET and STONE protect low-entropy positions and syntactic code tokens, respectively (Kim et al., 2026; Lee et al., 2024), while AgentMark watermarks planning decisions (Huang et al., 2026). Generation-time watermarking is typically deployed on the model-provider side to attribute generated outputs. SkillMarker instead enables skill developers to embed watermarks into reusable instructions before distribution, independently of the models that later execute them.

Post-hoc watermarking. PostMark inserts semantically selected words, REMARK-LLM encodes binary signatures, and SrcMarker applies semantics-preserving code transformations (Chang et al., 2024; Yang et al., 2024; Zhang et al., 2024). RedAct watermarks released agent traces through selective rewriting and behavioral signals (Xu et al., 2026); SkillTrace audits skill reuse through expression, implementation, and operational traces (Chen et al., 2026). These methods establish watermarking and provenance auditing for existing artifacts, but semantic preservation in general text or released traces does not by itself establish instruction fidelity in skills. Skill protection must preserve operational requirements while embedding watermarks that provide sufficient evidence for attribution after reuse. SkillMarker addresses this joint requirement in distributed skill text; reliability studies of copy-paste attacks (Kirchenbauer et al., 2024) motivate its evaluation under copy-paste and skill merging.

4.2. Text Steganography

Neural Linguistic Steganography conceals recoverable messages through arithmetic coding (Ziegler et al., 2019); zero-width characters offer invisible encoding but must survive redistribution. ICW induces output patterns through instructions, including covert triggers in modified documents (Liu et al., 2026). Its signal is carried by subsequent model outputs, whereas SkillMarker embeds watermarks directly into the

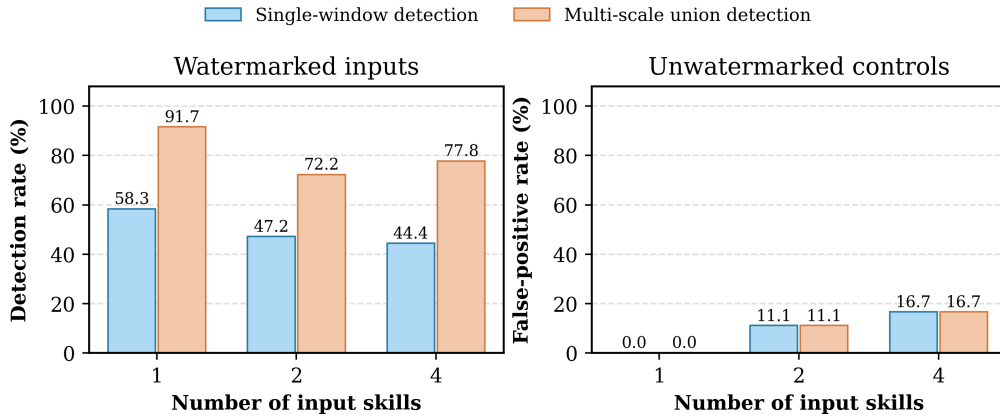


Figure 6 | **Watermarks remain detectable when merged with a limited number of background skills, although added background content weakens detection overall.** Detection rates for watermarked inputs and false-positive rates for unwatermarked controls when merging one, two, or four input skills.

Original skill		Watermarked skill	
<pre> typescript // BAD: string interpolation - unqueryable, inconsistent [...] As with tracing, the vendor-neutral path is the OpenTelemetry metrics API (same SDK and context as step 5). The example below uses Prometheus' `prom-client` - one common backend choice, not the only one; the RED/USE and cardinality rules are identical either way. [...] Rule 2 above requires every alert to link to a runbook. A runbook's job is to answer three questions without requiring the reader to think: what is happening, what to check first, and who to call if that doesn't resolve it. Store in `docs/runbooks/` named after the alert. </pre>		<pre> typescript // BAD: string interpolation - unqueryable, inconsistent [...] As with tracing, the vendor-neutralthe path is the OpenTelemetry metrics API (same SDK and context as step 5). The example below uses Prometheus' `prom-client` - One commonly used backend choice, but not the only one; the RED/USE and cardinality rules are identical either way. [...] Rule 2 aboveEvery alert must link to a runbook. A runbook's purpose is to answer three questions without requiring the reader to think: what is happening, what to check first, and whom to contact if that does not resolve it. Store in `docs/runbooks/` named after the alert. </pre>	

Figure 7 | **Protected text is preserved while editable prose carries watermark evidence.** Original and watermarked excerpts from observability-and-instrumentation.

distributed skill. Linguistic steganography focuses on concealed message recovery. Neither objective by itself establishes attribution of copied or merged skill instructions under strict functionality constraints. SkillMarker instead supports ownership verification and leakage tracing through statistical watermark signals under preservation constraints.

5. Conclusion

We present SkillMarker, a KGW-based method for protecting distributed agent skills through quality-constrained rewriting and signal-oriented candidate search. Across four datasets, watermarked skills largely preserve downstream per-

formance, with relative changes of -2.61% to $+0.93\%$ and dataset-dependent token overhead. Window-based detection improves detection of copied content diluted by unwatermarked text, while fragmentation weakens detection and localization. Watermarks remain detectable after merging with a limited number of background skills, although added background content weakens detection overall and elevated control false positives highlight the need for calibration. These findings support watermarking as a practical direction for skill ownership verification and leakage tracing.

References

- Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003. URL <https://www.jmlr.org/papers/v3/bengio03a.html>.
- Y. Chang, K. Krishna, A. Houmansadr, J. F. Wieting, and M. Iyyer. PostMark: A robust blackbox watermark for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8969–8987. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.506. URL <https://aclanthology.org/2024.emnlp-main.506/>.
- J. Chen, M. Wang, L. Jiang, J. Ma, X. Deng, X. Du, R. Lin, Y. Feng, L. Du, and J. Wang. SkillTrace: Multi-trace provenance auditing for LLM-agent skill reuse. arXiv preprint arXiv:2608.05204, 2026. URL <https://arxiv.org/abs/2608.05204v2>. Version 2.
- S. Dathathri, A. See, S. Ghaisas, P.-S. Huang, R. McAdam, J. Welbl, V. Bachani, A. Kaskasoli, R. Stanforth, T. Matejovicova, J. Hayes, N. Vyas, M. Al Merey, J. Brown-Cohen, R. Bunel, B. Balle, T. Cemgil, Z. Ahmed, K. Stacpoole, I. Shumailov, C. Baetu, S. Gowal, D. Hassabis, and P. Kohli. Scalable watermarking for identifying large language model outputs. *Nature*, 634:818–823, 2024. doi: 10.1038/s41586-024-08025-4. URL <https://www.nature.com/articles/s41586-024-08025-4>.
- C. Haines. Marketing skills. GitHub repository, 2026. URL <https://github.com/coreyhaines31/marketingskills>. Accessed October 9, 2026.
- J. Han, Y. Xu, Y. Liao, X. Wang, Z. Jiang, Z. Di, F. Lu, Z. Hu, and Y. Xiao. Skill-use: Can LLMs actually use skills in agentic harnesses? arXiv preprint arXiv:2608.04828, 2026. URL <https://arxiv.org/abs/2608.04828>.
- A. Hou, J. Zhang, T. He, Y. Wang, Y.-S. Chuang, H. Wang, L. Shen, B. Van Durme, D. Khashabi, and Y. Tsvetkov. SemStamp: A semantic watermark with paraphrastic robustness for text generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4067–4082. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.naacl-long.226. URL <https://aclanthology.org/2024.naacl-long.226/>.
- K. Huang, J. Tan, Y. Wei, W. Li, Z. Zhang, H. Tian, Z. Yang, and L. Zhou. AgentMark: Utility-preserving behavioral watermarking for agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12581–12603. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.573. URL <https://aclanthology.org/2026.acl-long.573/>.
- J. Kim, S. Park, and Y.-S. Han. Marking code without breaking it: Code watermarking for detecting LLM-generated code. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 3990–4002. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-eacl.207. URL <https://aclanthology.org/2026.findings-eacl.207/>.
- J. Kirchenbauer, J. Geiping, Y. Wen, J. Katz, I. Miers, and T. Goldstein. A watermark for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 17061–17084. PMLR, 2023. URL <https://proceedings.mlr.press/v202/kirchenbauer23a.html>.
- J. Kirchenbauer, J. Geiping, Y. Wen, M. Shu, K. Saifullah, K. Kong, K. Fernando, A. Saha, M. Goldblum, and T. Goldstein. On the reliability of watermarks for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://arxiv.org/abs/2306.04634>.
- T. Lee, S. Hong, J. Ahn, I. Hong, H. Lee, S. Yun, J. Shin, and G. Kim. Who wrote this code? watermarking for code generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4890–4911. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.268. URL <https://aclanthology.org/2024.acl-long.268/>.
- X. Li, Y. Liu, W. Chen, B. You, Z. Di, Y. He, S. Zheng, K. W. Choe, J. Sun, S. Wang, C. Tao, B. Li, X. Zhao, H. Geng, X. Wu, J. Zhou, X. Chen, H. Xing, Y. Li, Q. Zeng, D. Wang, Y. Wang, R. B. Chaim, P. Jiang, H. Shen, L. Kong, X. Liu, R. Wang, X. Liu, J. Li, X. Lan, Y. Lin, W. Ye, J. He, S. Li, Y. Zhang, Y. Gao, Y. Li, Z. Ma, L. Jing, T. Wang, K. Li, Y. Xue, H. Lyu, Y. He, Y. Tian, S. Wu, B. Wang, Y. Gao, B. Chen, L. Liu, S. Cheng, J. Bao, S. Tong, S. Xu, T. Y. Zhuo, T. Ye, Q. Qi, M. Li, L. Liao, Z. Tan, C. Shi, X. Tang, S. Tankasala, B. Yuan, Y. Qian, J. Tu, C. Wang, Y. Sun, W. Wang, A. Taylor, Z. Yang, C. Guan, Z. Dong, X. Zhang, S. Dillmann, H.-c. Lee, and D. Song. Skillsbench: Benchmarking how well agent skills work across diverse tasks. arXiv preprint arXiv:2602.12670, 2026a.
- X. Li, G. Wen, X. Chen, Q. Long, A. Jain, F. Joly, M. Lam, Q. Song, and W. Su. textGrain: Entropy-

- calibrated watermarking for language model text. Technical report, OpenAI, Oct. 2026b. URL <https://cdn.openai.com/pdf/e9508624-d767-41b6-a26d-e34ca798ada6/textgrain-entropy-calibrated-watermarking-for-language-model-text.pdf>. Technical report, October 5, 2026.
- Y. Liu, X. Zhao, C. Kruegel, D. Song, and Y. Bu. In-context watermarks for large language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/571082ea18d30060177dfcaf662ff0e5-Abstract-Conference.html.
- A. Osmani. Agent skills. GitHub repository, 2026. URL <https://github.com/addyosmani/agent-skills>. Accessed October 9, 2026.
- sickn33. Agentic awesome skills. GitHub repository, 2026. URL <https://github.com/sickn33/agentic-awesome-skills>. Accessed October 9, 2026.
- I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27, 2014. URL https://papers.nips.cc/paper_files/paper/2014/hash/5a18e133cbf9f257297f410bb7eca942-Abstract.html.
- S. Xu, Z. He, and Y. R. Fung. RedAct: Redacting agent capability traces for procedural skill protection. arXiv preprint arXiv:2606.10813, 2026. URL <https://arxiv.org/abs/2606.10813v3>. Version 3.
- B. Yang, W. Li, L. Xiang, and B. Li. SrcMarker: Dual-channel source code watermarking via scalable code transformations. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024. doi: 10.1109/SP54263.2024.00097. URL <https://ieeexplore.ieee.org/abstract/document/10646683/>.
- B. Zhang, K. Lazuka, and M. Murag. Equipping agents for the real world with agent skills. Anthropic Engineering, 2025. URL <https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills>. Published October 16, 2025; accessed October 8, 2026.
- R. Zhang, S. S. Hussain, P. Neekhara, and F. Koushanfar. REMARK-LLM: A robust and efficient watermarking framework for generative large language models. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 1813–1830. USENIX Association, 2024. URL <https://www.usenix.org/conference/usenixsecurity24/presentation/zhang-ruisi>.
- X. Zhao, P. Ananth, L. Li, and Y.-X. Wang. Provable robust watermarking for AI-generated text. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://arxiv.org/abs/2306.17439>.
- Z. Ziegler, Y. Deng, and A. Rush. Neural linguistic steganography. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1210–1215. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1115. URL <https://aclanthology.org/D19-1115/>.

A. Additional Experimental Details

A.1. Absolute Downstream Results

Table 1 reports the absolute values underlying Figure 3 for the same settings and datasets. Table 1 | Updated downstream task performance and resource usage with no skills, original skills, and watermarked skills across everyday/general and professional scenarios.

Setting	Everyday / General Scenarios						Professional Scenarios					
	agent-skills			awesome-skills			marketingskills			skill-use-bench		
	Passed			Passed			Passed			Passed		
	checks ↑ (%)	Tokens ↓ (K)	Time ↓ (s)	checks ↑ (%)	Tokens ↓ (K)	Time ↓ (s)	checks ↑ (%)	Tokens ↓ (K)	Time ↓ (s)	checks ↑ (%)	Tokens ↓ (K)	Time ↓ (s)
No Skill	89.83	337.5	219.7	82.93	297.3	55.0	77.64	200.3	45.3	42.66	1066.0	194.0
Original Skill	97.46	836.6	106.4	92.03	772.1	96.7	92.75	709.8	75.2	59.85	1221.3	196.0
Watermarked Skill	94.92	954.3	96.6	90.18	998.3	91.5	91.86	879.6	93.2	60.41	1181.3	146.0

A.2. Main Experimental Hyperparameters

The main hyperparameters are listed below. Robustness experiments use the frozen English watermarked release; its embedding settings are listed separately from the updated downstream evaluation.

Experiment	Hyperparameter	Value
Downstream	Agent / judge	DeepSeek-V4-Flash [1m] / same model
	Judgments per case	1
	Bootstrap replicates / seed	10,000 / 20261010
Robustness embedding	Model / beam width	Qwen3-8B / 5
	Green fraction γ / logit bias δ	0.75 / 8
	Hash key / embedding seed	15485863 / 123
Copy-paste	Copied length (tokens)	256, 512, 1024, 1536, 2048
	Unwatermarked text length (tokens)	1024, 4096
	Number of fragments	1, 2, 4, 8
	Selection / insertion seeds	123, 456, 789
	Detection window / stride (tokens)	1024 / 256
	Per-window significance threshold	$p \leq 0.01$
Skill merging	Merging model / reasoning effort	deepseek-flash / high
	Maximum output tokens	65,536
	Total input skills	1, 2, 4 (one target plus 0, 1, 3 backgrounds)
	Background selection seeds	123, 456, 789
	Single-window length / stride (tokens)	3952 / 1027
	Single-window probability threshold	$p \leq 0.03332253$
	Multi-scale union	OR of 13 component decisions

Copy-paste window tests use nominal probabilities without multiple-testing correction. Merge detector parameters were selected on the same evaluation cohort, without a held-out test set.

B. Experimental Prompts

B.1. Watermark Embedding

The English embedding prompt below contains the system instruction and two user–assistant demonstrations. The final user message is replaced by the current editable passage. Role labels and the bracketed input field are presentation annotations; the remaining text is reproduced verbatim from the saved generation configuration.

Watermark embedding: English prompt with demonstrations

```
[System]
Preserve the original meaning, key facts, conditions, negations, numerical values, technical
identifiers, and formatting. You may freely rephrase the remaining wording. Output only the
rewritten passage in its original language.

[User: demonstration 1]
**Core principle**: Automatically write patterns that meet the automatic-write conditions
(frequency  $\geq 3$ , consistent fixes, certainty 100%) to skill references; otherwise, retain them
as a staged patch.

[Assistant: demonstration 1]
**Core principle**: Write patterns to skill references automatically only when all
automatic-write conditions are met (frequency  $\geq 3$ , consistent fixes, certainty 100%); keep
patterns that do not meet these conditions as a staged patch.

[User: demonstration 2]
Purely visual ALIGN differences in CHECK-8 and CHECK-11 (style compliance) do not block the
gate; record them in the report for later improvement. **CRASH/URL issues in CHECK-8 are
functional failures** (a page failing to open or crashing is not a "visual issue"); they block
the gate and enter the $5 loop.

[Assistant: demonstration 2]
Record purely visual ALIGN differences in CHECK-8 and CHECK-11 (style compliance) in the
report for future improvement; they do not block the gate. **CRASH/URL issues in CHECK-8 are
functional failures** (a page failing to open or crashing is not a "visual issue"), so they
block the gate and enter the $5 loop.

[User: actual input]
[CURRENT EDITABLE PASSAGE]
```

B.2. Skill-Merging Instruction

For the merging evaluation, the system message contains the instruction below; the user message contains the complete input skills.

Skill-merging evaluation prompt

```
[System]
Merge the provided skills into one skill.

[User]
[COMPLETE INPUT SKILLS IN THE RECORDED ORDER]
```

C. Extended Structured Skill Case Study

Figure 8 retains the earlier nine-excerpt presentation of observability-and-instrumentation, complementing the compact view in Figure 7. The skill was watermarked with Qwen3-8B using an English prompt and default protection. The saved mapping records exact preservation of protected spans and full-text nominal $p = 4.62 \times 10^{-9}$. Long protected blocks show their opening lines only; omission marks separate non-contiguous excerpts. Original generation defects are retained, so literal preservation and detection do not establish semantic or functional equivalence.

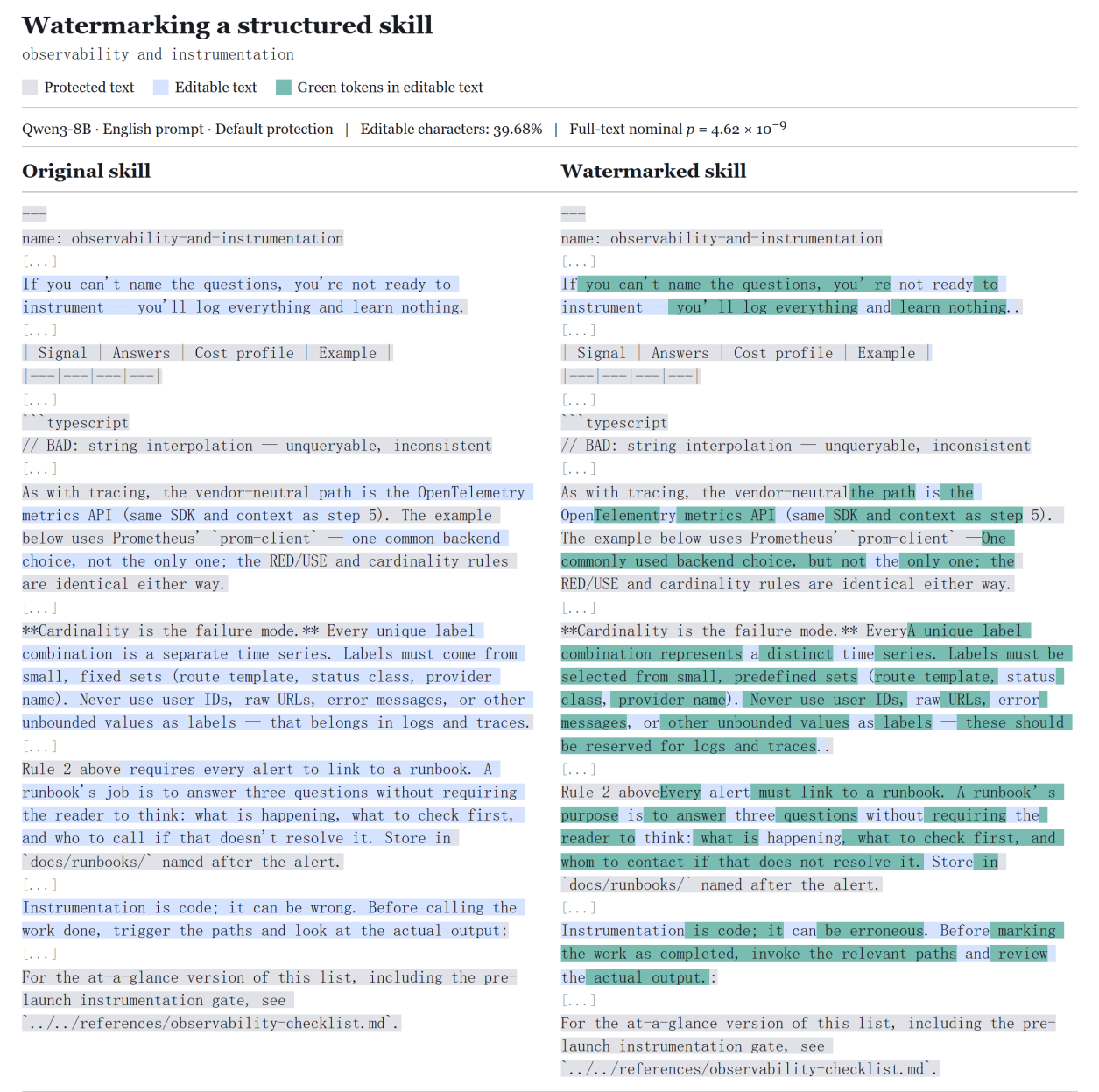


Figure 8 | Protected text is preserved while editable prose carries watermark evidence. Extended original and watermarked excerpts from observability-and-instrumentation.