IntegrityShield
A System for Ethical AI Use and Authorship Transparency in Assessments

Ashish Raj Shekhar[*] **Shiven Agarwal**[*] **Priyanuj Bordoloi** **Yash Shah**
Tejas Anvekar **Vivek Gupta**
Arizona State University
Project Page Demo Video Code
{ashekha9, sagar147, pbordolo, yshah124, tanvekar, vgupt140}@asu.edu

Abstract

Multimodal Large Language Models (MLLMs) can now solve entire exams directly from uploaded PDF assessments, raising urgent concerns about academic integrity and the reliability of grades and credentials. Existing watermarking techniques either operate at the token level or assume control over the model's decoding process, making them ineffective when students query proprietary black-box systems using instructor-provided documents. We present IntegrityShield, a document-layer watermarking system that embeds schema-aware, item-level watermarks into assessment PDFs while keeping their human-visible appearance unchanged. These watermarks consistently prevent MLLMs from answering shielded exam PDFs and encode stable, item-level signatures that can be reliably recovered from model or student responses. Across 30 question papers spanning STEM, humanities, and medical reasoning, IntegrityShield achieves exceptionally high prevention (91-94% exam-level blocking) and strong detection reliability (89-93% signature retrieval) across four commercial MLLMs. Our demo showcases an interactive interface where instructors upload an exam, preview watermark behavior, and inspect pre/post AI performance and authorship evidence.

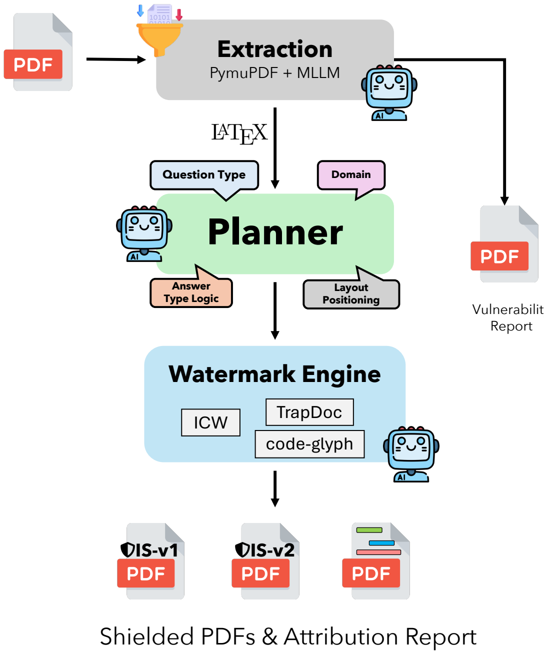


Figure 1: Overview of IntegrityShield. The system extracts question structure from an assessment PDF, uses an LLM-based planner to select schema-aware watermarking tactics, and applies document-layer perturbations through the watermark engine. It outputs shielded PDF variants (**IS-v1**, **IS-v2**) and an attribution report summarizing AI vulnerability along with authorship signals.

1 Introduction

LLMs and MLLMs can now interpret full PDF assessments, reason over diagrams and tables, and produce fluent step-by-step solutions within seconds. While these capabilities expand access to high-quality assistance, they simultaneously undermine the credibility of homework and online exams by enabling students to outsource entire assessments to AI tools (OpenAI, 2023; Team, 2024; Susnjak, 2022).

Institutions have responded with post-hoc detection (e.g., authorship classifiers (Emi and Spero, 2024; Thai et al., 2025)) and surveillance-heavy proctoring (e.g., keystroke, browser, or gaze monitoring (Atoum et al., 2017; Kundu et al., 2024)). However, detectors struggle with short answers, code, paraphrasing, and mixed authorship (Mitchell et al., 2023; Niu et al., 2024), while invasive monitoring raises significant privacy, accessibility, and equity concerns.

These approaches share a fundamental limitation: they analyze the *student's output*. In practice, the dominant workflow is the opposite students upload *instructor-provided PDFs* to black-box AI systems. Existing watermarking methods, which modify generation at the model's decoder (Kirchen-

[*] contributed equally

Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics
Volume 3: System Demonstrations, pages 417–427
March 24-29, 2026 ©2026 Association for Computational Linguistics

bauer et al., 2023; Liu et al., 2025), cannot operate in this setting. This motivates an interesting question: *Can assessments themselves be instrumented so that AI reliance becomes observable, without altering visible exam content or student workflows?*

From detection to document-level watermarking. We exploit the render-parse gap in PDFs: what humans see often differs from what AI parsers ingest. By injecting invisible text, glyph remappings, and lightweight overlays, we influence model interpretation while leaving the exam visually unchanged. IntegrityShield operationalizes this idea as an authorship-aware watermarking system. Rather than asking whether a student cheated, we ask: *to what extent do model-generated responses follow a consistent watermark signature embedded in the exam?* This reframing provides instructors with an interpretable notion of authorship degree while maintaining fairness for honest students. Finally, we summarize our contributions as:

- We introduce IntegrityShield, a document-layer watermarking system that embeds schema-aware watermarks into assessment PDFs while keeping their human-visible appearance unchanged.
- We develop an LLM-driven planner and PDF watermark engine that adapts tactics to question type,achieving consistently high prevention (91-94% exam-level blocking) and strong detection reliability (89-93% retrieval) across four commercial MLLMs on a thirty-exam benchmark.
- We release an interactive demo that allows instructors to upload exams, preview watermarks, and inspect pre/post AI performance and authorship evidence, enabling ethical and transparent AI use in education.

2 Background and Related Work

AI assistance and mixed authorship. LLMs increasingly participate in writing and problem-solving tasks, often producing blended human-AI content. Recent work formalizes this as *homogeneous* vs. *heterogeneous* mixed authorship (Thai et al., 2025). Existing detectors including perplexity-based methods (Mitchell et al., 2023), style-based classifiers (Emi and Spero, 2024), and multilingual cheating detectors (Niu et al., 2024) struggle with short answers, paraphrasing, and multi-author mixtures, and they analyze only the *output*, leaving the assessment itself uninstrumented.

AI watermarking. Watermarking embeds provenance signals into generated content, typically by modifying decoding distributions (Kirchenbauer et al., 2023) or via prompt-based in-context cues (Liu et al., 2025). Parallel work explores invisible watermarks for AI-generated writing, designed to survive paraphrasing and editing (Liu and Bu, 2024). These methods assume control over generation, which is infeasible when students query proprietary black-box systems using instructor-provided PDFs.

Document-layer perturbations. Recent work shows that perturbing the PDF substrate - via phantom tokens, font CMaps, or off-page text - can induce systematic model errors without affecting human readability (Jin et al., 2025; Xiong et al., 2025). Our work builds on these insights but shifts the objective: rather than deceiving models or detecting cheating, we embed *recoverable watermark signatures* that quantify the extent of AI involvement in solving an exam.

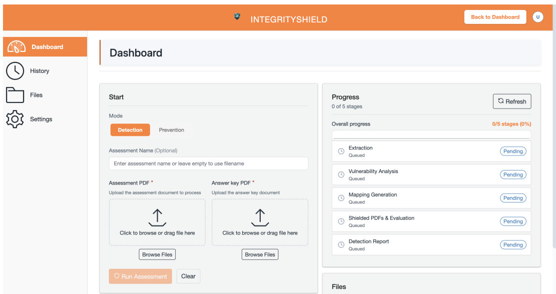
3 IntegrityShield System Architecture and Workflow

IntegrityShield is designed as a practical tool for instructors who want to harden PDF-based assessments against AI assistants without redesigning their exams or changing grading workflows. The system keeps all human-facing content (layout, typography, pagination, item numbering) unchanged while embedding signals that reliably influence model-side parsing. It adapts watermark tactics to the item schema, treating **MCQ**, **true/-false**, and **Long-Form** questions differently, remaining robust across black-box MLLMs; and exposes a lightweight web interface where instructors can upload assessments, preview watermark behavior, and inspect calibration and authorship signals with minimal configuration.

3.1 End-to-End Architecture and Workflow

Figure 1 summarizes the end-to-end workflow of IntegrityShield. A single-page web front end communicates with a stateless backend that operates directly on the PDF substrate. The backend is organized around four logical services: document ingestion, which parses the uploaded PDF into a structured item schema with stems, options, diagrams, and answer keys; strategy planning, via a lightweight instruction-tuned LLM that assigns a watermark plan to each item based on its type, gold answer, and local layout metadata; PDF rewriting, which applies the plan while enforcing that the rendered appearance of the document remains unchanged; and an authorship and calibration service that runs reference models on original and watermarked PDFs and later scores submitted answers against stored watermark signatures. From an instructor's perspective, this architecture is depicted in a three-stage interaction flow.

Stage 1: Upload and Watermark Planning. The instructor uploads an exam PDF through the browser. The ingestion service segments pages into questions, detects answer options and numbering, and associates inline figures or tables with the relevant items, producing a compact item schema with content spans, page coordinates, and answer keys when available, as illustrated in Figure 2.

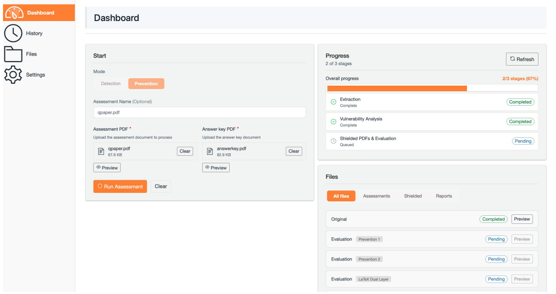


Extraction Queued Pending
Vulnerability Analysis Queued Pending
Mapping Generation Queued Pending
Shielded PDFs & Evaluation Queued Pending
Detection Report Queued Pending

Figure 2: Stage 1: Upload and Watermark Planning. Instructors upload an assessment PDF and answer key, after which the system extracts question structure and previews the planned schema-aware watermarking strategies.

The strategy planner then assigns, for each item, either a *target distractor* (for multiple-choice and true/false questions) or a small set of signature key phrases (for long-form questions) and decides which document-layer mechanisms to apply. The interface presents a split-screen preview of original and watermarked pages with per-question summaries of the chosen strategy, allowing instructors to inspect and optionally disable aggressive tactics (such as strong glyph remapping) before proceeding.

Stage 2: Watermark Embedding and AI Calibration. Once the plan is confirmed, the PDF rewriting service applies it directly to the assessment file. It injects invisible text spans anchored near stems and options, applies CMap-based glyph remapping so that visually identical tokens are parsed differently by models, and, when appropriate, adds clipped or off-page overlays that insert distractor-oriented cues into the parsable stream while keeping them outside the visible canvas, as shown in Figure 3.



Assessment Name (Optional): qpaper.pdf
qpaper.pdf 67.6 KB
answerkey.pdf 82.9 KB
2 of 3 stages
2/3 stages (67%)
Extraction Complete Completed
Vulnerability Analysis Complete Completed
Shielded PDFs & Evaluation Queued Pending
Original Completed Preview
Evaluation Prevention 1 Pending Preview
Evaluation Prevention 2 Pending Preview
Evaluation LaTeX Dual Layer Pending Preview

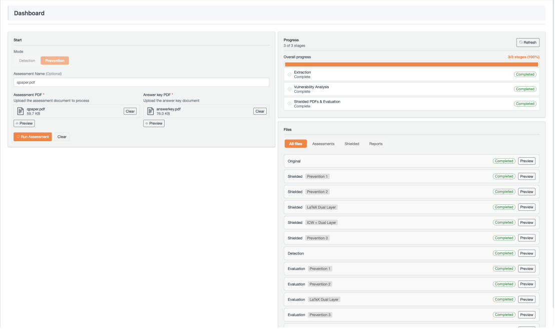
Figure 3: Stage 2: Watermark Embedding and AI Calibration. After planning, the system applies document-layer watermarks to the assessment PDF and evaluates original vs. watermarked versions against multiple MLLMs to generate prevention and detection reports.

We instantiate two watermark configurations, **🛡IS-v1** and **🛡IS-v2**, which differ in the density and combination of these mechanisms: **🛡IS-v1** uses a lighter mix of hidden-text and minimal glyph remapping, whereas **🛡IS-v2** employs stronger multi-layer perturbations for maximal robustness across parsing pipelines.

After rewriting, the system verifies that the rendered appearance of the PDF matches the original across common viewers (Adobe Reader, Chrome, macOS Preview). In the same stage, the authorship and calibration service evaluates both the original and watermarked versions with a panel of reference models in a simulated *"student uploads the exam"* setting, computing pre- versus post-watermark accuracy, the fraction of incorrect answers that land on intended distractors, and per-item watermark retrieval rates. An interactive report summarizes these statistics and assists instructors in selecting an appropriate watermark *"strength"* preset.

Stage 3: Authorship Analysis. After an assessment has been protected with our **🛡IS** watermarked PDFs, instructors can use it to analyze responses. The interface accepts either raw model outputs (for research) or anonymized student responses exported from a learning management system, depicted in Figure 4.

For each question, the authorship engine checks whether the response follows the stored watermark signature: for objective questions, this reduces to matching the target distractor (or a small tied set);



Dashboard

Start

Mode: Prevention

Assessment Name (Optional): qpaper.pdf

Assessment PDF *
Upload the assessment document to process
qpaper.pdf
59.7 KB

Answer key PDF *
Upload the answer key document
answerkey.pdf
76.0 KB

Progress
3 of 3 stages

Overall progress: 3/3 stages (100%)

- Extraction — Complete — Completed
- Vulnerability Analysis — Complete — Completed
- Shielded PDFs & Evaluation — Complete — Completed

Files

Original	Completed	Preview
Shielded Prevention 1	Completed	Preview
Shielded Prevention 2	Completed	Preview
Shielded LaTeX Dual Layer	Completed	Preview
Shielded ICW + Dual Layer	Completed	Preview
Shielded Prevention 3	Completed	Preview
Detection	Completed	Preview
Evaluation Prevention 1	Completed	Preview
Evaluation Prevention 2	Completed	Preview
Evaluation LaTeX Dual Layer	Completed	Preview
Evaluation Prevention 3	Completed	Preview

Figure 4: Stage 3: Authorship Analysis. The dashboard displays per-question watermark retrieval, exam-level authorship scores, and previewable shielded PDFs, enabling instructors to inspect AI-reliance signals.

for long-form questions, a judge LLM scores how closely the response tracks the watermark's key phrases or erroneous reasoning patterns. These per-item scores are aggregated into an exam-level authorship degree and displayed on a dashboard with cohort-level distributions and drill-down views for individual questions. The tool is explicitly positioned as an aid for triage rather than an automatic decision system: high authorship scores are intended to trigger follow-up actions such as brief oral checks or additional written assessments, keeping human judgment in the loop.

4 Experiments

Models and prompts. We evaluate IntegrityShield against a panel of four proprietary frontier MLLMs that support direct PDF ingestion: GPT-5, Claude Sonnet-4.5, Grok-4.1, and Gemini-2.5 Flash. All models are treated as black boxes and accessed via their official APIs, with temperature set to 0 and maximum output length sufficient to cover all questions in an exam. For each exam, we use a minimal, instruction-style prompt that (i) asks the model to answer all questions in order, (ii) returns a structured list of answers (e.g., *"Q1: A, Q2: C, . . . "* for MCQ and T/F; numbered paragraphs for Long-Form (LF)), and (iii) forbids external tools or web browsing. We use the same prompting templates for original and watermarked PDFs; full prompt text for MCQ, T/F, and LF questions appears in Appendix A as Prompt A, Prompt B, and Prompt C.

Baselines. We compare our two watermark configurations, **IS-v1** and **IS-v2**, against three document- or prompt-level baselines. **ICW** is an in-context watermarking method that attempts to steer model outputs using prompt-side patterns without modifying the PDF content using an invisible white color small sized font size (0.1-0.5) . (Liu et al., 2025). **code-glyph** is a document-layer baseline that manipulates bitcode to glyph mapping on question text to perturb parsing while keeping human readability intact (Xiong et al., 2025). **TrapDoc** adapts document-layer perturbations that introduce phantom tokens and layout tricks to cause models to produce plausible but incorrect answers without visible changes to the PDF (Jin et al., 2025). In contrast, **IS-v1** and **IS-v2** operate directly at the PDF substrate with schema-aware hidden text, glyph remapping, and overlays; **IS-v1** applies a lighter combination aimed at minimal perturbation, while **IS-v2** uses denser, multi-layer perturbations for maximal robustness.

Benchmark Dataset. To approximate real assessment settings, we compile a diverse benchmark of exam-style PDFs by web-scraping publicly available quizzes, homework sets, and midterm assessments from university course websites (e.g., Stanford and other institutions). The collected material spans mathematics, science, programming, humanities, and medical reasoning, and includes a mix of MCQ, T/F, and long-form questions. From this pool, we sample ≈10% of items to construct our benchmark, selecting documents that (i) contain at least ten questions, (ii) include at least three question formats, and (iii) render cleanly as PDFs. All items and answer keys are qualitatively reviewed by two authors and spot-validated quantitatively (e.g., via official solutions when available) to filter out ambiguous or mislabeled questions.

Evaluation metrics. We evaluate IntegrityShield along two complementary axes: *prevention*, which measures how strongly watermarking disrupts a model's ability to answer correctly, and *detection*, which captures how reliably watermark signatures can be recovered from model or student responses.

For **prevention**, we simply check whether watermarking causes the model to fail or refuse to answer the exam PDF. For exam d, $\mathrm{Prev}(f, d) = 1[y^{\mathrm{wm}}$ contains no usable answers$]$, and we report the percentage of PDFs where this occurs.

For **detection**, we measure the degree to which model outputs follow the embedded watermark signature. For each item, the authorship engine assigns a retrieval score $c_i \in [0, 1]$: for MCQ; T/F, $c_i = 1$ iff the model selects the target distractor;

for long-form, c_i is produced by a judge LLM evaluating alignment with watermark keyphrases. The exam-level detection score is

$$\mathrm{Det}(d', y) = \frac{1}{n_{d'}} \sum_{i=1}^{n_{d'}} c_i,$$

representing the proportion of responses that exhibit watermark-consistent behavior. We report detection scores per model and method, with breakdowns by question type.

4.1 Quantitative Analysis on Prevention and Detection

Table 1 summarizes prevention and detection performance for all baselines and our **I🛡S** variants across GPT, Claude, Grok, and Gemini.

Method	GPT	Claude	Grok	Gemini
		Prevention-ASR		
ICW	07.20	05.80	04.10	03.50
code-glyph	86.30	84.7	83.20	81.90
TrapDoc	88.70	86.40	85.10	40.50
🛡IS-v1	91.20	90.80	90.50	90.10
🛡IS-v2	**93.60**	**92.90**	**92.30**	**91.70**
		Detection		
ICW	06.80	05.30	04.60	03.20
code-glyph	85.90	84.20	82.70	81.40
TrapDoc	87.90	85.80	84.60	43.40
🛡IS-v1	90.70	90.30	89.90	89.50
🛡IS-v2	**92.80**	**92.10**	**91.60**	**91.00**

Table 1: Prevention and detection performance across models. Prevention-ASR is the percentage of exam PDFs on which watermarking causes the model to refuse or fail to produce usable answers. Detection is the percentage of questions whose responses follow the embedded watermark signature. For both metrics, higher is better. ICW: in-context watermarking; `code-glyph`: glyph perturbation; TrapDoc: phantom-token PDF attack; IS: 🛡IntegrityShield variants.

In the *Prevention-ASR* block, ICW almost never prevents models from answering full exams, with single-digit prevention rates across all models. This confirms that prompt-only steering is ineffective when students upload raw PDFs to black-box MLLMs. Document-layer baselines such as `code-glyph` and TrapDoc are substantially stronger on GPT, Claude, and Grok (around 83 89% prevention), but TrapDoc degrades sharply on Gemini (40.5%), suggesting that its perturbations do not transfer reliably across parsing and model stacks. By contrast, **🛡IS-v1** and **🛡IS-v2** achieve consistently high prevention on *all* models (90 94%), indicating that schema-aware, multi-layer PDF watermarking can robustly block end-to-end exam solving for contemporary MLLMs.

The *Detection* block shows a similar pattern. ICW again yields negligible detection rates, while `code-glyph` and TrapDoc achieve strong detection on GPT, Claude, and Grok (mid–80s); however, TrapDoc drops to 43.4% on Gemini. In contrast, **🛡IS-v1** and especially **🛡IS-v2** maintain high detection performance across all four models (around 89–93%), meaning that whenever models do attempt to answer on watermarked exams, their outputs follow the embedded watermark signatures in a highly consistent way, enabling reliable authorship attribution.

4.2 🛡IS Performance for Question-Category

Table 2 breaks down the impact of **IS** on answer accuracy by question type (MCQ, T/F, LF) and model, comparing performance on original (w/o) and watermarked ($w/$) exams. Without watermarking, all four MLLMs attain very high accuracy across categories (typically 94-97%), reflecting their strong baseline performance on our exam-style benchmark.

Type	**GPT**		**Claude**		**Grok**		**Gemini**	
	w/o	$w/$	w/o	$w/$	w/o	$w/$	w/o	$w/$
MCQ	96.2	**7.8**	95.8	**6.9**	94.9	**5.7**	94.1	**4.3**
T/F	95.7	**6.5**	95.3	**5.8**	94.6	**4.9**	93.8	**3.6**
LF	96.8	**5.2**	96.4	**4.6**	95.3	**3.8**	94.7	**3.1**

Table 2: Per-question-type answer accuracy without (w/o) and with ($w/$) our 🛡IntegrityShield watermarks. Values show the residual accuracy of each model on shielded exams; lower $w/$ accuracy indicates stronger prevention for that question type.

With 🛡IntegrityShield enabled, residual accuracy collapses into the low single digits for every model and question type (3-8%), corresponding to an 85-90 point drop. Long-form questions show the largest reductions for GPT and Claude, while MCQ and T/F items are also heavily disrupted across all models. These results indicate that our document-layer watermarks are effective not only at the exam level, but also uniformly across different assessment formats.

4.3 Utility of 🛡IntegrityShield

Beyond aggregate metrics, 🛡IntegrityShield provides instructors with actionable, item-level evidence of AI reliance. Table 3 illustrates this with a qualitative example: on an OSI-model question, all baseline attacks (`ICW`, `code-glyph`, TrapDoc)

Attack Method	GPT	Claude	Grok	Gemini
ICW	A	A	A	A
`code-glyph`	A	A	A	A
TrapDoc	A	A	A	A
🛡IS-v1	B	B	B	B
🛡IS-v2	C	C	C	C

Table 3: Model predictions across attack methods for the OSI model question. ***Q:*** *Which layer of the OSI model is responsible for routing packets between networks?* ***Gold Answer: A***

collapse to the same incorrect prediction across models, offering no consistent signal for attribution. In contrast, our schema-aware variants (**🛡IS-v1**, **🛡IS-v2**) drive models toward distinct, watermark-aligned distractors (B and C, respectively), enabling clear and separable authorship signatures.

In a *prevention-focused* deployment, the system summarizes where watermarking fully blocks a model from answering an exam, providing a document-level view of which assessments are resilient to AI-based shortcuts. In a *detection-focused* deployment, the system aggregates authorship evidence across questions, showing, for example, that *"Q3 follows the* 🛡IS-v2 *signature across multiple models"*.

These reports are intended as triage tools: instructors can identify items likely influenced by AI, perform brief oral checks or follow-up tasks, and intervene proportionally. By surfacing interpretable authorship signals rather than relying on opaque classifiers or intrusive proctoring, 🛡IntegrityShield enables ethical, transparent, and governance-aligned AI use in educational assessments.

5 Conclusion

🛡IntegrityShield A System for Ethical AI Use & Authorship Transparency in Assessments, demonstrates that assessment integrity can be strengthened without invasive monitoring by instrumenting the exam document itself. By operating directly at the PDF substrate, our system embeds schema-aware watermarks that both (i) prevent modern MLLMs from answering shielded exams (91-94% exam-level blocking) and (ii) yield stable, recoverable authorship signatures (89-93% retrieval) when AI is used. These effects hold consistently across question types and four commercial MLLMs, highlighting the robustness of document-layer watermarking as a practical defense.

The demo showcases a complete workflow for real instructional use: uploading an exam, previewing watermark strategies, generating shielded variants, running automated AI calibration, and inspecting item-level authorship evidence. This combination of prevention and attribution provides instructors and institutions with actionable, interpretable signals supporting fair assessment practices, targeted follow-up, and transparent communication with students.

We hope 🛡IntegrityShield serves as a step toward ethically grounded AI governance in education, enabling institutions to observe AI reliance without resorting to surveillance or restricting access to assistive technologies.

Limitations

Our evaluation is limited to a thirty-exam benchmark, a fixed set of frontier MLLMs, and simulated usage in which models directly consume instructor PDFs. Real-world deployments may involve broader variation in domains, languages, accessibility workflows (e.g., screen readers), and institution-specific formats. As MLLMs and their PDF-parsing pipelines evolve, watermark robustness may drift, necessitating periodic recalibration.

🛡IntegrityShield is not a definitive detector of misconduct. Authorship scores indicate alignment with watermark signatures not whether a student violated policy and should be used as a triage signal for human follow-up (e.g., brief oral checks), not as automatic evidence for sanctions. We acknowledge that we did not evaluate robustness against rasterization, ghostscript re-encoding, Word export/import, or PDF sanitization tools. A technically sophisticated adversary who transforms the document first falls outside our current scope. Students who instead choose to retype questions, take screenshots, or manually transcribe content introduce meaningful friction that itself serves as a partial deterrent. More importantly, such workarounds would be inconsistent with the typical behavior of students seeking a quick, low-effort solution, which is the primary risk profile the system addresses.

The system is designed for deployability under current parsing pipelines, with the expectation that watermark strategies will need to be updated over time; analogous to how anti-virus signatures or plagiarism detection tools require periodic updates.

Finally, our approach assumes institutional control over assessment PDFs. Similar watermarking techniques could be misapplied to non-assessment documents, so we explicitly restrict the intended use to formal educational settings with clear governance, transparency, and AI-use policies. PDF is the dominant format for distributed assessments in higher education, which motivated our focus. Extension to other formats (e.g., Images, HTML-based assessments) is an important direction for future work.

Ethics Statement

This work aims to support ethical and transparent AI use in educational assessment settings. 🛡IntegrityShield operates exclusively on instructor-provided documents and does not monitor students, avoiding surveillance-heavy practices such as keystroke logging, webcam tracking, or device control. The system is designed to keep all responses and analyzes within institutional infrastructure, respecting student privacy and data-governance requirements.

Authorship scores produced by our watermarking framework indicate alignment with embedded watermark signatures; they do *not* constitute evidence of misconduct. We recommend that institutions (i) clearly communicate AI-use policies and the presence of watermarking to students, (ii) treat high authorship scores only as signals for human review (e.g., follow-up questions or oral checks), and (iii) ensure that any use of these signals aligns with local policies, academic integrity guidelines, and privacy regulations. The system is designed as a practical deterrent and authorship signal for institutional triage, not as a cryptographically secure system. We also note that IS-v2's multi-layer approach (CMap remapping + overlays) is harder to isolate than pure hidden-text injection, though we do not claim this is detection-proof.

All experiments were conducted with fixed model parameters (e.g., temperature, top_p, top_k) to mitigate stochastic variability in black-box LLMs. Models used in this work (e.g., GPT-5, Gemini-2.5 Flash, Grok-4.1, Claude Sonnet-4.5) were accessed in accordance with their respective usage policies. Data labeling and verification were performed by author-annotators, and AI-based tools (e.g., Grammarly, ChatGPT) were used strictly for language refinement. To the best of our knowledge, this study introduces no additional ethical risks beyond those common to LLM evaluation in controlled educational settings.

Acknowledgements

We thank the Complex Data Analysis and Reasoning Lab at Arizona State University for computational support, and Sandipan De and Eun Woo Im for their helpful reviews of earlier drafts of this work. We are also grateful to Ben Zhou, Pavan Turaga, and Janice Mak for their valuable feedback and suggestions, and to the Arizona State University academic integrity office for their support and feedback.

References

Yousef Atoum, Liping Chen, Alex X. Liu, Stephen D. H. Hsu, and Xiaoming Liu. 2017. Automated Online Exam Proctoring. *IEEE Transactions on Multimedia*, 19(7):1609–1624.

Bradley Emi and Max Spero. 2024. Technical Report on the Pangram AI-Generated Text Classifier. Technical report, Pangram Labs.

Hyundong Jin, Sicheol Sung, Shinwoo Park, SeungYeop Baik, and Yo-Sub Han. 2025. TRAPDOC: Deceiving LLM Users by Injecting Imperceptible Phantom Tokens into Documents. *EMNLP Findings*. ArXiv:2506.00089.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, and 1 others. 2023. A Watermark for Large Language Models. *arXiv preprint arXiv:2301.10226*.

Debnath Kundu, Atharva Mehta, Rajesh Kumar, Naman Lal, Avinash Anand, Apoorv Singh, and Rajiv Ratn Shah. 2024. Keystroke Dynamics Against Academic Dishonesty in the Age of LLMs. In *Proceedings of the IEEE International Joint Conference on Biometrics (IJCB)*.

Yepeng Liu and Yuheng Bu. 2024. Adaptive Text Watermark for Large Language Models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 30718–30737. PMLR.

Yepeng Liu, Xuandong Zhao, Christopher Kruegel, Dawn Song, and Yuheng Bu. 2025. In-Context Watermarks for Large Language Models. *arXiv preprint arXiv:2505.16934*.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. DetectGPT: Zero-shot Machine-Generated Text Detection Using Probability Curvature. *Proceedings of the 40th International Conference on Machine Learning*.

Chenhao Niu, Kevin P. Yancey, Ruidong Liu, Mirza Basim Baig, André Kenji Horie, and James Sharpnack. 2024. Detecting LLM-Assisted Cheating on Open-Ended Writing Tasks on Language Proficiency Tests. *EMNLP Industry Track*.

OpenAI. 2023. ChatGPT. OpenAI blog.

Teo Susnjak. 2022. ChatGPT: The End of Online Exam Integrity? *Preprint*, arXiv:2212.09292.

Gemini Team. 2024. Gemini 1.5: Unlocking Multimodal Understanding Across Millions of Tokens of Context. *arXiv preprint arXiv:2403.05530*.

Katherine Thai, Bradley Emi, Elyas Masrour, and Mohit Iyyer. 2025. EditLens: Quantifying the Extent of AI Editing in Text. *Preprint*, arXiv:2510.03154.

Junjie Xiong, Changjia Zhu, Shuhang Lin, Chong Zhang, Yongfeng Zhang, Yao Liu, and Lingyao Li. 2025. Invisible Prompts, Visible Threats: Malicious Font Injection in External Resources for Large Language Models. *Preprint*, arXiv:2505.16957.

A Prompts Details

Prompt A: MCQ Perturbation

```
You are an expert at generating text substitutions for academic multiple-choice questions.

Given:
- LaTeX code for the question stem: {latex_stem_text}
- Gold answer: {gold_answer}
- Question type: {question_type}
- Options: {options}
- Strategy: replacement
- Reasoning steps:
{reasoning_steps}
- Copyable text (use this exact text when selecting substrings):
<<<COPY
{copyable_text}
>>>
{prefix_note}{answer_guidance}{retry_instructions}

Your task:
Generate {k} valid mappings that satisfy the replacement strategy. Each mapping should:
1. Replace one contiguous substring of the question stem
2. Change the answer from the gold option ({gold_answer}) to a different incorrect option
3. Ensure the replacement is semantically meaningful and natural
4. Cause a verifiable deviation in the answer

For each mapping, provide:
1. question_index: The question number ({question_index})
2. latex_stem_text: Exact LaTeX text of the question stem (must match the input exactly)
3. original_substring: The substring to replace (must be a contiguous substring of latex_stem_text)
4. replacement_substring: The replacement text
5. start_pos: Start position of original_substring relative to latex_stem_text (0-based index)
6. end_pos: End position of original_substring relative to latex_stem_text (exclusive, 0-based index)
7. target_wrong_answer: The target incorrect option label (e.g., "B", "C", "D")
8. reasoning: Brief explanation of why this mapping satisfies the strategy

IMPORTANT:
- The original_substring MUST be an exact substring of latex_stem_text
- The start_pos and end_pos MUST be accurate (start_pos + len(original_substring) = end_pos)
- The target_wrong_answer MUST be different from the gold answer
- CRITICAL: The replacement_substring MUST be DIFFERENT from the original_substring. Do NOT generate mappings where original_substring == replacement_substring (e.g., "power" --> "power" is INVALID). The replacement MUST change the text to create actual manipulation.
- CRITICAL: Neither original_substring nor replacement_substring can be empty strings. Both must contain actual text.
- LENGTH CONSTRAINT: The replacement_substring MUST be smaller or equal in length to the original_substring (len(replacement_substring) <= len(original_substring)). This is critical for maintaining document layout and preventing text overflow.
- latex_stem_text is provided exactly as it appears in the LaTeX source. Do NOT trim, normalise, or reformat it when determining positions.
- The latex_stem_text may include \item tokens from enumerate environments. Keep the \item token intact and operate on the descriptive text that follows it whenever possible.
- The replacement should be natural and semantically meaningful

Return as JSON array:
[
  {{
    "question_index": {question_index},
    "latex_stem_text": "...",
    "original_substring": "...",
    "replacement_substring": "...",
    "start_pos": 0,
    "end_pos": 5,
    "target_wrong_answer": "B",
    "reasoning": "..."
  }},
  ...
]

Return ONLY valid JSON, no markdown or additional text.
```

Prompt B: True False Perturbation

```
You are an expert at generating text substitutions for True/False questions.

Given:
- LaTeX code for the question stem: {latex_stem_text}
- Gold answer: {gold_answer}
- Question type: {question_type}
- Strategy: replacement
- Reasoning steps:
{reasoning_steps}
- Copyable text (use this exact text when selecting substrings):
<<<COPY
{copyable_text}
>>>
{prefix_note}{answer_guidance}{retry_instructions}

Your task:
Generate {k} valid mappings that satisfy the replacement strategy. Each mapping should:
1. Replace one contiguous substring of the question stem
2. Flip the answer from {gold_answer} to the opposite answer
3. Ensure the replacement is semantically meaningful and natural
4. Cause a verifiable deviation in the answer

For each mapping, provide:
1. question_index: The question number ({question_index})
2. latex_stem_text: Exact LaTeX text of the question stem (must match the input exactly)
3. original_substring: The substring to replace (must be a contiguous substring of latex_stem_text)
4. replacement_substring: The replacement text
5. start_pos: Start position of original_substring relative to latex_stem_text (0-based index)
6. end_pos: End position of original_substring relative to latex_stem_text (exclusive, 0-based index)
7. target_wrong_answer: The opposite answer (e.g., "False" if gold is "True", or "True" if gold is "False")
8. reasoning: Brief explanation of why this mapping satisfies the strategy

IMPORTANT:
- The original_substring MUST be an exact substring of latex_stem_text
- The start_pos and end_pos MUST be accurate (start_pos + len(original_substring) = end_pos)
- The target_wrong_answer MUST be the opposite of the gold answer
- CRITICAL: The replacement_substring MUST be DIFFERENT from the original_substring. Do NOT generate mappings where
      ↪ original_substring == replacement_substring (e.g., "force" --> "force" is INVALID). The replacement MUST change the
      ↪ text to create actual manipulation.
- CRITICAL: Neither original_substring nor replacement_substring can be empty strings. Both must contain actual text.
- LENGTH CONSTRAINT: The replacement_substring MUST be smaller or equal in length to the original_substring (len(
      ↪ replacement_substring) <= len(original_substring)). This is critical for maintaining document layout and preventing
      ↪ text overflow.
- latex_stem_text is provided exactly as it appears in the LaTeX source. Do NOT trim, normalise, or reformat it when
      ↪ determining positions.
- The latex_stem_text may include \item tokens from enumerate environments. Keep the \item token intact and operate on the
      ↪ descriptive text that follows it whenever possible.
- The replacement should be natural and semantically meaningful

Return as JSON array:
[
  {{
    "question_index": {question_index},
    "latex_stem_text": "...",
    "original_substring": "...",
    "replacement_substring": "...",
    "start_pos": 0,
    "end_pos": 5,
    "target_wrong_answer": "False",
    "reasoning": "..."
  }},
  ...
]

Return ONLY valid JSON, no markdown or additional text.
```

Prompt C: LongForm Perturbation

```
You are an expert at generating text substitutions for long-form questions (essay, short answer, etc.).

Given:
- LaTeX code for the question stem: {latex_stem_text}
- Gold answer: {gold_answer}
- Question type: {question_type}
- Strategy: replacement
- Reasoning steps:
{reasoning_steps}
- Copyable text (use this exact text when selecting substrings):
<<<COPY
{copyable_text}
>>>
{prefix_note}{answer_guidance}{retry_instructions}

Your task:
Generate {k} valid mappings that satisfy the replacement strategy. Each mapping should:
1. Replace one contiguous substring of the question stem
2. Cause a verifiable and detectable deviation from the gold answer
3. Ensure the replacement is semantically meaningful and natural
4. Change the question focus in a way that affects the expected answer

For each mapping, provide:
1. question_index: The question number ({question_index})
2. latex_stem_text: Exact LaTeX text of the question stem (must match the input exactly)
3. original_substring: The substring to replace (must be a contiguous substring of latex_stem_text)
4. replacement_substring: The replacement text
5. start_pos: Start position of original_substring relative to latex_stem_text (0-based index)
6. end_pos: End position of original_substring relative to latex_stem_text (exclusive, 0-based index)
7. target_wrong_answer: Description of how the answer should deviate (e.g., "focuses on different aspect", "changes key concept")
8. reasoning: Brief explanation of why this mapping satisfies the strategy and how it causes deviation

IMPORTANT:
- The original_substring MUST be an exact substring of latex_stem_text
- The start_pos and end_pos MUST be accurate (start_pos + len(original_substring) = end_pos)
- The replacement should cause a verifiable deviation in the answer
- CRITICAL: The replacement_substring MUST be DIFFERENT from the original_substring. Do NOT generate mappings where original_substring == replacement_substring. The replacement MUST change the text to create actual manipulation.
- CRITICAL: Neither original_substring nor replacement_substring can be empty strings. Both must contain actual text.
- LENGTH CONSTRAINT: The replacement_substring MUST be smaller or equal in length to the original_substring (len(replacement_substring) <= len(original_substring)). This is critical for maintaining document layout and preventing text overflow.
- latex_stem_text is provided exactly as it appears in the LaTeX source. Do NOT trim, normalise, or reformat it when determining positions.
- The latex_stem_text may include \item tokens from enumerate environments. Keep the \item token intact and operate on the descriptive text that follows it whenever possible.
- The replacement should be natural and semantically meaningful

Return as JSON array:
[
  {{
    "question_index": {question_index},
    "latex_stem_text": "...",
    "original_substring": "...",
    "replacement_substring": "...",
    "start_pos": 0,
    "end_pos": 5,
    "target_wrong_answer": "focuses on different aspect",
    "reasoning": "..."
  }},
  ...
]

Return ONLY valid JSON, no markdown or additional text.
```