FRAGILE WATERMARK IN MASKED DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in diffusion language models have demonstrated competitive performance across diverse language tasks, highlighting the potential of parallel text generation. However, ensuring the integrity of their outputs remains a challenge, as generated content may be modified during editing and redistribution. Existing fragile watermarking methods rely on the sequential dependencies of autoregressive generation and therefore cannot be directly applied to the iterative parallel denoising process of masked diffusion language models (MDLMs). To address this gap, we introduce **Dandelion**, to our knowledge the first fragile watermarking framework specifically designed for MDLMs. Dandelion constructs a block-level Markov chain that links content-dependent watermark tags through successive block states, enabling integrity verification while preserving parallel generation within each block. It embeds watermark bits through constrained maximum-probability token selection to preserve generation quality, and verifies received text without access to the original output or the generation model. Experiments on six representative text-only and multimodal MDLMs show that Dandelion achieves a 100% true-positive rate on intact watermarked outputs and an average true-negative rate exceeding 99% across 13 perturbation types. Further evaluations demonstrate high semantic similarity and preserved task performance, with an average runtime overhead of 5.42%. Our code is available at https://anonymous.4open.science/r/Dandelion-DLLM_Fragile_Watermark-32EF.

1 INTRODUCTION

With the advancement of generative artificial intelligence, **D**iffusion **L**anguage **M**odels (DLMs) have emerged as one of the most promising classes of language models because of their ability to generate tokens in parallel (Li et al., 2022; Sahoo et al., 2024). Among these models, **M**asked **D**iffusion **L**anguage **M**odels (MDLMs) are one of the most extensively studied and widely adopted paradigms (Zhu et al., 2025; 2026). They have demonstrated performance comparable to that of autoregressive language models of similar size across a range of tasks (Zhao et al., 2025b; Gong et al., 2023). In these tasks, generated content is often copied, edited, and redistributed. Ensuring its detectability and integrity is therefore essential. Watermarking (Kirchenbauer et al., 2023; Zhao et al., 2024a) is widely regarded as one of the most effective strategies: statistical watermarking (Wen et al., 2023; Dathathri et al., 2024) addresses this need by introducing a bias during generation that encourages the model to select tokens from a designated green list. The detector can then identify the watermark from statistical patterns without access to the model's internal parameters. However, in sensitive applications, the robustness of statistical watermarks to local edits and content tampering may create a risk that false or harmful content will be authenticated. Because the detector relies solely on statistical features, it cannot determine whether the content has been modified by a third party. Fragile watermarking (Zhang et al., 2024; Sahu et al., 2022) is designed to address this limitation. When the content remains intact, the detector can use a secret key to verify its provenance. If any portion of the content is modified, however, the fragile watermark is invalidated, and the content is classified as untrusted.

Existing fragile watermarking methods are primarily designed for autoregressive language models, in which each token is generated conditionally on the preceding sequence. This causal generation process enables watermark signals to be embedded and verified on a token-by-token basis. However, MDLMs generate text through iterative denoising and may update multiple token positions in

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.

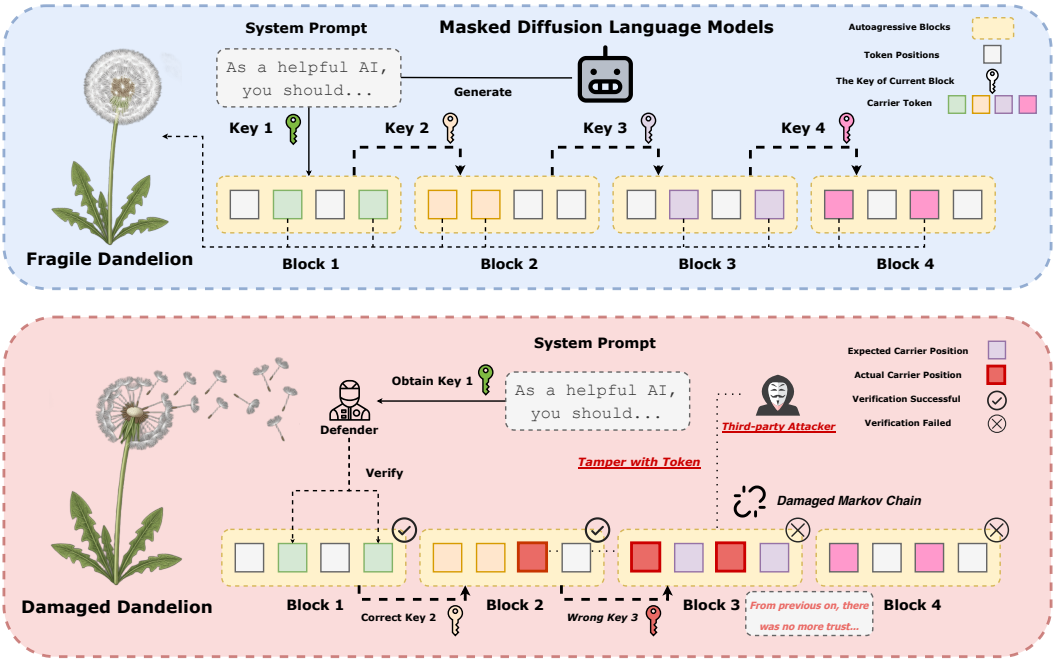

Figure 1: **Dandelion's pipeline:** Dandelion forms a block-level Markov chain through autoregressive dependencies. Each block key is derived from a server-held master key and the system prompt; its tag binds the payload to the preceding state and commits the realized carrier tokens. A modification that changes the authenticated token sequence disrupts this dependency and almost certainly causes verification to fail, subject to the boundary cases discussed in the appendix.

parallel Ben-Hamu et al. (2025). Consequently, their generation process does not follow the left-to-right Markov structure assumed by autoregressive watermarking methods. Directly applying these methods to masked diffusion models may therefore result in unreliable watermark embedding and verification. Despite the increasing adoption of masked diffusion models, fragile watermarking for this generative paradigm remains largely unexplored.

To address this issue, we propose **Dandelion**: a block-based fragile watermarking framework specifically designed for masked diffusion language models. Dandelion constructs the dependencies between token blocks and uses block-level Markov states to verify sequence integrity. This design is compatible with the parallel generation mechanism of masked diffusion models, and an insightful observation allows us to select high-entropy tokens as watermark carriers with minimal impact on the model's generation quality. In our experiments, Dandelion exhibited consistent performance across several mainstream models, including LLaDA-8B-Instruct (trained from scratch), LLaDA 2.0 (incorporating autoregressive priors), and LaViDA (a multimodal vision–language model). Our contributions are as follows:

- **We propose Dandelion, to our knowledge the first fragile watermarking framework for masked diffusion language models.** It embeds watermarks through autoregressive dependencies across token blocks and key-dependent carriers, while preserving generation quality with minimal degradation.
- **We analyze the incompatibility between autoregressive fragile watermarking and masked diffusion language models.** Dandelion exploits parallel generation to avoid interference between watermark carriers and generation-critical tokens.
- **We evaluate Dandelion on six representative MDLMs.** It achieves a 100% true-positive rate and an average true-negative rate above 99% under 13 perturbation types. Further experiments confirm its robustness across generation settings and reveal the trade-off between watermark capacity and final-block detection.

2 Related Work

2.1 Masked Diffusion Language Models (MDLMs)

Masked Diffusion Language Models (MDLMs) generate discrete token sequences by reversing an absorbing-mask corruption process, rather than decoding strictly from left to right (Sahoo et al., 2024; He et al., 2023).

Let $\mathbf{x}_0 = (x_0^1, \ldots, x_0^n) \in \mathcal{V}^n$ denote a clean token sequence, where $\mathcal{V}$ is the vocabulary and $m \notin \mathcal{V}$ is the mask token. For $t \in \{0, \ldots, T\}$, let α_t be a decreasing retention schedule with $\alpha_0 = 1$ and $\alpha_T \approx 0$. The forward process independently retains or masks each position:

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \prod_{i=1}^{n} \left[\alpha_t \, \delta_{x_0^i}(x_t^i) + (1 - \alpha_t) \, \delta_m(x_t^i) \right], \tag{1}$$

where $\delta_a(b)$ is the point mass at a. Equivalently, with $z_t^i \sim \mathrm{Bernoulli}(\alpha_t)$:

$$x_t^i = \begin{cases} x_0^i, & z_t^i = 1, \\ m, & z_t^i = 0. \end{cases} \tag{2}$$

For a stepwise chain, let β_t be the masking probability at step t. The transition is defined as follows:

$$q(x_t^i \mid x_{t-1}^i) = \begin{cases} (1 - \beta_t)\delta_{x_{t-1}^i}(x_t^i) + \beta_t \delta_m(x_t^i), & x_{t-1}^i \neq m, \\ \delta_m(x_t^i), & x_{t-1}^i = m, \end{cases} \tag{3}$$

Here, $\alpha_t = \prod_{\tau=1}^{t}(1 - \beta_\tau)$, and the second case makes m absorbing.

Reverse Sampling. Generation starts from $\mathbf{x}_T = (m, \ldots, m)$. Given $\mathbf{x}_t$, the denoiser predicts $p_\theta(x_0^i \mid \mathbf{x}_t)$ (Arriola et al., 2025). For $0 \leq s < t \leq T$, the exact posterior at a masked position is:

$$q(x_s^i \mid x_t^i = m, x_0^i) = \frac{\alpha_s - \alpha_t}{1 - \alpha_t} \, \delta_{x_0^i}(x_s^i) + \frac{1 - \alpha_s}{1 - \alpha_t} \, \delta_m(x_s^i), \qquad s < t. \tag{4}$$

Replacing the unknown x_0^i with the model prediction gives the probability of revealing $u \in \mathcal{V}$:

$$p_\theta(x_s^i = u \mid x_t^i = m, \mathbf{x}_t) = \frac{\alpha_s - \alpha_t}{1 - \alpha_t} \, p_\theta(x_0^i = u \mid \mathbf{x}_t), \qquad u \in \mathcal{V}, \tag{5}$$

and

$$p_\theta(x_s^i = m \mid x_t^i = m, \mathbf{x}_t) = \frac{1 - \alpha_s}{1 - \alpha_t}. \tag{6}$$

In practice, the model predicts a candidate $\hat{x}_{0,t}^i$ and confidence c_i at every masked position. Let $\mathcal{S}_t = \mathrm{TopK}_{i:\, x_t^i = m}(c_i)$ contain the K_t highest-confidence masked positions. They are revealed in parallel:

$$x_{t-1}^i = \begin{cases} x_t^i, & x_t^i \neq m, \\ \hat{x}_{0,t}^i, & i \in \mathcal{S}_t, \\ m, & \text{otherwise.} \end{cases} \tag{7}$$

Under a linear masking schedule, the transfer count is commonly chosen as:

$$|\mathcal{S}_t| \approx \min\left\{ \left| \{i : x_t^i = m\} \right|, n \, (\alpha_{t-1} - \alpha_t) \right\}. \tag{8}$$

Thus, MDLMs update multiple positions without a fixed left-to-right order.

Block Sampling. Blockwise variants restrict parallel denoising to contiguous masked positions, improving sampling structure while retaining within-block parallelism (Arriola et al., 2025). Given $p_\theta(\mathbf{x}_0 \mid \mathbf{x}_t)$, let $\mathcal{B}_t \subseteq \{i : x_t^i = m\}$ be the selected block. Its update is:

$$x_{t-1}^i = \begin{cases} \hat{x}_{0,t}^i, & i \in \mathcal{B}_t, \\ x_t^i, & \text{otherwise.} \end{cases} \tag{9}$$

This block-level trajectory supplies the sequential inter-block dependency used by Dandelion, while tokens within a block remain non-autoregressive.

2.2 Watermarking in Language Models

Static Watermark. Existing watermarking methods for language models mainly focus on static watermarks (Zhao et al., 2024b; Un & Choi, 2025), which embed persistent signals into the generated text and enable ownership verification through statistical or semantic patterns. Representative approaches modify the token generation distribution, such as biasing selected token sets, introducing predefined patterns, or embedding signals into model outputs (Peng et al., 2025; Hao et al., 2020). These methods have demonstrated effectiveness in detecting machine-generated text under benign conditions. However, static watermarks are increasingly challenged by watermark removal attacks. Recent studies show that attackers can erase or weaken watermark signals through paraphrasing, rewriting, translation, or regeneration with another language model while preserving the original meaning (Wu & Chandrasekaran, 2024; Zhao et al., 2025a; Jovanović et al., 2024). Since static watermarks rely on fixed and observable generation patterns, they are vulnerable once the attacker identifies or disrupts these patterns. This motivates the exploration of integrity-oriented watermarking schemes that can detect unauthorized modifications rather than relying only on persistent output-level signals (Barni et al., 1998; Wan et al., 2022).

Fragile Watermark. Fragile watermarking provides a different paradigm from static watermarking by shifting the goal from ownership identification to content integrity verification (Lin et al., 2000; Ali et al., 2022). Instead of embedding a persistent signal that remains detectable after various transformations, fragile watermarks are designed to be sensitive to unauthorized modifications. A valid generation preserves the embedded watermark, while operations such as rewriting, editing, or tampering will disrupt the watermark signal and trigger detection. This property makes fragile watermarking particularly suitable for applications requiring authenticity verification, where identifying whether a generated content has been altered is more important than only determining its origin (Botta et al., 2021; Yuan et al., 2025).

Recent studies have explored fragile watermarking for autoregressive language models by exploiting the sequential dependency of token generation (Gao et al., 2026; Wu et al., 2026). Since autoregressive models generate tokens following a fixed left-to-right order, watermark signals can be coupled with generation trajectories or token dependencies, enabling the detection of modifications that violate the original generation process. However, these autoregressive fragile watermarking methods cannot be directly transferred to masked diffusion language models (Tsur et al., 2025b;a). Unlike autoregressive models, MDLMs generate text through iterative parallel denoising, where multiple tokens are recovered simultaneously without a predefined generation order. Consequently, there is no unique autoregressive trajectory or token dependency chain to associate with a fragile watermark. Developing fragile watermarking methods that exploit the characteristics of parallel denoising remains an open challenge for MDLMs.

3 Method

3.1 Watermark Embedding

Dandelion embeds a content-dependent fragile watermark during the blockwise generation of an MDLM. Let $\mathbf{x}_0 = (x_0^1, \ldots, x_0^n) \in \mathcal{V}^n$ denote the generated sequence and let $\widetilde{\mathbf{x}}_0$ denote its watermarked counterpart. We divide the generated suffix into contiguous blocks of length B. The position set of block j is defined as follows:

$$\mathcal{I}_j = \{jB + 1, \ldots, \min((j+1)B, n)\}. \tag{10}$$

After block j has been fully denoised, Dandelion identifies a deterministic set of carrier positions. A position is lexically eligible when its token is not a special token and its decoded piece begins with whitespace followed by an alphabetic character. Number-initial pieces are excluded because years, constants, and parameter values often have exact factual or executable meaning. This restriction gives the carrier a stable word boundary and prevents punctuation or control symbols from being selected. Lexical eligibility alone is insufficient for exact-content regions: changing an identifier in executable code or a symbol in a formula can invalidate the answer even when the replacement is locally probable. We therefore construct $\mathcal{R}(\mathbf{x}_0)$, a deterministic set of token positions that overlap fenced or inline code, HTML code elements, indented code, dollar- or bracket-delimited mathematics, or LaTeX mathematical environments. The scan is performed over serialized text and uses no generation metadata. Let $\mathcal{E}_j(\mathbf{x}_0)$ contain the lexically eligible positions in block j that are not in $\mathcal{R}(\mathbf{x}_0)$. The public carrier rule is written as follows:

$$\mathcal{C}_j(\mathbf{x}_0) = \mathrm{Select}(\mathcal{E}_j(\mathbf{x}_0)) \subseteq \mathcal{I}_j, \tag{11}$$

where $\mathrm{Select}(\cdot)$ returns a predetermined ordered subset of eligible positions. The same rule is used during detection, so carrier locations and exact-region exclusions do not need to be stored as side information. A strict-heavy block can carry fewer bits than the configured maximum; Dandelion never substitutes a strict-region token merely to reach the target capacity. If any nonempty block contains no safe carrier, embedding fails rather than returning an incompletely protected output. The ordered tokens in $\mathcal{I}_j \setminus \mathcal{C}_j$ form the payload $\mathcal{P}_j(\mathbf{x}_0)$. Separating the carriers from the payload is important: carrier values can be changed to encode the watermark without changing the content from which the watermark is computed.

After completing a block, an MDLM also permits a leave-one-out scan in which each observed token is masked and rescored. We record the normalized score $s_i = \log p_\theta(x_0^i \mid \mathbf{x}_0^{\setminus i}, \mathbf{p})$ as a carrier-risk diagnostic. In our pilot observations, prepositions and conjunctions tend to receive lower scores than content-bearing words. We do not directly use this score in the public carrier rule: changing a carrier changes its observed-token score, and a detector that does not run the MDLM could then reconstruct a different carrier set. Instead, we evaluate low-score selection as a structure-aware ablation while retaining the deterministic exact-region exclusion in the deployable method. We compare normalized log probabilities rather than raw logits, whose additive scale is not calibrated across masked positions. A low observed-token probability is not itself evidence that arbitrary substitution is semantically harmless, so the ablation must be judged by downstream quality and execution tests.

Dandelion maintains an independent high-entropy master key K_{master} on the server. Let $\mathbf{p}_{\mathrm{sys}}$ be the server-controlled system prompt and $\mathrm{Enc}(\cdot)$ its canonical UTF-8 encoding. The watermark key is derived using a key-derivation function:

$$K = \mathrm{KDF}(K_{\mathrm{master}}, \mathrm{Enc}(\mathbf{p}_{\mathrm{sys}}))\,. \tag{12}$$

Thus, the system prompt provides domain-specific context but is not itself the secret. Write the ordered carrier set as $\mathcal{C}_j = (i_{j,1}, \ldots, i_{j,R_j})$, where $R_j = |\mathcal{C}_j|$. Before embedding block j, Dandelion computes a block tag t_j from its payload and the preceding chain state, and extracts the target carrier bits:

$$t_j = F_K(n, j, \mathcal{P}_j(\mathbf{x}_0), h_{j-1})\,, \qquad b_{j,r} = \mathrm{bit}_r(t_j), \quad r \in \{1, \ldots, R_j\}. \tag{13}$$

Here, F_K is a keyed pseudorandom function, h_{j-1} is the preceding chain state, and $b_{j,r}$ is the bit assigned to the r-th carrier in block j. For the first block, $h_{-1} = \epsilon$ is a fixed empty state.

To embed $b_{j,r}$, Dandelion temporarily replaces carrier position $i_{j,r} \in \mathcal{C}_j$ with the mask token m and queries the MDLM again. Let $\overline{\mathbf{x}}_{j,r}$ denote this masked sequence, and let $\mathcal{A}_{j,r}$ be the high-probability candidates that preserve the original carrier's whitespace and lexical signature. Candidates whose token IDs change after decoding and re-tokenization are removed from $\mathcal{A}_{j,r}$, ensuring that generation and text-only detection observe the same token sequence. A domain-separated keyed function

maps every candidate token v to a binary carrier value $g_K(n, j, r, v)$. Dandelion selects the most probable candidate carrying the required bit as follows:

$$\widetilde{x}_0^{i_{j,r}} = \underset{\substack{v \in \mathcal{A}_{j,r}: \\ g_K(n,j,r,v)=b_{j,r}}}{\arg\max} \; p_\theta\Big(x_0^{i_{j,r}} = v \mid \overline{\mathbf{x}}_{j,r}, \mathbf{p}\Big), \tag{14}$$

where $\mathbf{p}$ is the prompt. The constraint embeds the target bit, while the maximum-probability objective minimizes the deviation from the model's preferred prediction. Once all carriers in block j have been filled, Dandelion commits both the tag and the realized carrier positions and token values to the chain state:

$$h_j = G_K\Big(n, j, t_j, R_j, \big((i_{j,r}, \widetilde{x}_0^{i_{j,r}})\big)_{r=1}^{R_j}\Big), \tag{15}$$

where G_K is domain-separated from F_K and g_K. Consequently, changing either a block payload or a realized carrier changes the state consumed by the next block. The updated block is written back to the denoising state before block $j+1$ is generated, so later blocks condition on the already-watermarked prefix rather than on the unmodified draft. If a completed block contains fewer safe positions than the configured capacity, Dandelion uses the available safe positions and records the reduced capacity. It never fills the deficit from an exact-content region. A final decode–tokenize consistency check repairs any carrier whose serialized form does not preserve its token IDs.

3.2 Watermark Detection

Detection operates directly on the received text and does not require the MDLM, its logits, the original output, or generation metadata. The detector first tokenizes the text into $\mathbf{z} = (z^1, \ldots, z^{n'})$, partitions it using the same block rule, and reconstructs $\mathcal{C}_j(\mathbf{z})$ from Eq. 11. Write the resulting ordered set as $\mathcal{C}_j(\mathbf{z}) = (i_{j,1}, \ldots, i_{j,\widehat{R}_j})$, where $\widehat{R}_j = |\mathcal{C}_j(\mathbf{z})|$. The detector removes these reconstructed carriers to recover $\mathcal{P}_j(\mathbf{z})$ and recomputes the chained block states using the shared key. For compactness, let $\widehat{\mathbf{c}}_j = \big((i_{j,r}, z^{i_{j,r}})\big)_{r=1}^{\widehat{R}_j}$ denote the ordered record of reconstructed carrier positions and token values. The block tag and the expected and observed bits are then obtained as follows:

$$\widehat{t}_j = F_K\Big(n', j, \mathcal{P}_j(\mathbf{z}), \widehat{h}_{j-1}\Big), b_{j,r}^{\mathrm{exp}} = \mathrm{bit}_r(\widehat{t}_j), b_{j,r}^{\mathrm{obs}} = g_K(n', j, r, z^{i_{j,r}}). \tag{16}$$

After checking the carrier bits, the detector updates the chain state by committing the reconstructed carrier record:

$$\widehat{h}_j = G_K\Big(n', j, \widehat{t}_j, \widehat{R}_j, \widehat{\mathbf{c}}_j\Big). \tag{17}$$

The expected bit is determined by the observed non-carrier content and the preceding block state, whereas the observed bit is determined by the token currently occupying the carrier position. The detector accepts the sequence only when every nonempty block contains a reconstructible carrier and all reconstructed carriers agree with their expected bits. The detector rejects immediately if either condition is violated. The decision rule is defined as follows:

$$\mathrm{Verify}_K(\mathbf{z}) = \mathbf{1}\Big\{b_{j,r}^{\mathrm{obs}} = b_{j,r}^{\mathrm{exp}},\ \forall j,\ r \in \{1, \ldots, \widehat{R}_j\}\Big\}. \tag{18}$$

A payload edit changes the recomputed block state and therefore its expected carrier bits. A carrier edit changes the observed keyed bit directly. Insertions, deletions, and truncations additionally alter the sequence length, block boundaries, or chained states. These operations therefore create one or more fresh comparisons and almost certainly cause verification to fail, apart from the keyed-collision and final-block boundary cases analyzed in the appendix. This allows Dandelion to verify integrity without retaining a reference hash for each generated sequence. Detailed proofs of completeness, tamper sensitivity, and embedding stability are provided in Appendix A, while the complete embedding and detection procedures are given in Appendix B.

Table 1: Integrity-verification performance across six MDLMs in the primary evaluation run. TPR is reported on intact watermarked samples, while TNR is reported for each perturbation. **Avg.** is the mean TNR over all perturbations.

Metric	LLaDA-8B Instruct	LLaDA-8B MoE	LLaDA-1.5	LLaDA-2.0-Flash	LLaDA-V	LaViDA
Intact						
TPR ↑	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
Perturbed (TNR)						
CC ↑	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
NC ↑	99.3 ± 0.5	99.6 ± 0.2	99.3 ± 0.5	99.6 ± 0.3	99.4 ± 0.4	99.8 ± 0.1
CS ↑	99.8 ± 0.1	99.8 ± 0.1	99.3 ± 0.5	99.8 ± 0.1	99.8 ± 0.1	99.3 ± 0.4
CI ↑	99.8 ± 0.1	99.7 ± 0.2	99.7 ± 0.1	99.9 ± 0.1	99.6 ± 0.1	99.7 ± 0.2
CD ↑	99.7 ± 0.1	99.8 ± 0.1	99.7 ± 0.3	99.8 ± 0.1	99.8 ± 0.2	99.9 ± 0.1
WC ↑	99.8 ± 0.1	99.8 ± 0.1	99.9 ± 0.0	99.9 ± 0.1	99.9 ± 0.0	99.8 ± 0.1
PC ↑	98.9 ± 0.4	99.9 ± 0.1	99.7 ± 0.2	99.7 ± 0.2	98.9 ± 0.3	99.9 ± 0.1
T ↑	97.9 ± 1.5	99.7 ± 0.1	99.2 ± 0.4	99.9 ± 0.1	99.1 ± 0.2	99.8 ± 0.4
BR ↑	99.8 ± 0.1	99.6 ± 0.1	98.9 ± 0.4	99.9 ± 0.1	99.4 ± 0.1	99.6 ± 0.4
UN ↑	99.9 ± 0.1	99.8 ± 0.1	99.2 ± 0.1	98.9 ± 0.7	99.8 ± 0.1	98.7 ± 0.7
CaC ↑	99.2 ± 0.5	99.4 ± 0.3	99.1 ± 0.5	99.9 ± 0.1	99.3 ± 0.4	99.2 ± 0.7
PI ↑	99.9 ± 0.1	99.8 ± 0.1	99.0 ± 0.5	99.7 ± 0.2	99.9 ± 0.0	99.3 ± 0.0
SI ↑	99.9 ± 0.0	99.8 ± 0.1	99.9 ± 0.1	97.9 ± 0.8	99.2 ± 0.4	98.2 ± 0.8
Avg. ↑	99.53	99.75	99.45	99.61	99.55	99.48

4 EXPERIMENTS

4.1 EVALUATION CRITERIA.

Dandelion is designed both to authenticate intact watermarked outputs and to reject outputs that have been modified. Accordingly, we evaluate the true-positive rate (TPR) on intact watermarked samples and the true-negative rate (TNR) on perturbed samples. To assess sensitivity to realistic content alterations, we consider 13 perturbation types: carrier-token modification (CC), non-carrier-token modification (NC), character substitution (CS), character insertion (CI), character deletion (CD), whitespace modification (WC), punctuation modification (PC), truncation (T), block reordering (BR), Unicode normalization (UN), case modification (CaC), prefix insertion (PI), and suffix insertion (SI). Table 1 summarizes the corresponding integrity-verification results. Because perturbed-sample TNR does not measure accidental acceptance of naturally unwatermarked text, we report a separate control-set analysis in Appendix C.

4.2 SETUP

We evaluate Dandelion on six representative masked diffusion language models that span dense and sparse architectures, different post-training strategies, and text-only and multimodal settings. This model suite allows us to examine whether the watermark remains effective across distinct MDLM generation paradigms rather than being specialized to a single backbone.

LLaDA-8B-Instruct Nie et al. (2025): A dense MDLM trained from scratch via masked diffusion. It serves as our primary backbone for evaluating Dandelion under standard blockwise sampling.

LLaDA-8B-MoE Zhu et al. (2025): A sparse mixture-of-experts variant. We use it to test whether expert routing and sparse parameter activation impact watermark stability and detection.

LLaDA-1.5 Zhu et al. (2026): Enhanced via preference optimization, this model assesses Dandelion's robustness to distribution shifts introduced by post-training alignment.

Table 2: Robustness of Dandelion under different generation lengths and block sizes in an independent run using a random seed distinct from that of Table 1. Each entry reports the average **TNR (%)** across all perturbation types.

	Generation Length								
Model	**128**			**256**			**512**		
Block Size	**16**	**32**	**64**	**16**	**32**	**64**	**16**	**32**	**64**
LLaDA-8B-Instruct	99.4	99.6	98.5	99.6	99.5	99.5	99.7	99.6	99.1
LLaDA-8B-MoE	99.8	99.4	99.0	99.9	99.3	99.7	99.9	99.7	98.7
LLaDA-1.5	99.9	99.7	99.2	99.5	99.5	99.4	99.8	99.3	98.6
LLaDA-2.0-Flash	99.6	99.8	99.3	99.7	99.4	99.6	99.6	99.3	98.9
LLaDA-V	99.9	99.7	99.4	99.9	99.5	99.5	99.8	99.8	99.3
LaViDA	99.8	99.6	99.4	99.7	98.9	99.4	99.5	99.2	98.7

LLaDA-2.0-Flash Bie et al. (2025): Integrates autoregressive priors into masked diffusion. It tests Dandelion's generalization to models initialized with causal knowledge rather than pure diffusion objectives.

LLaDA-V You et al. (2026): A multimodal extension of LLaDA. It evaluates whether the watermark remains detectable when text generation is jointly conditioned on visual and textual inputs.

LaViDA Li et al. (2025): An independently developed multimodal MDLM. It provides a cross-family test to ensure Dandelion transfers beyond the LLaDA lineage while retaining its text-only verification property.

Generation and perturbation settings. We sample 1,000 prompts from the HC3 dataset (Guo et al., 2023) for each model and generate one response for each prompt. Dandelion embeds a separate watermark into every generated response before the integrity evaluation. For the results in Table 1, we use a fixed block size of 64, a generation length of 256, and four carrier positions per block. The sampler reveals one token at each denoising step. This configuration was selected empirically as a practical compromise between generation quality and sampling efficiency. For every intact watermarked (positive) sample, we construct 1,300 perturbed instances by applying each of the 13 perturbation types 100 times. For each TNR entry in Table 1, the value after $\pm$ is the standard deviation over these 100 perturbation repetitions; intact TPR is evaluated on the 1,000 unperturbed watermarked responses. Table 2 is obtained from a second, independent experimental run with a different fixed random seed. It evaluates the effect of generation length and block size while retaining the same non-seed settings, perturbation protocol, and number of perturbation evaluations as Table 1. The prompts and perturbation instances are resampled in this run; consequently, its $(256, 64)$ entries are independent replications of, rather than reused values from, the primary evaluation. All experiments are conducted on a single NVIDIA H100 GPU with 96 GB of memory. A paired evaluation of the quality difference between watermarked and unwatermarked responses is provided in Appendix D, followed by a systematic runtime overhead evaluation in Appendix E.

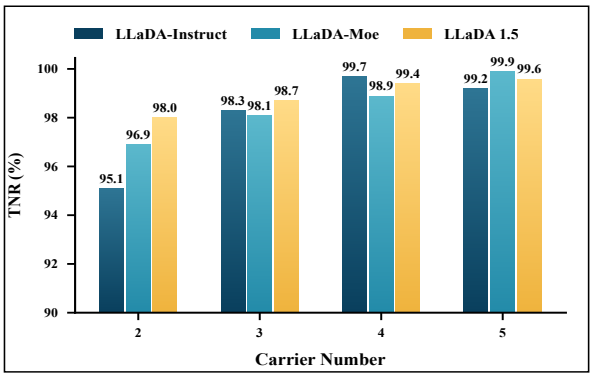

Figure 2: The performance of Dandelion in different quantities of carriers.

4.3 Main Results

As shown in Table 1, Dandelion achieves a true positive rate (TPR) of 100% for intact watermarked samples across all six evaluated models. These results demonstrate its consistent effectiveness across diverse model architectures, including models with multimodal capabilities and MDLMs adapted from autoregressively pretrained models. Moreover, when averaged across the 13 perturbation types, the true negative rate (TNR) exceeds 99% for every model, indicating that Dandelion reliably identifies tampered samples. The few false positives arise primarily from two cases: (1) model generation failures, such as abnormally short responses or garbled text; and (2) random perturbations that modify a carrier in the final autoregressive block but coincidentally encode the same bit as the original carrier. Representative examples and case-specific explanations are provided in Appendix F. The final-block boundary and its implications for the intended use of Dandelion are discussed in Appendix G.

To evaluate Dandelion under a broader range of generation settings, we use six MDLMs as the backbone models and vary the block size over {16, 32, 64} and the generation length over {128, 256, 512} tokens. As shown in Table 2, Dandelion maintains a high average TNR across nearly all evaluated configurations, demonstrating its reliability under varying block sizes and generation lengths. Appendix E systematically quantifies the additional runtime introduced by watermark embedding.

4.4 Ablation Study

In this section, we investigate how Dandelion's configuration affects detection performance by evaluating the impact of **the number and placement of watermark carriers within each block**.

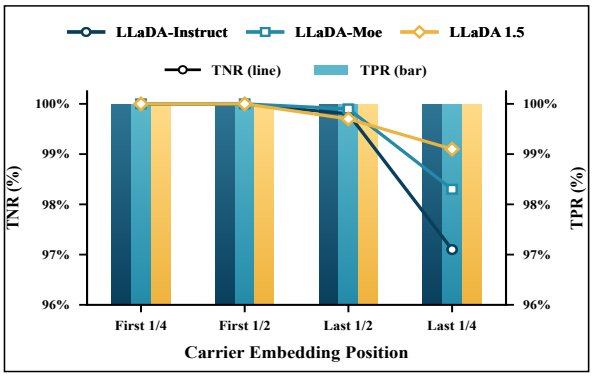

Figure 3: The performance of Dandelion when dealing with different perturbation embedding positions.

Different Carriers. In Figure 2, we selected LLaDA-8B-Instruct, LLaDA-8B-Moe and LLaDA-1.5 as the baseline models, and chose {2, 3, 4, 5} as the number of watermark carriers included in each block. The experiment shows that for any carrier quantity greater than 1, Dandelion can almost always detect the corresponding positive samples. However, conversely, when the number of carriers is too small, additional false positive samples may be generated. This is because the original position of the sequence coincides exactly with the position of the carrier.

Position of Carriers. In Figure 3, we analyze the primary cause of the misclassified samples observed in our experiments. Nearly all errors result from perturbations occurring in the final block of the sequence. When the perturbation affects the last block, the detector cannot perform cross-block verification using subsequent blocks and must rely solely on the carrier information in the affected block. Consequently, if the perturbation coincides with a carrier position and happens to produce the same verification bit as the original carrier, the detector cannot identify the modification.

5 Conclusion

In this paper, we introduce **Dandelion**, the first fragile watermarking framework specifically designed for masked diffusion language models. By binding watermark bits through sequential block states, Dandelion enables reference-free integrity verification. Across text-only and multimodal MDLMs, it achieves a 100% true-positive rate on intact outputs and a watermark invalidation rate above 99% under 13 perturbation types. Future work will extend Dandelion to broader MDLM generation settings and content types.

LARGE LANGUAGE MODEL USAGE STATEMENT

Large language models were used to assist with code implementation and language polishing. The authors reviewed and verified all generated content and take full responsibility for the paper.

REFERENCES

Zulfiqar Ali, Fazal-e Amin, and Muhammad Hussain. A novel fragile zero-watermarking algorithm for digital medical images. *Electronics*, 11(5), 2022.

Marianne Arriola, Subham Sahoo, Aaron Gokaslan, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Justin Chiu, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. *ICLR 2025*, 2025.

M Barni, F Bartolini, V Cappellini, and A Piva. A dct-domain system for robust image watermarking. *Signal Processing*, 66(3):357–372, 1998.

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. *NeurIPS 2025*, 2025.

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, Chongxuan Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Xiaocheng Lu, Yuxin Ma, Jianfeng Tan, Lanning Wei, Ji-Rong Wen, Yipeng Xing, Xiaolu Zhang, Junbo Zhao, Da Zheng, Jun Zhou, Junlin Zhou, Zhanchao Zhou, Liwang Zhu, and Yihong Zhuang. Llada2.0: Scaling up diffusion language models to 100b. *CoRR*, abs/2512.15745, 2025.

Marco Botta, Davide Cavagnino, and Roberto Esposito. Neunac: A novel fragile watermarking algorithm for integrity protection of neural networks. *Information Sciences*, 576:228–241, 2021.

Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, Jamie Hayes, Nidhi Vyas, Majd Al Merey, Jonah Brown-Cohen, Rudy Bunel, Borja Balle, Taylan Cemgil, Zahra Ahmed, Kitty Stacpoole, Ilia Shumailov, Ciprian Baetu, Sven Gowal, Demis Hassabis, and Pushmeet Kohli. Scalable watermarking for identifying large language model outputs. *Nature*, 634 (8035):818–823, 2024.

Zhenzhe Gao, Yu Cheng, and Zhaoxia Yin. A survey of fragile model watermarking. *Signal Processing*, 238:110088, 2026. ISSN 0165-1684. doi: https://doi.org/10.1016/j.sigpro.2025.110088. URL https://www.sciencedirect.com/science/article/pii/S0165168425002026.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. Diffuseq: Sequence to sequence text generation with diffusion models. *ICLR 2023*, 2023.

Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *arXiv preprint arxiv:2301.07597*, 2023.

Kangli Hao, Guorui Feng, and Xinpeng Zhang. Robust image watermarking based on generative adversarial network. *China Communications*, 17(11):131–140, 2020.

Zhengfu He, Tianxiang Sun, Qiong Tang, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. Diffusionbert: Improving generative masked language models with diffusion models. *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics, ACL 2023, Vol 1*, pp. 4521–4534, 2023.

Nikola Jovanović, Robin Staab, and Martin Vechev. Watermark stealing in large language models. *ICML 2024*, 2024.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. *ICML 2023*, 2023.

Shufan Li, Konstantinos Kallidromitis, Hritik Bansal, Akash Gokul, Yusuke Kato, Kazuki Kozuka, Jason Kuen, Zhe Lin, Kai-Wei Chang, and Aditya Grover. Lavida: A large diffusion language model for multimodal understanding. 2025.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori Hashimoto. Diffusion-lm improves controllable text generation. *NEURIPS 2022*, 2022.

ET Lin, CI Podilchuk, and EJ Delp. Detection of image alterations using semifragile watermarks. *Security and Watermarking of Multimedia Contents Ii*, 3971:152–163, 2000.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, JUN ZHOU, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *NeurIPS 2025*, 2025.

Yiran Peng, Qingqing Hu, Jing Xu, U. KinTak, and Junming Chen. A novel deep learning zero-watermark method for interior design protection based on image fusion. *Mathematics*, 13(6), 2025.

Subham Sekhar Sahoo, Marianne Arriola, Aaron Gokaslan, Edgar Mariano Marroquin, Alexander M Rush, Yair Schiff, Justin T Chiu, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *NeurIPS 2024*, 2024.

Aditya Kumar Sahu, M. Hassaballah, Routhu Srinivasa Rao, and Gulivindala Suresh. A logistic map based blind and fragile watermarking for tamper detection and localization in images. *Multimedia Tools and Applications*, 82(16):24069–24100, 2022.

Dor Tsur, Carol Xuan Long, Claudio Mayrink Verdun, Hsiang Hsu, Haim Permuter, and Flavio P. Calmon. Optimized couplings for watermarking large language models. *2025 IEEE International Symposium on Information Theory (ISIT)*, pp. 1–6, 2025a.

Dor Tsur, Carol Xuan Long, Claudio Mayrink Verdun, Sajani Vithana, Hsiang Hsu, Chun-Fu Chen, Haim H. Permuter, and Flavio Calmon. Heavywater and simplexwater: Distortion-free llm watermarks for low-entropy distributions. *NeurIPS 2025*, 2025b.

Cheng-Hin Un and Ka-Cheng Choi. Deep watermarking based on swin transformer for deep model protection. *Applied Sciences-basel*, 15(10), 2025.

Wenbo Wan, Jun Wang, Yunming Zhang, Jing Li, Hui Yu, and Jiande Sun. A comprehensive survey on robust image watermarking. *Neurocomputing*, 488:226–247, 2022.

Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. Tree-ring watermarks: Fingerprints for diffusion images that are invisible and robust. *NEURIPS 2023*, 2023.

Qilong Wu and Varun Chandrasekaran. Bypassing llm watermarks with color-aware substitutions. *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, Vol 1 Long Papers*, pp. 8549–8581, 2024.

Zhihao Wu, Gracia Gong, Qinglin Zhu, Yudong Chen, and Runcong Zhao. Linear ensembles wash away watermarks: on the fragility of distributional perturbations in llms. *ICML 2026*, 2026.

Zebin You, Shen Nie, Xiaolu Zhang, JUN ZHOU, Zhiwu Lu, Ji-Rong Wen, and Chongxuan Li. Llada-v: Large language diffusion models with visual instruction tuning. *CVPR 2026*, 2026.

Zihan Yuan, Li Li, Zichi Wang, and Xinpeng Zhang. Integrity protection of generative adversarial networks using fragile watermarking. *ACM Transactions on Multimedia Computing Communications and Applications*, 21(12), 2025.

Xuanyu Zhang, Runyi Li, Jiwen Yu, Youmin Xu, Weiqi Li, and Jian Zhang. Editguard: Versatile image watermarking for tamper localization and copyright protection. *CVPR 2024*, 2024.

Lei Zhao, Han Wang, Tao Zhang, and Liming Xu. A sentence-semantic-based watermark for large language models against paraphrasing attacks. *2025 10th International Conference on Information Science, Computer Technology and Transportation (ISCTT)*, pp. 61–66, 2025a.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. D1: Scaling reasoning in diffusion large language models via reinforcement learning. *NeurIPS 2025*, 2025b.

Xuandong Zhao, Prabhanjan Vijendra Ananth, Lei Li, and Yu-Xiang Wang. Provable robust watermarking for ai-generated text. *ICLR 2024*, 2024a.

Xuandong Zhao, Yu-Xiang Wang, and Lei Li. Watermarking for large language models. *Annual Meeting of the Association for Computational Linguistics*, pp. 10–11, 2024b.

Fengqi Zhu, Zebin You, Yipeng Xing, Zenan Huang, Lin Liu, Yihong Zhuang, Guoshan Lu, Kangyu Wang, Xudong Wang, Lanning Wei, Hongrui Guo, Jiaqi Hu, Wentao Ye, Tieyuan Chen, Chenchen Li, Chengfu Tang, Haibo Feng, Jun Hu, Jun Zhou, Xiaolu Zhang, Zhenzhong Lan, Junbo Zhao, Da Zheng, Chongxuan Li, Jianguo Li, and Ji-Rong Wen. Llada-moe: A sparse moe diffusion language model. *CoRR*, abs/2509.24389, 2025.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. *ACL 2026*, pp. 11425–11460, 2026.

A Detailed Proof

This section establishes the completeness, tamper sensitivity, and embedding stability of Dandelion. The guarantees are stated for the canonical token sequence observed by the detector. This distinction is necessary because two different byte strings may be mapped to the same token sequence by a normalizing tokenizer; no token-level watermark can distinguish such a pair. Let $L = \lceil n/B \rceil$ be the number of blocks and let $R_j = |\mathcal{C}_j|$ be the number of carriers in block j.

A.1 Security Model and Assumptions

We use the following standard assumptions.

Assumption 1 (Deterministic reconstruction)**.** *The tokenizer, block partition, strict-region scan, and* Select *rule are deterministic and identical during embedding and detection. A returned output satisfies* $\mathrm{Tok}(\mathrm{Dec}(\widetilde{\mathbf{x}}_0)) = \widetilde{\mathbf{x}}_0$.

Assumption 2 (Embedding admissibility)**.** *Every nonempty protected block contains at least one reconstructible carrier. For each carrier (j, r), the admissible candidate set $\mathcal{A}_{j,r}$ contains a token whose keyed carrier value equals the target bit. If either condition is violated, the embedder returns failure rather than an unauthenticated sequence.*

Assumption 3 (Keyed-function security)**.** *Under an unknown key K, the domain-separated functions F_K, G_K, and g_K are computationally indistinguishable from independent random functions. Their inputs are encoded unambiguously, including the sequence length, block index, carrier index, and token identities.*

Assumption 1 is enforced by the final decode–tokenize repair. Assumption 2 is an explicit liveness condition rather than a security assumption: Dandelion does not claim to protect an output for which no bit can be embedded. The final assumption is the usual computational model for keyed authentication.

A.2 Correctness on Intact Outputs

Theorem 4 (Perfect completeness)**.** *If Dandelion returns a watermarked sequence $\widetilde{\mathbf{x}}_0$ under Assumptions 1 and 2, then the text-only detector accepts it:*

$$\mathrm{Verify}_K(\mathrm{Dec}(\widetilde{\mathbf{x}}_0)) = 1. \tag{19}$$

Proof. By Assumption 1, tokenizing the received text recovers exactly $\widetilde{\mathbf{x}}_0$. Consequently, the detector recovers the same length, block boundaries, strict regions, ordered carrier sets, and payloads as the embedder. We prove equality of the chain states by induction over the block index.

For block 0, both procedures initialize the preceding state to $h_{-1} = \widehat{h}_{-1} = \epsilon$. They therefore evaluate F_K on the same tuple $(n, 0, \mathcal{P}_0, \epsilon)$ and obtain $t_0 = \widehat{t}_0$. During embedding, every selected

token is constrained to satisfy $g_K(n, 0, r, \widetilde{x}_0^{i_0,r}) = \mathrm{bit}_r(t_0)$. Thus, all observed and expected bits in block 0 agree. Both procedures then evaluate G_K on the same tag, carrier count, positions, and realized token identities, which gives $h_0 = \widehat{h}_0$.

Assume $h_{j-1} = \widehat{h}_{j-1}$. The reconstructed payload of block j is identical to the embedded payload, so Eq. 13 gives $t_j = \widehat{t}_j$. The constrained choice in Eq. 14 makes every carrier comparison pass, and Eq. 15 then gives $h_j = \widehat{h}_j$. Induction proves the claim for all L blocks, after which Eq. 18 returns one. □

Theorem 4 explains why the intact-sample detection rate does not depend on a statistical threshold: once embedding and canonical serialization succeed, acceptance follows deterministically.

A.3 Tamper Sensitivity and Chain Amplification

Let $\widetilde{\mathbf{x}}_0$ be an authentic token sequence and let $\mathbf{z} \neq \widetilde{\mathbf{x}}_0$ be a key-independent tampered sequence. For a comparison (j, r) performed on $\mathbf{z}$, call the comparison *fresh* if either the input used to compute its expected bit or the input used to compute its observed bit differs from the corresponding input for the authentic sequence. Let $M(\mathbf{z})$ denote the number of fresh comparisons. This quantity makes the scope of the guarantee explicit: an edit is detectable only if it changes at least one authenticated input.

Theorem 5 (False-acceptance bound)**.** *Under Assumption 3, the probability that a fixed tampered sequence with $M(\mathbf{z}) \geq 1$ is accepted is bounded by:*

$$\Pr[\mathrm{Verify}_K(\mathbf{z}) = 1] \leq 2^{-M(\mathbf{z})} + \mathrm{Adv}_{\mathrm{PRF}}\,. \tag{20}$$

For at most q independently submitted tampering attempts, the union bound is $q\,2^{-M_{\min}} + \mathrm{Adv}_{\mathrm{PRF}}$, where $M_{\min}$ is the minimum number of fresh comparisons among the attempts.

Proof. Replace the three domain-separated keyed functions with independent random functions. In every fresh comparison, at least one of the two compared bits is evaluated at a new input. Conditioned on the complete authentic transcript and all preceding comparisons, that bit is uniform and independent of the other bit. Hence, a fresh comparison succeeds with probability $1/2$. Distinct carrier indices use distinct inputs, and different bits of a random block tag are jointly uniform; therefore, all $M(\mathbf{z})$ fresh comparisons succeed with probability $2^{-M(\mathbf{z})}$. Replacing the random functions with their keyed implementations changes this probability by at most the PRF distinguishing advantage. The multi-attempt result follows by a union bound. □

The chain state makes an early edit affect later blocks even if their visible tokens remain unchanged. The resulting amplification is formalized below.

Corollary 6 (Blockwise chain amplification)**.** *Suppose the block partition and carrier layout are preserved and block q is the first altered block. Except with negligible keyed-function collision probability, the following bounds hold:*

1. *If the payload of block q changes, then $M(\mathbf{z}) \geq \sum_{j=q}^{L-1} R_j$.*
2. *If only s_q realized carriers in block q change while its payload is preserved, then $M(\mathbf{z}) \geq s_q + \sum_{j=q+1}^{L-1} R_j$.*

Proof. A payload edit changes the input to F_K in block q, making all R_q expected-bit comparisons fresh. It also changes t_q and hence the committed state h_q. The altered state is an input to F_K in block $q+1$, and this argument repeats through the final block. Thus, every carrier comparison from block q onward is fresh.

For a carrier-only edit, the tag t_q is unchanged because carriers are excluded from $\mathcal{P}_q$. Each of the s_q modified carrier identities, however, changes an input to g_K and therefore creates a fresh comparison. Because G_K commits the realized carriers, the edit also changes h_q and makes every comparison in each later block fresh. □

Corollary 6 also identifies the intrinsic final block limitation observed in our experiments. A single carrier substitution in the final block creates only one fresh comparison and therefore collides with probability $1/2$; no later block remains to amplify the evidence. In contrast, the same edit in an earlier block is checked again indirectly by all subsequent carriers. Insertions, deletions, and truncations usually modify the committed length, block partition, or multiple chained inputs and therefore create a larger $M(\mathbf{z})$.

The bound is intentionally conditional on $M(\mathbf{z}) \geq 1$. Dandelion cannot distinguish byte-level changes erased by tokenization. Moreover, both embedding and detection reject a nonempty block without a reconstructible carrier, preventing an adversary from turning a protected block into an empty verification condition. This fail-closed behavior is necessary for the coverage claims above.

A.4 Embedding and Serialization Stability

For a carrier (j, r), order the $k_{j,r}$ admissible candidates by their model probabilities as $v_1, \ldots, v_{k_{j,r}}$, with $p_\theta(v_1 \mid \overline{\mathbf{x}}_{j,r}, \mathbf{p}) \geq \cdots \geq p_\theta(v_{k_{j,r}} \mid \overline{\mathbf{x}}_{j,r}, \mathbf{p})$. Let $D_{j,r}$ be the rank of the first candidate whose keyed value equals the target bit, with $D_{j,r} = \infty$ if no such candidate exists.

Proposition 7 (Candidate availability and rank)**.** *Under Assumption 3, the candidate search obeys:*

$$\Pr[D_{j,r} = d] = 2^{-d}, \quad \Pr[D_{j,r} > k_{j,r}] = 2^{-k_{j,r}}, \tag{21}$$

for $1 \leq d \leq k_{j,r}$*. Conditioned on a successful search, its expected selected rank is:*

$$\mathbb{E}[D_{j,r} \mid D_{j,r} \leq k_{j,r}] = \frac{2 - (k_{j,r} + 2)2^{-k_{j,r}}}{1 - 2^{-k_{j,r}}} < 2. \tag{22}$$

Across all carriers, the probability that at least one candidate search fails is at most $\sum_{j,r} 2^{-k_{j,r}}$.

Proof. The target bit is independent of the domain-separated carrier values. Each candidate matches it with probability $1/2$. Selecting rank d requires $d - 1$ consecutive mismatches followed by one match, which has probability 2^{-d}; failure requires $k_{j,r}$ mismatches. Summing the truncated geometric distribution and conditioning on success yields Eq. 22. The final statement follows by a union bound over carrier searches. □

For the realized rank $D_{j,r}$, the local log-probability cost is defined as follows:

$$\Delta_{j,r} = \log p_\theta(v_1 \mid \overline{\mathbf{x}}_{j,r}, \mathbf{p}) - \log p_\theta(v_{D_{j,r}} \mid \overline{\mathbf{x}}_{j,r}, \mathbf{p}) \geq 0. \tag{23}$$

Equation 14 minimizes $\Delta_{j,r}$ among all admissible candidates carrying the required bit. Thus, Dandelion is locally optimal for each fixed masked context, while Proposition 7 shows that the selected token has expected rank below two when the candidate bits are pseudorandom. This is a likelihood-based stability statement, not an unconditional semantic guarantee; the deterministic exclusion of code, mathematics, numeric values, and unstable token boundaries addresses cases in which even a high-probability replacement could change exact meaning.

Proposition 8 (Serialization stability)**.** *If the embedding procedure returns a sequence only after* $\mathrm{Tok}(\mathrm{Dec}(\widetilde{\mathbf{x}}_0)) = \widetilde{\mathbf{x}}_0$*, then transporting the output as decoded text does not alter its carrier positions, payloads, or verification result.*

Proof. The round trip recovers the identical token sequence. All downstream objects, including block boundaries, strict regions, carrier sets, payloads, tags, and chain states, are deterministic functions of that sequence and the shared configuration. They are therefore unchanged, and Theorem 4 applies. □

Algorithm 1 Online Dandelion embedding

Require: MDLM p_θ, prompt $\mathbf{p}$, key K, length n, block size B, carrier budget R
Ensure: Canonical watermarked sequence $\widetilde{\mathbf{x}}_0$ or failure

Initialize the generated suffix with masks and set $h_{-1} \leftarrow \epsilon$
for $j = 0, \ldots, \lceil n/B \rceil - 1$ **do**
  Denoise positions $\mathcal{I}_j$ using p_θ and the current prefix
  $\mathcal{E}_j \leftarrow \mathrm{Eligible}(\mathbf{x}_0, \mathcal{I}_j)$
  $\mathcal{C}_j \leftarrow \mathrm{Select}(\mathcal{E}_j, R)$
  if $\mathcal{C}_j = \varnothing$ **then**
    return failure
  end if
  $\mathcal{P}_j \leftarrow \mathbf{x}_0[\mathcal{I}_j \setminus \mathcal{C}_j]$
  $t_j \leftarrow F_K(n, j, \mathcal{P}_j, h_{j-1})$
  for $r = 1, \ldots, |\mathcal{C}_j|$ **do**
    $i \leftarrow i_{j,r}$; $b \leftarrow \mathrm{bit}_r(t_j)$
    Mask i and obtain $p_\theta(\cdot \mid \overline{\mathbf{x}}_{j,r}, \mathbf{p})$
    $\mathcal{A}_{j,r} \leftarrow \mathrm{Candidates}(i, \mathbf{x}_0)$
    Remove candidates that change the carrier's lexical signature
    $\mathcal{A}^b_{j,r} \leftarrow \{v \in \mathcal{A}_{j,r} : g_K(n, j, r, v) = b\}$
    if $\mathcal{A}^b_{j,r} = \varnothing$ **then**
      return failure
    end if
    $\widetilde{x}_0^i \leftarrow \arg\max_{v \in \mathcal{A}^b_{j,r}} p_\theta(v \mid \overline{\mathbf{x}}_{j,r}, \mathbf{p})$
  end for
  $h_j \leftarrow G_K(n, j, t_j, |\mathcal{C}_j|, ((i_{j,r}, \widetilde{x}_0^{i_{j,r}}))_r)$
  Write block j back before denoising block $j+1$
end for
$\widetilde{\mathbf{x}}_0 \leftarrow \mathrm{CanonicalRepair}(\widetilde{\mathbf{x}}_0)$
if repair fails or $\mathrm{Verify}_K(\widetilde{\mathbf{x}}_0) = 0$ **then**
  return failure
end if
return $\widetilde{\mathbf{x}}_0$

B Algorithm Implementation

Dandelion is implemented as an online callback to blockwise MDLM generation. The current block is first completed by the model's original denoising schedule. Before denoising the next block, the callback selects the carriers, re-scores each carrier under a leave-one-out mask, writes the constrained token back to the generation state, and commits the realized block state. Therefore, every later block is generated conditionally on the already-watermarked prefix. Algorithms 1 and 2 give the complete model-level procedures.

The eligibility routine decodes the current token sequence and maps token positions to character spans. It removes special tokens, pieces without a stable leading word boundary, number-initial pieces, and tokens overlapping deterministically recognized code or mathematical regions. The remaining positions are traversed in a fixed order, so generation and text-only detection select the same prefix of eligible positions. A strict-heavy block may use fewer than R carriers, but a nonempty block with no carrier causes a fail-closed return. The candidate routine preserves the original leading whitespace and lexical signature. During final repair, it additionally retains only candidates whose decoded complete sequence tokenizes back to the same IDs.

During the online callback, future blocks are still masked, so their final token boundaries are not yet available. After all blocks are complete, $\mathrm{CanonicalRepair}$ decodes and re-tokenizes the full sequence. If its IDs change, the routine recomputes the carrier layout, tags, and chained states on the canonical IDs and re-embeds all carriers. It repeats until the token IDs reach a fixed point; a bounded nonconvergent repair is reported as a generation failure. The implementation also performs an immediate self-check on both the in-memory IDs and the serialized text before returning an output.

Algorithm 2 Text-only Dandelion detection

Require: Received text y, tokenizer Tok, key K, block size B, carrier budget R
Ensure: Integrity decision in $\{0, 1\}$

$\mathbf{z} \leftarrow \mathrm{Tok}(y)$; $n' \leftarrow |\mathbf{z}|$; $\widehat{h}_{-1} \leftarrow \epsilon$
Partition $\mathbf{z}$ into blocks $\{\mathcal{I}_j\}$ of size B
for $j = 0, \ldots, \lceil n'/B \rceil - 1$ **do**
  $\widehat{\mathcal{C}}_j \leftarrow \mathrm{Select}(\mathrm{Eligible}(\mathbf{z}, \mathcal{I}_j), R)$
  if $\widehat{\mathcal{C}}_j = \varnothing$ **then**
    return 0
  end if
  $\widehat{\mathcal{P}}_j \leftarrow \mathbf{z}[\mathcal{I}_j \setminus \widehat{\mathcal{C}}_j]$
  $\widehat{t}_j \leftarrow F_K(n', j, \widehat{\mathcal{P}}_j, \widehat{h}_{j-1})$
  for $r = 1, \ldots, |\widehat{\mathcal{C}}_j|$ **do**
    $i \leftarrow i_{j,r}$
    if $g_K(n', j, r, z^i) \neq \mathrm{bit}_r(\widehat{t}_j)$ **then**
      return 0
    end if
  end for
  $\widehat{h}_j \leftarrow G_K(n', j, \widehat{t}_j, |\widehat{\mathcal{C}}_j|, ((i_{j,r}, z^{i_{j,r}}))_r)$
end for
return 1

Detection requires only tokenization, deterministic surface-syntax scanning, and keyed-function evaluations; it does not query the MDLM or retain the original response. For n' received tokens and $R_{\mathrm{tot}} = \sum_j \widehat{R}_j$ reconstructed carriers, its cost is $O(n' + R_{\mathrm{tot}})$ excluding tokenizer internals, and its working memory is $O(n')$. A successful embedding pass adds one masked-model scoring call per carrier to the original blockwise generation, plus any bounded canonical repair passes. Candidate lists indexed by lexical signature can be cached across carriers without changing the algorithm or its verification result.

C False-Acceptance Analysis on Unwatermarked Text

Detection on perturbed watermarked text measures tamper sensitivity but does not by itself measure whether naturally unwatermarked text is incorrectly authenticated. We therefore construct a control set $\mathcal{D}_{\mathrm{ctrl}}$ containing unwatermarked responses from each evaluated MDLM and human answers from HC3. The false-acceptance rate (FAR) is

$$\mathrm{FAR} = \frac{1}{|\mathcal{D}_{\mathrm{ctrl}}|} \sum_{\mathbf{z} \in \mathcal{D}_{\mathrm{ctrl}}} \mathbf{1}\{\mathrm{Verify}_K(\mathbf{z}) = 1\}\,. \tag{24}$$

We report FAR separately for human text and for each unwatermarked model. Verification failures caused by an unusable tokenization or the absence of eligible carriers are recorded separately from ordinary watermark mismatches.

Table 3 reports the control-set results. Both failure columns contain counts and are included in N; because the detector fails closed, these samples are not counted as false acceptances.

D Generation Quality Analysis

To quantify the quality cost of watermark embedding, we compare each watermarked response with an unwatermarked response generated from the same prompt under identical decoding settings. For an MDLM response $\mathbf{x} = (x^1, \ldots, x^n)$ and prompt $\mathbf{p}$, we measure masked pseudo-perplexity (MPPL) by masking and predicting one token at a time:

Table 3: False-acceptance analysis on naturally unwatermarked text. N is the number of control samples, and #Accepted counts samples incorrectly accepted as watermarked. FAR is #Accepted divided by N. "Tok. fail" denotes an unusable canonical tokenization, and "No carrier" denotes a sample with no eligible carrier position.

Control source	N	**#Accepted**	**FAR (%)**	**Tok. fail**	**No carrier**
Human answers (HC3)	1000	0	0.0	0	0
LLaDA-8B-Instruct	1000	0	0.0	0	0
LLaDA-8B-MoE	1000	0	0.0	0	0
LLaDA-1.5	1000	0	0.0	0	0
LLaDA-2.0-Flash	1000	0	0.0	0	0
LLaDA-V	1000	0	0.0	0	0
LaViDA	1000	0	0.0	0	0
All controls	**7000**	**0**	**0.0**	**0**	**0**

Table 4: Generation quality before and after Dandelion embedding. Δ MPPL denotes the relative increase over the corresponding unwatermarked response. Semantic Similarity and Correctness Retention measure content preservation. Reported values are means over paired responses, and the average row is the macro-average across all six models.

	Masked Pseudo-Perplexity			**Content Preservation**	
Model	**Unwatermarked** $\downarrow$	**Dandelion** $\downarrow$	**Δ (%)** $\downarrow$	**Semantic Sim. (%)** $\uparrow$	**Correctness Ret. (%)** $\uparrow$
LLaDA-8B-Instruct	1.19	1.98	66.39	98.1	100.0
LLaDA-8B-MoE	1.43	1.89	32.17	99.3	100.0
LLaDA-1.5	1.38	1.81	31.16	99.1	100.0
LLaDA-2.0-Flash	1.15	1.31	13.91	98.7	100.0
LLaDA-V	1.92	2.23	16.15	99.6	100.0
LaViDA	1.85	2.64	42.70	99.4	100.0
Average	**1.49**	**1.98**	**33.75**	**99.03**	**100.0**

$$\mathrm{MPPL}(\mathbf{x} \mid \mathbf{p}) = \exp\left(-\frac{1}{n}\sum_{i=1}^{n} \log p_\theta\Big(x^i \mid \mathbf{x}^{(i\leftarrow m)}, \mathbf{p}\Big)\right), \tag{25}$$

where $\mathbf{x}^{(i\leftarrow m)}$ denotes the sequence obtained by replacing position i with the mask token m. We report MPPL for both the unwatermarked and watermarked responses, together with the relative MPPL increase $100\big(\mathrm{MPPL}(\widetilde{\mathbf{x}}_0 \mid \mathbf{p})/\,\mathrm{MPPL}(\mathbf{x}_0 \mid \mathbf{p}) - 1\big)$. Semantic similarity (embedded by CLIP) and correctness retention (reported by human) are evaluated separately so that a small likelihood change is not treated as sufficient evidence of preserved meaning. Correctness retention is the ratio between the task-level scores of the watermarked and unwatermarked responses. All metrics are computed on paired responses generated from the same prompts and decoding settings. Table 4 summarizes the reporting format.

E Watermark Embedding Overhead

We evaluate the time required to embed Dandelion using the same prompts, generation settings, and hardware as the main experiments. For each prompt, we measure the latency T_{base} of standard MDLM generation and the latency T_{wm} of generation with online watermark embedding. The watermarked latency includes blockwise carrier selection, masked-token rescoring, constrained candidate selection, canonical serialization repair, and the final self-verification. It excludes model loading, prompt tokenization, output serialization, and the separate unwatermarked draft used only to con-

Table 5: Runtime overhead of Dandelion watermark embedding. Base denotes standard MDLM generation, while Dandelion latency measures the online watermarked generation stage, including canonical repair and self-verification. Latencies are reported as mean seconds per response, and token counts denote the mean number of generated tokens per successful response. The average row reports the macro-average across all six models.

Model	Base Latency (s) ↓	Dandelion Latency (s) ↓	ΔT (s) ↓	Overhead (%) ↓	Base Tokens	Dandelion Tokens
LLaDA-8B-Instruct	16.984	17.973	0.989	5.82	209.13	209.97
LLaDA-8B-MoE	16.733	17.991	1.258	7.51	226.30	226.12
LLaDA-1.5	17.531	18.777	1.246	7.11	224.63	224.68
LLaDA-2.0-Flash	18.521	19.226	0.705	3.80	220.38	220.12
LLaDA-V	20.126	20.985	0.859	4.26	200.32	200.98
LaViDA	20.672	21.503	0.831	4.01	213.10	210.45
Average	**18.428**	**19.409**	**0.981**	**5.42**	**215.64**	**215.39**

Table 6: Counts of accepted perturbed cases in the 1,000-sample primary evaluation, grouped by failure category. Each case is assigned to exactly one category.

Failure category	Count
Final-block carrier collision	29,087
Ineffective perturbation	732
Tokenization-equivalent modification	103
Cut off more than 70%	4,401
Total accepted cases	**34,323**

struct paired quality samples. After model warm-up, the accelerator is synchronized immediately before and after every timed generation to avoid measuring asynchronous kernel-launch delay.

For each generated output, we compute the absolute embedding latency and relative overhead as follows:

$$\Delta T = T_{\mathrm{wm}} - T_{\mathrm{base}}, \qquad \mathrm{Overhead} = 100\frac{T_{\mathrm{wm}} - T_{\mathrm{base}}}{T_{\mathrm{base}}}. \tag{26}$$

We report mean latency across prompts. Table 5 reports the mean absolute runtime, relative overhead, and generated-token count so that the cost is comparable across model architectures while accounting for responses that terminate at different lengths.

F Perturbation Failure-Case Analysis

Despite the consistently high TNR reported in the main experiments, a small number of perturbed samples remain accepted. We examine these samples individually and distinguish genuine detector misses from perturbations that do not produce a valid change in the canonical token sequence. We visualize representative examples by highlighting the modified spans, their block locations, and the corresponding detector outcomes. We additionally assign every accepted perturbed case in the 1,000-sample primary evaluation to one of the three mutually exclusive categories summarized in Table 6.

Final-block carrier collision. If a perturbation changes a carrier in the final block but the replacement token produces the same keyed carrier bit, the comparison in that block still passes. Earlier carrier edits alter the committed state and are checked again by subsequent blocks, whereas the final block has no successor to provide this additional evidence, like Figure 4 (left). This is therefore the principal intrinsic source of false acceptance and is consistent with the bound in Appendix A.

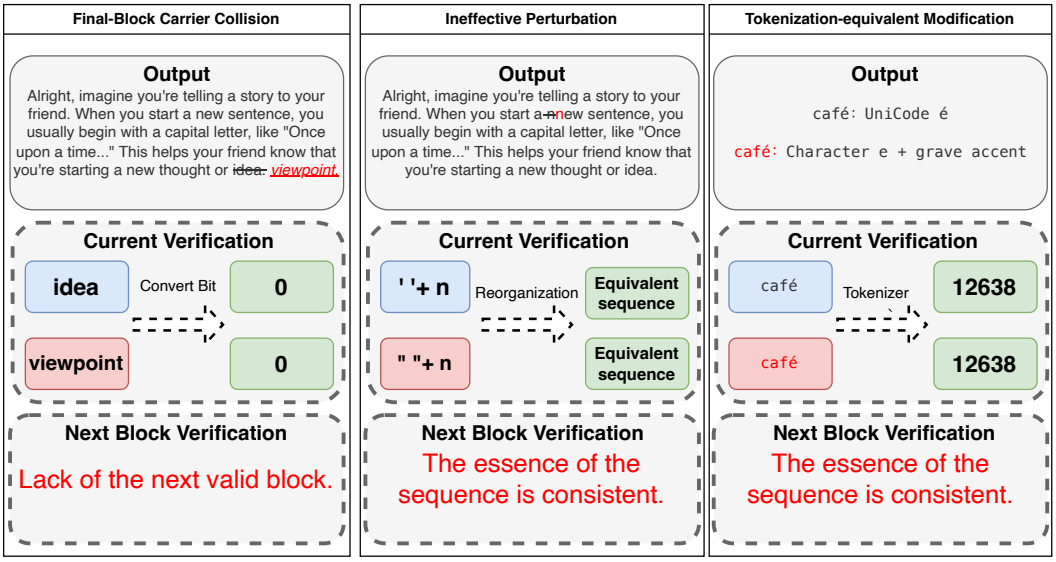

Figure 4: **Failure Cases:** Nearly all apparent failures fell into three categories. The first reflects an intrinsic boundary limitation: because the final block has no subsequent block to validate its committed state, a same-bit carrier substitution in that block may evade detection. The remaining two categories are not genuine token-level detection failures. In one case, the perturbation produces no effective modification; in the other, the surface form changes while canonical tokenization yields the same token sequence. Since Dandelion authenticates canonical token sequences, it correctly accepts both cases.

Ineffective perturbation. Some perturbation operators cannot find an applicable edit in an abnormally short or malformed response, or return text identical to their input, like Figure 4 (mid). Such a sample does not contain an effective modification and should be recorded as a perturbation-construction failure rather than a detector miss.

Tokenization-equivalent modification. A surface-level transformation may change the serialized bytes while leaving the canonical token sequence unchanged. Because Dandelion authenticates the token sequence observed after canonical tokenization, the detector cannot distinguish these representations, like Figure 4 (right). We report such cases separately from edits that change at least one authenticated token, as shown in Table 6.

G Limitations and Intended Scope

Dandelion provides probabilistic tamper evidence for canonical token sequences rather than a deterministic guarantee that every surface-level modification is rejected. Its blockwise chain substantially amplifies changes that occur before the final block, but the final block has no successor through which its committed state can be validated.

Final-block boundary. Under the idealized keyed-function model, modifying the payload of a final block with R_{L-1} reconstructed carriers is accepted with probability at most $2^{-R_{L-1}}$, up to the keyed-function advantage. A substitution that changes only one final-block carrier creates a single fresh comparison and may therefore collide with probability $1/2$. These are worst-case boundary events and remain intrinsic to the current construction.

Aggregate reliability. Semantically consequential edits in the evaluated perturbation setting commonly change a payload token, several canonical tokens, the sequence length, or a block boundary, rather than only one final-block carrier bit. Such edits create multiple fresh comparisons, while an edit in an earlier block is further amplified by the subsequent chain states. Consequently, the ag-

gregate detection rate characterizes reliability against the sampled, key-independent perturbations, rather than cryptographic security against a deliberately chosen worst-case modification.

Contextual dependence. During blockwise generation, each later block is produced conditionally on the preceding watermarked blocks, and carrier substitution selects the most probable admissible token under that context. This dependence limits embedding-induced distortion and makes many isolated, context-preserving changes local in scope. It does not, however, constrain an adversary after generation: a final block may still contain a decisive fact, negation, or conclusion. Dandelion should therefore be treated as a lightweight tamper-detection signal rather than the sole integrity mechanism in adversarial settings. Increasing the carrier budget strengthens final-block payload verification, while stronger terminal authentication remains an important direction for future work.