LoRo-Mark: Provably Lossless And Robust Agent Watermarking

Anonymous authors
Paper under double-blind review

Abstract

As large language model (LLM) agents are increasingly deployed as commercial services, protecting their proprietary orchestration logic and tool-use policies has become an important concern. We consider a realistic infringement scenario, termed **agent repackaging**, in which an adversary integrates a protected agent into its own application through an API and presents it under its own service identity. The adversary may further modify parts of its execution process to obscure the original source. In such cases, the owner typically has access only to the repackaged service interface, making black-box ownership verification essential. Agent watermarking provides a natural way to embed ownership evidence into the agent's behavior for later verification. Under this setting, we argue that an effective watermark should satisfy two key requirements: losslessness, requiring watermarking to preserve the agent's original functionality, and robustness, requiring ownership evidence to remain recoverable after partial modification of the execution process. Existing methods, however, typically embed ownership signals into behavior selection or execution trajectories, intervene in normal agent decisions and provide limited robustness to behavior modification. To address these limitations, we propose **LoRo-Mark**, a provably **lo**ssless and **ro**bust agent water**mark**ing mechanism. For losslessness, LoRo-Mark isolates watermarking into a cryptographically authenticated forensic branch that remains inactive during normal execution and can only be activated by owner-authorized requests. By reducing unauthorized branch activation to standard MAC security, LoRo-Mark provides a formal cryptographic guarantee of performance preservation. For robustness, it redundantly distributes ownership information across forensic behavior sequences, enabling reliable recovery under partial behavior substitution and sequence truncation. Experiments across multiple LLM agents demonstrate zero degradation on normal tasks and reliable ownership verification under sequence modifications.

1 Introduction

Large language model (LLM) agents are increasingly deployed in enterprise automation, software engineering, and other real-world applications (Yang et al., 2024; Zhou et al., 2023; Bran et al., 2023; Wang et al., 2023). Beyond the underlying foundation model, their practical value often lies in system-level capabilities such as task decomposition, tool-use strategies, domain-specific prompting, and the handling of intermediate observations, which together determine how an agent plans and executes complex multi-step tasks (Xi et al., 2025; Yao et al., 2023b; Schick et al., 2023; Park et al., 2023; Liu et al., 2024a; Shinn et al., 2023; Yao et al., 2023a; Wei et al., 2022; Nakano et al., 2021; Qin et al., 2023; Patil et al., 2023; Karpas et al., 2022). Developing such orchestration logic typically requires substantial engineering effort and accumulated operational expertise, making it an increasingly important form of intellectual property. Protecting this system-level IP has therefore become a growing concern as agent-based services move toward large-scale deployment (Wang et al., 2026).

Among the potential threats to agent IP, we focus on a realistic yet under-explored infringement scenario, which we term agent repackaging. As illustrated in Figure 1, an adversary may integrate a protected agent into its own application through an API and present it under its own service identity. To obscure the original source, the adversary can alter the host application's front-end

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.

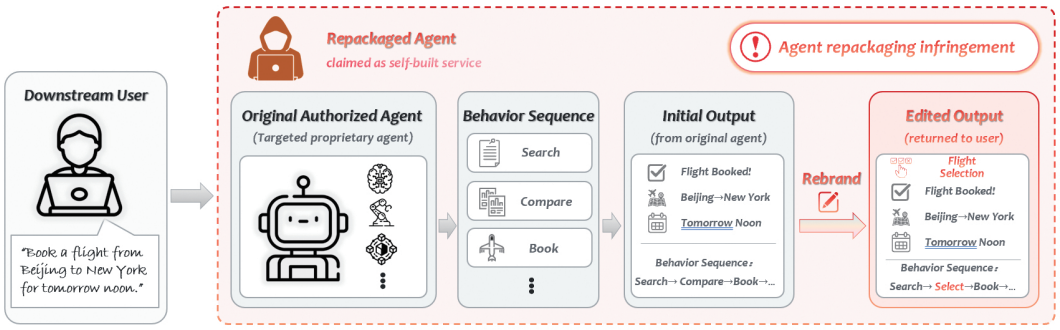

Figure 1: Scenario of repackaging infringement: A downstream user sends a request to the repackaged agent to book a specific flight. The repackaged agent completes the booking task via a sequence of tool calls and returns an edited output modified from the initial output to obscure the original source.

presentation, including the displayed agent thinking process, behavior sequence, action trajectory, and observation results. Such repackaging allows the adversary to exploit the capabilities of a non-self-developed agent while presenting them as its own. More importantly, the original owner typically has no access to the internal implementation of the suspicious service and can only interact with its exposed interface, making black-box ownership verification particularly important. Agent watermarking therefore provides a natural means of embedding ownership evidence into agent behavior for subsequent verification (Uchida et al., 2017; Adi et al., 2018; Rouhani et al., 2018; Chen et al., 2018; 2019; Kirchenbauer et al., 2023; Christ et al., 2024; Kuditipudi et al., 2024; Dathathri et al., 2024; Huang et al., 2025; 2026; Wang et al., 2026).

Under agent repackaging, an effective watermark should satisfy two key requirements: losslessness and robustness (Christ et al., 2024; Kuditipudi et al., 2024). First, watermarking is introduced for ownership protection and should therefore preserve the agent's original functionality during normal use. Second, a repackaging adversary may partially modify the agent's execution process to obscure its source, requiring ownership evidence to remain recoverable after such modifications. Existing agent watermarks, however, typically encode ownership signals through specific behavior selections or execution trajectories (Huang et al., 2025; 2026; Wang et al., 2026; An et al., 2026). Such designs intervene in the agent's decision process, making strict functionality preservation difficult, while the resulting ownership evidence remains vulnerable to downstream behavior modification.

These limitations motivate us to rethink how watermarking should interact with agent execution. For provable losslessness, the key is to maintain an untouched normal execution path while making unauthorized activation of the watermark branch cryptographically negligible. Rather than minimizing watermark-induced perturbations, we preserve the original execution path for ordinary requests and activate watermarking only through a MAC-authenticated forensic branch (Bellare et al., 1996; Goldwasser et al., 2022; National Institute of Standards and Technology, 2008). For robustness, we redundantly distribute ownership information across the forensic behavior sequence, preventing the evidence from depending on any single behavior and enabling its recovery after partial behavior modification or truncation (Lin & Costello, 2004).

Building on these principles, we propose **LoRo-Mark**, a provably **lo**ssless and **ro**bust agent wa-ter**mark**ing mechanism for black-box ownership verification. LoRo-Mark uses cryptographic authentication to control access to the forensic branch and redundantly embeds copyright information across the resulting behavior sequence (Bellare et al., 1996; National Institute of Standards and Technology, 2008). During verification, the owner uses the secret key to reconstruct the pseudorandom behavior mappings, translate the observed behaviors back into watermark bits, and recover the copyright message through redundancy-aware decoding (Goldreich et al., 1986; Lin & Costello, 2004). We formally reduce unauthorized branch activation to MAC-EUF-CMA security, establishing provable performance preservation for ordinary requests, and derive recovery guarantees under behavior substitution and sequence truncation.

Our main contributions are summarized as follows:

- We focus on **agent repackaging**, a realistic yet under-explored infringement scenario in which a protected agent can be proxied, redeployed, and partially modified behind a black-box service. We identify **losslessness** and **robustness** as two key requirements for agent watermarking under this setting: preserving the agent's original functionality while retaining ownership evidence after repackaging.
- We propose **LoRo-Mark**, a provably lossless and robust agent watermarking mechanism. LoRo-Mark maintains an untouched normal execution path through a MAC-authenticated forensic branch, making unauthorized activation cryptographically negligible, while redundantly encoding ownership information across forensic behavior sequences to enable reliable recovery under partial behavior substitution and sequence truncation.
- We provide rigorous theoretical and empirical validation of both properties. Extensive experiments across diverse agents, models, and tasks demonstrate zero degradation on normal execution and reliable ownership verification under substantial sequence modifications.

2 Related Work

2.1 Agent Watermarking

Existing agent watermarking approaches embed ownership signals across three principal loci—planning behaviors, execution trajectories, and memory-state evolutions. At the behavior layer, *Agent-Guide* (Huang et al., 2025) targets high-level planning decisions rather than surface tokens (Kirchenbauer et al., 2023; Christ et al., 2024; Kuditipudi et al., 2024; Dathathri et al., 2024). It biases the behavior distribution over a predefined set $\mathcal{B}$ via a pseudorandom green-list (Kirchenbauer et al., 2023) and uses a z-statistic aggregated over multiple rounds for single-bit presence detection. *AgentMark* (Huang et al., 2026) instead encodes multi-bit identifiers through *distribution-preserving conditional sampling*, which preserves the marginal behavior distribution while remaining compatible with black-box agent APIs. At the trajectory layer, *SeqWM* (An et al., 2026) conditions each embedding step on history-dependent state transitions; *ACTHOOK* (Meng et al., 2026) injects auxiliary decoy actions activated by predefined trigger keys; and *AGENTWM* (Wang et al., 2026) discovers pools of *semantically equivalent tool-execution paths* and biases the agent toward user-specific subsets, with verification based on Jensen–Shannon divergence. At the memory layer, *MEMMARK* (Zhang et al., 2026) embeds signals into latent memory-state transitions, enabling provenance attribution from the final memory snapshot even when intermediate trajectories are unavailable.

2.2 Cryptographically Undetectable Backdoors

Shafi Goldwasser and his coworkers demonstrated that a malicious trainer can plant undetectable backdoors in classifiers: no computationally bounded observer, whether black-box or white-box, can tell backdoored models apart from clean ones without the secret key (Goldwasser et al., 2022). Their design runs a digital-signature verification circuit parallel to the base model, activating the backdoor solely for valid-signature inputs while leaving others intact, with white-box undetectability grounded in Sparse PCA hardness. The core insight is that a secret key provides cryptographically secure control over model behavior — key holders can reliably trigger target outputs, whereas observers without keys detect nothing. We draw inspiration from this key-driven behavioral asymmetry. Nevertheless, agent scenarios bring unique hurdles including multi-step reasoning, tool calls, and emergent plan-level semantics, so we cannot directly reuse classifier-level constructions and must build an agent-tailored watermarking protocol instead.

3 Methodology

3.1 Threat Model

Adversary Capabilities. We model a protected agent as $A = (M, \Pi, \mathcal{T})$, where M denotes the underlying LLM, Π the orchestration policy, and $\mathcal{T}$ the available tool set. At step t, the agent selects a high-level behavior $b_t \in \mathcal{B}_t$ from a candidate behavior set $\mathcal{B}_t$ and instantiates it into a concrete action a_t. A complete execution produces a behavior sequence $\tau_b = (b_1, \ldots, b_n)$ and the corresponding action trace $\tau_a = (a_1, \ldots, a_n)$ (Huang et al., 2025; 2026; Wang et al., 2026). After obtaining

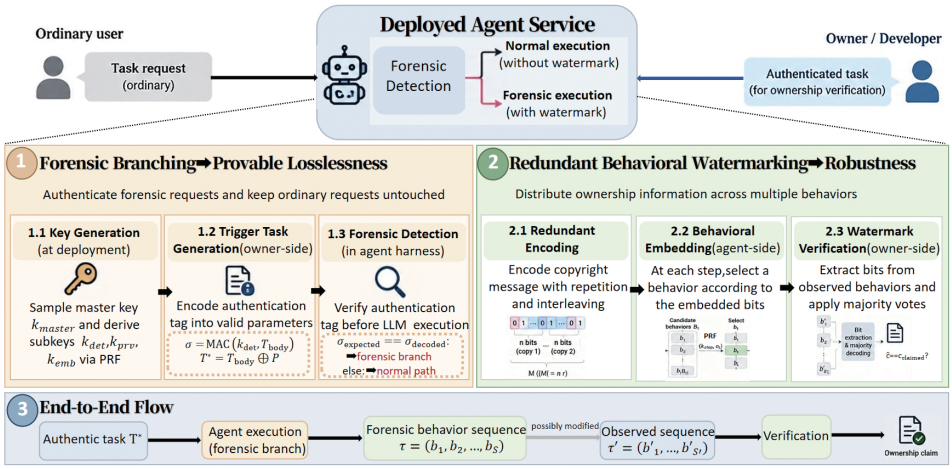

Figure 2: An overview of LoRo-Mark.

access to the protected agent, the adversary may repackage it into a new service $A' = (M, \Pi, \mathcal{T})$, while preserving the underlying LLM, orchestration policy, available tools, and actual execution process. Instead, the adversary may modify the host application's front-end presentation, including the displayed agent thinking process, behavior sequence, action trajectory, and observation results, before exposing them to downstream users. We therefore view agent repackaging as a service-level transformation

$$\mathcal{R} : A \to A',$$

where $\mathcal{R}$ leaves the execution flow, decision logic, tool set, and actual trajectory unchanged, but modifies their exposed presentation while still preserving the core utility and advertised functionality of the original agent. The adversary may know the watermarking mechanism and interact with the protected agent through public interfaces, but does not possess the owner's secret keys.

Defender Goal. Given only black-box access to the repackaged agent A', the defender aims to determine whether A' still contains or invokes the protected agent A (Huang et al., 2025; 2026; Wang et al., 2026). The defender constructs an owner-authorized forensic request x^* using the secret key k and submits it to A'. The repackaged service returns an observable execution trace $\tau' = (b'_1, a'_1, \ldots, b'_m, a'_m)$, which may already have been modified by the repackaging transformation $\mathcal{R}$. The defender then applies a keyed extractor $\hat{c} = \mathsf{Ext}_k(\tau')$ to recover the ownership message and accepts the ownership claim iff $\mathsf{Ver}_k(\hat{c}, c) = 1$, where c is the claimed watermark payload. Accordingly, the verification goal is successful black-box watermark recovery from the repackaged execution trace, even when τ' has undergone partial modification or truncation.

3.2 LoRo-Mark

Figure 2 presents an overview of LoRo-Mark. LoRo-Mark is designed around the two requirements identified above: *provable losslessness* and *robustness*. Accordingly, its design consists of two key components. First, *forensic branching* preserves an untouched execution path for ordinary requests while making unauthorized activation of the watermark branch cryptographically negligible, thereby enabling provable losslessness. Second, *redundant behavioral watermarking* (Lin & Costello, 2004) distributes ownership information across the forensic behavior sequence, allowing the watermark to remain recoverable under partial sequence modification.

3.2.1 Forensic Branching

The key to provable losslessness is not merely to reduce watermark-induced perturbations, but to ensure that ordinary requests remain on an execution path untouched by watermarking, except with cryptographically negligible probability. To achieve this, we place all watermark functionality behind an authenticated forensic branch. An ordinary request follows the original agent execution path without watermark intervention, whereas only a request carrying a valid owner-generated

authentication tag can activate the forensic branch. We instantiate this design through three modules: **Key Generation**, **Trigger Task Generation**, and **Forensic Detection**.

Key Generation. At deployment time, given a security parameter λ, the developer samples a master key k_{master} and derives three subkeys $\{k_{\text{det}}, k_{\text{prv}}, k_{\text{emb}}\}$ through domain-separated PRF evaluations (Goldreich et al., 1986). The detection key k_{det} authenticates forensic requests, k_{emb} determines how authentication information is encoded into valid task parameters, and k_{prv} is reserved for keyed behavioral watermarking in the forensic branch. The owner retains these keys for forensic verification, while the deployed agent stores the corresponding keys required for request detection and watermark generation.

Trigger Task Generation. To initiate ownership verification, the owner constructs an authenticated forensic request from an otherwise valid task T_{body}. Let P denote a set of semantically valid task parameters, such as budget, date, quantity, preference, or priority. We first compute (Bellare et al., 1996)

$$\sigma = \text{MAC}(k_{\text{det}}, T_{\text{body}}),$$

and then use k_{emb} to encode σ into the parameter values P. The resulting forensic request is

$$T^* = T_{\text{body}} \oplus P,$$

where $\oplus$ denotes assembling the task body with the encoded parameters. Because P occupies legitimate parameter slots and remains within their valid domains, T^* retains the syntax and semantics of a valid agent request. When the available parameter capacity is smaller than the full MAC output, the authentication tag can be truncated to ℓ bits; the resulting security–capacity trade-off is evaluated in Section 5.

Forensic Detection. Forensic detection (denoted as $\mathcal{F}$) is implemented in the agent harness and is performed before LLM execution. Given an incoming request $T = (T_{\text{body}}, P)$, it computes

$$\sigma_{\text{expected}} = \text{MAC}(k_{\text{det}}, T_{\text{body}}), \tag{1}$$

$$\sigma_{\text{decoded}} = \mathcal{D}(P, k_{\text{emb}}). \tag{2}$$

where $\mathcal{D}$ indicates the operation that decodes σ_{decoded} from the parameters P. If $\sigma_{\text{expected}} = \sigma_{\text{decoded}}$, the request is routed to the forensic branch; otherwise, it proceeds along the original execution path. Hence, activating the forensic branch without the owner's keys requires producing a valid authentication tag. For any unauthorized request T,

$$\Pr[\mathcal{F}(T) = 1] \leq \varepsilon_{\text{MAC}}(\lambda), \tag{3}$$

where $\varepsilon_{\text{MAC}}(\lambda)$ is negligible under standard MAC security (Bellare et al., 1996). Consequently, ordinary requests remain on the original execution path except with cryptographically negligible probability, providing the basis for LoRo-Mark's provable losslessness.

3.2.2 Redundant Behavioral Watermarking

LoRo-Mark embeds the ownership information into the resulting behavior sequence. The key design principle for robustness is to avoid relying on any individual behavior as ownership evidence. Instead, we redundantly encode the copyright message and distribute the encoded bits across multiple forensic behaviors. Consequently, even if part of the behavior sequence is substituted or truncated during repackaging, the surviving behaviors can still provide sufficient evidence for watermark recovery. This process consists of three stages: **Redundant Encoding**, **Behavioral Embedding**, and **Watermark Verification**.

Redundant Encoding. Let c denote the copyright message and $n = 8|c|$ its bit length, with $c = (c_0, c_1, \ldots, c_{n-1})$. To distribute each copyright bit across different positions of the forensic sequence, we encode c with repetition redundancy r (Lin & Costello, 2004). Rather than placing the r copies of each bit consecutively, we interleave multiple copies of the entire copyright message:

$$M = (c_0 c_1 \cdots c_{n-1}) \,\|\, \cdots \,\|\, (c_0 c_1 \cdots c_{n-1}), \tag{4}$$

where the message is repeated r times and $|M| = nr$. Hence, the k-th copy of copyright bit c_i appears at position $i + kn$ in M. This interleaved layout spreads different copies of the same copyright bit across distant positions of the behavior sequence, allowing them to survive partial sequence corruption and truncation.

Behavioral Embedding. LoRo-Mark next maps the redundantly encoded bitstream M into a sequence of forensic behaviors. Given the authentication tag σ generated for the forensic request, we derive a request-specific embedding key

$$k_{\text{step}} = \text{MAC}(k_{\text{prv}}, \sigma). \tag{5}$$

At execution step t, the agent observes o_t and constructs an admissible candidate behavior set $\mathcal{B}_t$. Using k_{step} and o_t, LoRo-Mark generates a pseudorandom permutation $\pi_t = \text{PRF}(k_{\text{step}}, o_t, \mathcal{B}_t)$ (Goldreich et al., 1986) over the candidate behaviors. The number of bits that can be embedded at this step is $q_t = \lfloor \log_2 |\mathcal{B}_t| \rfloor$. LoRo-Mark reads the next q_t bits from M, interprets them as an integer $j_t \in [0, 2^{q_t} - 1]$, and selects $b_t = \pi_t[j_t]$. Repeating this process across execution steps produces the forensic behavior sequence

$$\tau = (b_1, b_2, \ldots, b_S),$$

through which the redundant copyright bitstream is distributed across multiple observable behaviors.

Watermark Verification. Given a behavior sequence $\tau' = (b'_1, \ldots, b'_{S'})$ observed from a suspicious repackaged service, the owner reverses the embedding process using the secret key. For each surviving behavior b'_t, the verifier reconstructs the same candidate set and pseudorandom ordering π_t, identifies the index of b'_t in π_t, and converts that index back into the corresponding embedded bits. Aggregating the extracted bits yields a recovered redundant bitstream $\widehat{M}$.

The verifier then groups the surviving copies corresponding to each copyright bit and performs majority decoding (Lin & Costello, 2004) to obtain

$$\hat{c} = (\hat{c}_0, \ldots, \hat{c}_{n-1}).$$

The ownership claim is accepted iff $\hat{c} = c_{\text{claimed}}$.

Because each copyright bit is represented at multiple positions, as long as sufficient copies survive for decoding, the ownership message remains recoverable. We formalize these robustness guarantees under behavior substitution and sequence truncation in Section B.3.

4 Provable Losslessness

The losslessness of LoRo-Mark follows from a simple observation: the watermarked agent differs from the original agent only when the forensic branch is activated. Therefore, it suffices to show that an unauthorized request activates this branch only with cryptographically negligible probability. We first bound such unauthorized activation through the security of the underlying MAC (Bellare et al., 1996; National Institute of Standards and Technology, 2008), and then show that this bound directly implies losslessness.

Definition 1 ($(\varepsilon, t, q, \lambda)$-losslessness)**.** *Let* API_{WM} *denote the LoRo-Mark-protected agent and* $\text{API}_{\text{plain}}$ *the original agent. LoRo-Mark is* $(\varepsilon, t, q, \lambda)$-lossless *if, for every PPT distinguisher* $\mathcal{A}$ *running in time at most* t *and making at most* q *API queries without access to the owner's secret keys,*

$$\text{Adv}_{\text{loss}}(\mathcal{A}) := \left|\Pr[\mathcal{A}^{\text{API}_{\text{WM}}} \to 1] - \Pr[\mathcal{A}^{\text{API}_{\text{plain}}} \to 1]\right| \le \varepsilon(\lambda). \tag{6}$$

Theorem 1 (Provable Losslessness of LoRo-Mark)**.** *Suppose the MAC scheme used by LoRo-Mark is* $(\varepsilon_{\text{MAC}}, t, q, \lambda)$*-EUF-CMA secure. Then LoRo-Mark is* $(\varepsilon_{\text{MAC}}, t, q, \lambda)$*-lossless.*

Proof. We prove the result in two steps.

Step 1: Unauthorized forensic activation is negligible. Consider any request $T = (T_{\text{body}}, P)$ that is not generated by the authorized Trigger Task Generation process. Forensic Detection accepts T only if

$$\mathcal{D}(P, k_{\text{emb}}) = \text{MAC}(k_{\text{det}}, T_{\text{body}}). \tag{7}$$

Hence, an adversary that successfully constructs an unauthorized request accepted by Forensic Detection also produces a valid authentication tag for T_{body}.

Suppose an adversary $\mathcal{A}$ can cause such an activation with probability p. We construct a MAC forger $\mathcal{B}$ that samples k_{emb} independently and runs $\mathcal{A}$. Whenever $\mathcal{A}$ outputs an accepted request $T = (T_{\text{body}}, P)$, $\mathcal{B}$ outputs

$$\big(T_{\text{body}}, \mathcal{D}(P, k_{\text{emb}})\big) \tag{8}$$

as a MAC forgery. Therefore, by EUF-CMA security,

$$\Pr[\mathsf{Bad}] := \Pr[\exists T : \mathcal{F}(T) = 1 \text{ for an unauthorized } T] \le \varepsilon_{\mathrm{MAC}}(\lambda). \tag{9}$$

Step 2: No forensic activation implies identical normal execution. Conditioned on $\neg\mathsf{Bad}$, every query issued by $\mathcal{A}$ follows the original execution path of the agent. LoRo-Mark does not modify the underlying model, orchestration policy, tool invocation, or output generation on this path. Consequently, $\mathrm{API}_{\mathrm{WM}}$ and $\mathrm{API}_{\mathrm{plain}}$ induce identical output distributions conditioned on $\neg\mathsf{Bad}$.

The two APIs can therefore differ only if Bad occurs, yielding

$$\mathrm{Adv}_{\mathrm{loss}}(\mathcal{A}) \le \Pr[\mathsf{Bad}] \tag{10}$$

$$\le \varepsilon_{\mathrm{MAC}}(\lambda). \tag{11}$$

Thus, LoRo-Mark is $(\varepsilon_{\mathrm{MAC}}, t, q, \lambda)$-lossless. □

5 Experiments

We evaluate LoRo-Mark around the two central properties established in our design: *losslessness* and *robustness*. For losslessness, we examine whether forensic branching introduces any unintended activation or performance degradation during normal agent execution. For robustness, we evaluate whether the embedded ownership information remains recoverable when the forensic behavior sequence is partially modified during agent repackaging.

5.1 Experimental Setup

Dataset. We conduct our experiments on xLAM-function-calling-60k (Liu et al., 2024b), which contains 60,000 function-calling trajectories covering 3,673 tools across 21 domains. We use the natural-language queries as task bodies T_{body} and their associated tool specifications to construct agent execution tasks. For utility evaluation, we select three representative domains, *Data*, *Business*, and *Social*, and sample 100 tasks from each domain.

Agent Setup. Unless otherwise specified, we use Qwen3-4B (Yang et al., 2025) as the default LLM backbone and ReAct (Yao et al., 2023b) as the agent framework, with tools executed through a function-calling interface. To evaluate whether LoRo-Mark depends on a particular backbone, we additionally conduct experiments with Qwen3-8B (Yang et al., 2025) and Ministral-8B-Instruct-2410 (Mistral AI, 2024). The experimental results remain highly consistent across all models because both Forensic Detection and Behavioral Embedding operate within the agent harness before the underlying LLM is invoked, making LoRo-Mark independent of the backbone model. Due to space constraints, the following section focuses on agents built on a single LLM backbone as a representative example. For agent utility evaluation, we use DeepSeek-V3 (DeepSeek-AI et al., 2024) as the judge model to assess both final responses and execution trajectories.

Watermark Configuration. LoRo-Mark uses HMAC-SHA256 as the authentication primitive (Bellare et al., 1996; National Institute of Standards and Technology, 2008). Forensic requests are generated by computing an authentication tag over T_{body} and encoding the tag into semantically valid parameter slots, while normal tasks are left unchanged. Unless otherwise specified, we use the copyright message corresponding to 104 bits, and an interleaved repetition redundancy of $r = 3$.

Evaluation Metrics. We evaluate LoRo-Mark from two primary perspectives. For watermark verification, we report the *true-positive rate* (TPR) on authorized forensic requests and the *false-positive rate* (FPR) on normal or unauthorized requests. For agent utility, we report Pass Rate (PR), Response Score (RS), Trajectory Score (TS), and Success Rate (SR) (Huang et al., 2026; Wang et al., 2026), computed over N evaluation tasks as

$$\mathrm{PR} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\left[\text{response}_i \text{ fulfills the request}\right],$$

$$\mathrm{RS} = \frac{1}{N}\sum_{i=1}^{N} \frac{r_i - 1}{4}, \qquad \mathrm{TS} = \frac{1}{N}\sum_{i=1}^{N} \frac{t_i - 1}{4}, \qquad \mathrm{SR} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\left[\text{task}_i \text{ is successfully completed}\right],$$

Table 1: Utility comparison of different agent watermarking schemes.

Method	Data				Business				Social			
	PR ↑	RS ↑	TS ↑	SR ↑	PR ↑	RS ↑	TS ↑	SR ↑	PR ↑	RS ↑	TS ↑	SR ↑
Unwatermarked	0.6200	0.6175	0.8850	0.9900	0.6000	0.6025	0.8825	0.9900	0.7100	0.6375	0.8825	0.9800
AGENTWM	0.5500	0.5700	0.8375	0.8800	0.5100	0.5400	0.8075	0.8500	0.6300	0.5875	0.8150	0.8900
Per. Drop (Δ)	−11.29%	−7.69%	−5.37%	−11.11%	−15.00%	−10.37%	−8.50%	−14.14%	−11.27%	−7.84%	−7.65%	−9.18%
AgentMark	0.5600	0.5975	0.8675	0.9600	0.5000	0.5550	0.8300	0.9300	0.5900	0.5900	0.8250	0.9200
Per. Drop (Δ)	−9.68%	−3.24%	−3.03%	−3.03%	−16.67%	−7.88%	−5.95%	−6.06%	−16.90%	−7.45%	−6.52%	−6.12%
LoRo-Mark	**0.6200**	**0.6175**	**0.8850**	**0.9900**	**0.6000**	**0.6025**	**0.8825**	**0.9900**	**0.7100**	**0.6375**	**0.8825**	**0.9800**
Per. Drop (Δ)	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**

where $r_i, t_i \in \{1, \ldots, 5\}$ are the response-quality and trajectory-quality scores, respectively, and $\mathbb{I}[\cdot]$ is the indicator function. Higher values indicate better task performance. For robustness, we report TPR, computed using the same definition above, after applying different degrees of behavior substitution or sequence truncation to the forensic behavior sequence.

5.2 Losslessness

We first evaluate whether LoRo-Mark preserves normal agent execution in practice. According to our theoretical analysis, losslessness relies on two conditions: ordinary requests should not activate the forensic branch, and, when the branch is not activated, the agent should follow the original execution path without watermark intervention. We therefore evaluate both false activation and task utility.

False Activation. We first measure whether normal requests accidentally activate the forensic branch. We randomly sample 10,000 normal tasks from xLAM-function-calling-60k (Liu et al., 2024b) and pass them through Forensic Detection. None of the requests is accepted as a forensic task, yielding FPR = $\frac{0}{10{,}000} = 0.00\%$.

This empirical result is consistent with our theoretical analysis that unauthorized forensic activation is bounded by the security of the underlying authentication mechanism.

Utility Preservation. We next examine whether introducing LoRo-Mark changes the agent's performance on ordinary tasks. We compare LoRo-Mark with the unwatermarked agent and two existing agent watermarking methods, AGENTWM (Wang et al., 2026) and AgentMark (Huang et al., 2026), on the *Data*, *Business*, and *Social* domains. For each domain, we evaluate 100 tasks and report Pass Rate (PR), Response Score (RS), Trajectory Score (TS), and Success Rate (SR), with DeepSeek-V3 (DeepSeek-AI et al., 2024) serving as the judge. As shown in Table 1, existing watermarking methods introduce different degrees of performance degradation because watermark embedding intervenes in normal behavior selection or execution. In contrast, LoRo-Mark achieves exactly the same scores as the unwatermarked agent across all three domains and all four metrics, resulting in $\Delta\text{PR} = \Delta\text{RS} = \Delta\text{TS} = \Delta\text{SR} = 0$. These results empirically confirm that ordinary requests remain on the untouched execution path of the original agent, consistent with the losslessness guarantee established in Section 4.

5.3 Robustness

We evaluate the robustness of LoRo-Mark against two representative repackaging operations: partial behavior substitution and sequence truncation.

Behavior Substitution. We randomly replace a fraction ρ of behaviors in the forensic sequence and evaluate whether the copyright message can still be recovered. As shown in Figure 3(a), LoRo-Mark consistently achieves higher TPR than AGENTWM (Wang et al., 2026) and AgentMark (Huang et al., 2026) under increasing substitution ratios. With $r = 3$, LoRo-Mark remains reliable under moderate behavior modification.

Sequence Truncation. We further truncate the forensic sequence and retain only a prefix of the original execution trace. As shown in Figure 3(b), with $r = 3$, LoRo-Mark achieves 100% TPR once approximately 34.6% of the original steps are retained, while shorter prefixes are insufficient for complete recovery. These results confirm that redundant behavioral watermarking preserves ownership evidence under partial modification and truncation of the execution sequence.

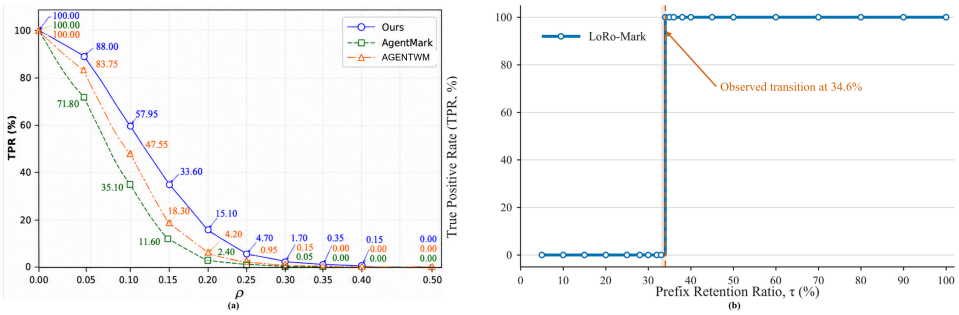

Figure 3: (a) Comparison of TPR for different watermarking schemes under behavior substitution with varying ratios ρ. (b) TPR for LoRo-Mark under different prefix retention ratios.

Table 2: True-positive rates under different behavior-substitution ratios (ρ) and repetition factors (r).

r	Steps	TPR (%)								
		$\rho = 0.05$	$\rho = 0.10$	$\rho = 0.15$	$\rho = 0.20$	$\rho = 0.25$	$\rho = 0.30$	$\rho = 0.35$	$\rho = 0.40$	$\rho = 0.50$
3	**52**	100.00	88.00	57.95	33.60	15.10	4.70	1.70	0.15	0.00
5	**87**	100.00	98.65	91.10	73.95	52.20	27.45	12.30	1.30	0.00
7	**122**	100.00	100.00	98.70	91.25	77.65	59.00	37.00	8.05	0.60

5.4 Ablation and Parameter Analysis

Effect of Redundancy. We first study the effect of the repetition factor r, which directly controls the trade-off between robustness and forensic trace length. For a copyright message of $|c|$ bytes and a candidate behavior set of size $|\mathcal{B}|$, LoRo-Mark embeds $\lfloor \log_2 |\mathcal{B}| \rfloor$ bits per step. Accordingly, the number of steps required to encode the redundant copyright message is $S = \left\lceil \frac{8|c|r}{\lfloor \log_2 |\mathcal{B}| \rfloor} \right\rceil$. Thus, increasing r introduces additional forensic steps but provides more redundant copies for watermark recovery. To quantify this trade-off, we fix $|\mathcal{B}| = 64$ and $|c| = 13$ bytes, and vary $r \in \{3, 5, 7\}$. As shown in Table 2, larger r consistently improves TPR under the same behavior-substitution ratio, confirming that additional redundancy strengthens resistance to sequence corruption (Lin & Costello, 2004). Meanwhile, the required forensic trace length grows linearly with r. We therefore use $r = 3$ as the default setting, which provides a practical balance between robustness and verification overhead.

Effect of Parameter Capacity. We evaluate how parameter capacity and MAC length jointly affect false activation. Across 12 configurations with capacity_bits $\in \{14, 64, 96\}$ and mac_bits $\in \{32, 64, 96, 128\}$, the empirical FPR consistently follows the theoretical bound $2^{-\min(\text{capacity_bits}, \text{mac_bits})}$. For the 14-bit capacity setting, we observe an FPR of 5.0×10^{-5}, close to the theoretical 2^{-14}, while all configurations with at least 32 effective bits produce zero false activations in 10,000 trials. These results confirm that the effective security is determined by the smaller of parameter capacity and MAC length.

6 Conclusion

We study black-box ownership verification under agent repackaging and propose **LoRo-Mark**, a provably lossless and robust agent watermarking mechanism. LoRo-Mark isolates watermarking from normal execution through an authenticated forensic branch and redundantly distributes ownership information across forensic behavior sequences. We formally establish performance preservation under standard MAC security and empirically demonstrate unchanged utility on normal tasks together with robust watermark recovery under behavior substitution and sequence truncation. Our results suggest a simple design principle for agent watermarking: ownership verification can be separated from normal execution by design.

AI USE STATEMENT

In this work, we used generative AI tools to assist with literature research on the agent-repackaging threat scenario and related work. We also used generative AI tools to help refine the experimental design for evaluating losslessness, robustness, and ablation settings; organize relevant benchmarks; and structure and summarize experimental data. After completing the manuscript, we used generative AI tools for language polishing, including correcting grammatical and typographical errors, checking presentation details, and improving clarity and fluency. All AI-assisted research, experimental designs, data summaries, citations, and textual revisions were independently reviewed and verified by the authors. We take full responsibility for the final content of this work, including all text, claims, experimental results, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the information and materials necessary to reproduce our theoretical and empirical results. The main paper describes the threat model, watermark construction, experimental settings, evaluation metrics, baselines, benchmarks, and parameter configurations. The appendix provides detailed algorithms, assumptions, and complete proofs of the losslessness, unforgeability, and robustness results. The supplementary materials include the implementation code, environment dependencies, configuration files, and instructions for reproducing the losslessness, robustness, and ablation experiments. The experimental settings and reported parameters are documented in the accompanying README to facilitate consistent reproduction of the results.

REFERENCES

Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX Security Symposium*, pp. 1615–1631, 2018.

Hyeseon An, Shinwoo Park, Dongsu Kim, and Yo-Sub Han. Sequential behavioral watermarking for LLM agents. *arXiv preprint arXiv:2605.11036*, 2026. URL https://arxiv.org/abs/2605.11036.

Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In *Advances in Cryptology—CRYPTO '96*, pp. 1–15, 1996. doi: 10.1007/3-540-68697-5_1.

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D. White, and Philippe Schwaller. ChemCrow: Augmenting large-language models with chemistry tools. *arXiv preprint arXiv:2304.05376*, 2023. URL https://arxiv.org/abs/2304.05376.

Huili Chen, Bita Darvish Rouhani, and Farinaz Koushanfar. DeepMarks: A digital fingerprinting framework for deep neural networks. *arXiv preprint arXiv:1804.03648*, 2018. URL https://arxiv.org/abs/1804.03648.

Huili Chen, Bita Darvish Rouhani, and Farinaz Koushanfar. BlackMarks: Blackbox multi-bit watermarking for deep neural networks. *arXiv preprint arXiv:1904.00344*, 2019. URL https://arxiv.org/abs/1904.00344.

Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models. *arXiv preprint arXiv:2306.09194*, 2024.

Sumanth Dathathri, Abigail See, Sumit Ghaisas, et al. Scalable watermarking for identifying large language model outputs. *Nature*, 634:818–823, 2024. doi: 10.1038/s41586-024-08025-4.

DeepSeek-AI et al. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024. URL https://arxiv.org/abs/2412.19437.

Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986. doi: 10.1145/6490.6503.

Shafi Goldwasser, Michael P. Kim, Vinod Vaikuntanathan, and Or Zamir. Planting undetectable backdoors in machine learning models. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 931–942, 2022. doi: 10.1109/FOCS54457.2022.00092.

Kaibo Huang, Zipei Zhang, Zhongliang Yang, and Linna Zhou. Agent guide: A simple agent behavioral watermarking framework. *arXiv preprint arXiv:2504.05871*, 2025.

Kaibo Huang, Jin Tan, Yukun Wei, Wanling Li, Zipei Zhang, Hui Tian, Zhongliang Yang, and Linna Zhou. AgentMark: Utility-preserving behavioral watermarking for agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12581–12603, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.573. URL https://aclanthology.org/2026.acl-long.573/.

Ehud Karpas, Omri Abend, Yonatan Belinkov, Barak Lenz, Opher Lieber, Nir Ratner, Yoav Shoham, Hofit Bata, Yoav Levine, Kevin Leyton-Brown, Dor Muhlgay, Noam Rozen, Erez Schwartz, Gal Shachaf, Shai Shalev-Shwartz, Amnon Shashua, and Moshe Tenenholtz. MRKL systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. *arXiv preprint arXiv:2205.00445*, 2022. URL https://arxiv.org/abs/2205.00445.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 17061–17084, 2023.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research*, 2024.

Shu Lin and Daniel J. Costello. *Error Control Coding: Fundamentals and Applications*. Pearson Prentice Hall, Upper Saddle River, NJ, 2 edition, 2004.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, 2024a. URL https://arxiv.org/abs/2308.03688.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Shirley Kokane, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, et al. APIGen: Automated pipeline for generating verifiable and diverse function-calling datasets. In *Advances in Neural Information Processing Systems*, volume 37, pp. 54463–54482, 2024b. doi: 10.52202/079017-1725. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/61cce86d180b1184949e58939c4f983d-Abstract-Datasets_and_Benchmarks_Track.html.

Wenlong Meng, Chen Gong, Terry Yue Zhuo, Fan Zhang, Kecen Li, Zheng Liu, Zhou Yang, Chengkun Wei, and Wenzhi Chen. Watermarking LLM agent trajectories. *arXiv preprint arXiv:2602.18700*, 2026. URL https://arxiv.org/abs/2602.18700.

Mistral AI. Ministral 8B Instruct 2410. Hugging Face model card, 2024. URL https://huggingface.co/mistralai/Ministral-8B-Instruct-2410. Accessed 2026-09-16.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021. URL https://arxiv.org/abs/2112.09332.

National Institute of Standards and Technology. The keyed-hash message authentication code (HMAC). Technical Report Federal Information Processing Standards Publication 198-1, National Institute of Standards and Technology, 2008.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023. doi: 10.1145/3586183.3606763.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. *arXiv preprint arXiv:2305.15334*, 2023. URL https://arxiv.org/abs/2305.15334.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. *arXiv preprint arXiv:2307.16789*, 2023. URL https://arxiv.org/abs/2307.16789.

Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. DeepSigns: A generic watermarking framework for IP protection of deep learning models. *arXiv preprint arXiv:1804.00750*, 2018. URL https://arxiv.org/abs/1804.00750.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023. URL https://arxiv.org/abs/2303.11366.

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

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023. URL https://arxiv.org/abs/2305.16291.

Liwen Wang, Zongjie Li, Yuchong Xie, Shuai Wang, Dongdong She, Wei Wang, and Juergen Rahmel. On protecting agentic systems' intellectual property via watermarking. *arXiv preprint arXiv:2602.08401*, 2026.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2201.11903.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101, 2025. doi: 10.1007/s11432-024-4222-0. URL https://doi.org/10.1007/s11432-024-4222-0.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. *arXiv preprint arXiv:2405.15793*, 2024. URL https://arxiv.org/abs/2405.15793.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023a. URL https://arxiv.org/abs/2305.10601.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023b.

Haobo Zhang, Xutao Mao, Guangyuan Dong, Ziwei Li, Xuanbo Su, Kaijie Chen, Jing Yang, and Zheng Lin. MemMark: State-evolution attribution watermarking for agent long-term memory systems. *arXiv preprint arXiv:2605.25002*, 2026. URL https://arxiv.org/abs/2605.25002.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023. URL https://arxiv.org/abs/2307.13854.

A Detailed Algorithms of LoRo-Mark

This appendix uses the component names and notation introduced in Section 3.2. In particular, $|c|$ denotes the byte length of the copyright message, $n = 8|c|$ denotes its bit length, and $c = (c_0, \ldots, c_{n-1})$ denotes its binary representation. When the available parameter capacity supports only an ℓ-bit tag, we write

$$\mathrm{MAC}_\ell(k, x) := \mathrm{Trunc}_\ell(\mathrm{MAC}(k, x));$$

without truncation, MAC_ℓ is simply MAC. The parameter encoder $\mathcal{E}$ and decoder $\mathcal{D}$ use the same key k_{emb} and tag length ℓ. Their correctness requires

$$\mathcal{D}\big(\mathcal{E}(T_{\mathrm{body}}, \sigma, k_{\mathrm{emb}}), k_{\mathrm{emb}}\big) = \sigma$$

for every encodable ℓ-bit tag σ. At each step, o_t denotes the reproducible, domain-separated execution context, including the step identifier and the observation needed to reconstruct $\mathcal{B}_t$. We write WatermarkVerify for Algorithm 5.

Algorithm 1 Key Generation

Require: Security parameter λ
Ensure: $k_{\mathrm{det}}, k_{\mathrm{prv}}, k_{\mathrm{emb}}$
1: $k_{\mathrm{master}} \leftarrow \{0,1\}^\lambda$
2: $k_{\mathrm{det}} \leftarrow \mathrm{PRF}(k_{\mathrm{master}}, \texttt{detect})$ (Goldreich et al., 1986)
3: $k_{\mathrm{prv}} \leftarrow \mathrm{PRF}(k_{\mathrm{master}}, \texttt{prove})$
4: $k_{\mathrm{emb}} \leftarrow \mathrm{PRF}(k_{\mathrm{master}}, \texttt{embed})$
5: **return** $(k_{\mathrm{det}}, k_{\mathrm{prv}}, k_{\mathrm{emb}})$

Algorithm 2 Trigger Task Generation

Require: Valid task body T_{body}, keys $k_{\mathrm{det}}, k_{\mathrm{emb}}$, tag length ℓ
Ensure: Authenticated forensic request T^*
1: $\sigma \leftarrow \mathrm{MAC}_\ell(k_{\mathrm{det}}, T_{\mathrm{body}})$
2: $P \leftarrow \mathcal{E}(T_{\mathrm{body}}, \sigma, k_{\mathrm{emb}})$
3: $T^* \leftarrow T_{\mathrm{body}} \oplus P$
4: **return** T^*

Algorithm 3 Forensic Detection

Require: Incoming request $T = (T_{\mathrm{body}}, P)$, keys $k_{\mathrm{det}}, k_{\mathrm{emb}}$, tag length ℓ
Ensure: Detection result (s, σ), where $s \in \{0,1\}$
1: $\sigma_{\mathrm{expected}} \leftarrow \mathrm{MAC}_\ell(k_{\mathrm{det}}, T_{\mathrm{body}})$
2: $\sigma_{\mathrm{decoded}} \leftarrow \mathcal{D}(P, k_{\mathrm{emb}})$
3: **if** $\sigma_{\mathrm{decoded}} = \sigma_{\mathrm{expected}}$ **then**
4: **return** $(1, \sigma_{\mathrm{expected}})$
5: **else**
6: **return** $(0, \perp)$
7: **end if**

Algorithm 4 Redundant Encoding and Behavioral Embedding

Require: Copyright message c, repetition factor r, tag σ, key k_{prv}
Require: Reproducible execution contexts $(o_t, \mathcal{B}_t)$ with $|\mathcal{B}_t| \geq 2$
Require: Sufficient cumulative capacity to encode nr bits
Ensure: Forensic behavior sequence $\tau = (b_1, \ldots, b_S)$
$n \leftarrow 8|c|$; $M \leftarrow (c_0 \cdots c_{n-1}) \| \cdots \| (c_0 \cdots c_{n-1})$ with r copies
$k_{\text{step}} \leftarrow \text{MAC}(k_{\text{prv}}, \sigma)$
$\tau \leftarrow [\,]$; $p \leftarrow 0$; $t \leftarrow 1$
while $p < nr$ **do**
    $q_t \leftarrow \lfloor \log_2 |\mathcal{B}_t| \rfloor$
    $\pi_t \leftarrow \text{PRF}(k_{\text{step}}, o_t, \mathcal{B}_t)$
    $\ell_t \leftarrow \min\{q_t, nr - p\}$
    $u_t \leftarrow M[p : p + \ell_t - 1] \| 0^{q_t - \ell_t}$
    $j_t \leftarrow \text{int}(u_t)$; $b_t \leftarrow \pi_t[j_t]$
    Append b_t to τ; $p \leftarrow p + \ell_t$; $t \leftarrow t + 1$
end while
$S \leftarrow |\tau|$
return τ

Algorithm 5 Watermark Verification

Require: $T^* = (T_{\text{body}}, P)$, observed behavior sequence $\tau' = (b'_1, \ldots, b'_{S'})$
Require: Claimed message c_{claimed}, keys $k_{\text{det}}, k_{\text{prv}}, k_{\text{emb}}$, repetition factor r, tag length ℓ
Ensure: 1 if ownership verification succeeds; otherwise 0
$\sigma_{\text{expected}} \leftarrow \text{MAC}_\ell(k_{\text{det}}, T_{\text{body}})$
$\sigma_{\text{decoded}} \leftarrow \mathcal{D}(P, k_{\text{emb}})$
if $\sigma_{\text{decoded}} \neq \sigma_{\text{expected}}$ **then**
    return 0
end if
$n \leftarrow 8|c_{\text{claimed}}|$; $k_{\text{step}} \leftarrow \text{MAC}(k_{\text{prv}}, \sigma_{\text{expected}})$
$\widehat{M} \leftarrow [\,]$; $p \leftarrow 0$
for $t \in \{1, \ldots, S'\}$ **do**
    Reconstruct $(o_t, \mathcal{B}_t)$ from T^* and the exposed execution context
    $q_t \leftarrow \lfloor \log_2 |\mathcal{B}_t| \rfloor$
    $\pi_t \leftarrow \text{PRF}(k_{\text{step}}, o_t, \mathcal{B}_t)$
    $j'_t \leftarrow \text{IndexOf}(b'_t, \pi_t)$ ▷ $\perp$ if $b'_t \notin \mathcal{B}_t$
    $\ell_t \leftarrow \min\{q_t, nr - p\}$
    if $j'_t \neq \perp$ **and** $j'_t < 2^{q_t}$ **then**
        Append the first ℓ_t bits of $\text{bits}(j'_t, q_t)$ to $\widehat{M}$
    else
        Append ℓ_t erasure symbols to $\widehat{M}$
    end if
    $p \leftarrow p + \ell_t$
    if $p = nr$ **then**
        break
    end if
end for
if $p < nr$ **then**
    Append $nr - p$ erasure symbols to $\widehat{M}$
end if
$\hat{c} \leftarrow \text{MajorityDecode}(\widehat{M}, r, n)$ (Lin & Costello, 2004) ▷ Ignore erasures; reject empty groups or ties
return $\mathbb{I}[\hat{c} = c_{\text{claimed}}]$

B Supplementary Proofs

B.1 Correctness (No False Negatives)

Proposition 1 (No false negatives)**.** *Let c be the copyright message configured in the protected agent, let r be a positive odd repetition factor, and let T^* be produced by Trigger Task Generation using a correct parameter encoder–decoder pair. If τ is the intact forensic behavior sequence produced by Redundant Encoding and Behavioral Embedding under reconstructible execution contexts, then*

$$\mathrm{WatermarkVerify}(T^*, \tau, c, k_{\mathrm{det}}, k_{\mathrm{prv}}, k_{\mathrm{emb}}, r, \ell) = 1.$$

Proof. Trigger Task Generation constructs $T^* = (T_{\mathrm{body}}, P)$ with

$$\mathcal{D}(P, k_{\mathrm{emb}}) = \mathrm{MAC}_\ell(k_{\mathrm{det}}, T_{\mathrm{body}}) = \sigma.$$

Consequently, Forensic Detection returns $(1, \sigma)$ and activates the forensic branch. Both Behavioral Embedding and Watermark Verification derive $k_{\mathrm{step}} = \mathrm{MAC}(k_{\mathrm{prv}}, \sigma)$ and reconstruct the same $\pi_t = \mathrm{PRF}(k_{\mathrm{step}}, o_t, \mathcal{B}_t)$ at every step. Therefore, each intact behavior $b_t = \pi_t[j_t]$ is mapped back to the same index j_t and hence to the same q_t-bit chunk. Because the verifier retains only the first ℓ_t bits of the final chunk, their concatenation is exactly

$$M = (c_0 \cdots c_{n-1}) \| \cdots \| (c_0 \cdots c_{n-1}), \qquad n = 8|c|,$$

with r identical copies. Majority decoding thus returns $\hat{c} = c$, so Watermark Verification returns 1. □

B.2 Behavior-Sequence Unforgeability

The following one-shot statement excludes replay of an authentic forensic sequence generated for the same execution contexts. Such a freshness condition is necessary because any accepted sequence could otherwise be replayed verbatim.

Proposition 2 (Fresh-context behavior-sequence unforgeability)**.** *Let $n = 8|c|$ and let r be positive and odd. Consider a one-shot PPT adversary that knows T^*, k_{det}, and k_{emb}, but not k_{prv}, has no oracle access to the target per-step permutations, and has not observed an authentic forensic behavior for any of the pairwise distinct, domain-separated target execution contexts. Assume also that $n \geq \max_t q_t$. Suppose the derivation of k_{step} is $\varepsilon_{\mathrm{KDF}}(\lambda)$-pseudorandom and the domain-separated PRF evaluations are jointly indistinguishable from independent uniform per-step permutations with multi-context advantage $\varepsilon_{\mathrm{PRF}}(\lambda, S')$. Then*

$$\Pr\big[\mathrm{WatermarkVerify}(T^*, \tau', c, k_{\mathrm{det}}, k_{\mathrm{prv}}, k_{\mathrm{emb}}, r, \ell) = 1\big] \tag{12}$$

$$\leq 2^{-n} + \varepsilon_{\mathrm{KDF}}(\lambda) + \varepsilon_{\mathrm{PRF}}(\lambda, S'). \tag{13}$$

Proof. Replace the derivation of k_{step} by a uniform key and then replace the domain-separated PRF evaluations by independent uniform permutations. The two hybrid transitions incur advantages at most $\varepsilon_{\mathrm{KDF}}(\lambda)$ and $\varepsilon_{\mathrm{PRF}}(\lambda, S')$, respectively.

In the ideal experiment, a supplied nonmember behavior is an erasure. For each supplied $b'_t \in \mathcal{B}_t$, its index under the independent uniform permutation is uniform on $\{0, \ldots, |\mathcal{B}_t| - 1\}$. Condition on the resulting per-step active-or-erasure pattern. At every active step, conditioning on $j'_t < 2^{q_t}$ leaves j'_t uniform on $\{0, \ldots, 2^{q_t} - 1\}$, so the recovered payload positions at that step are independent unbiased bits. Independence of the ideal per-step permutations makes these bits independent across active steps.

Since $n \geq \max_t q_t$, the r positions associated with any fixed copyright bit lie in distinct steps. For each copyright bit, an empty group or a tie is rejected; otherwise bit-complement symmetry implies that its surviving majority equals the fixed target bit with probability at most $1/2$. Distinct copyright bits use disjoint random bit positions, so the conditional acceptance probability is at most 2^{-n}. Averaging over the active-or-erasure pattern and adding the two hybrid losses proves the claim. □

B.3 Robustness

For closed-form bounds matching the experiments, suppose $|\mathcal{B}_t| = 2^\beta$ at every embedding step, so each step carries the constant capacity $\beta = \log_2 |\mathcal{B}_t| \geq 1$. Let $n = 8|c|$, let r be positive and odd, and let

$$M = (c_0 \cdots c_{n-1}) \| \cdots \| (c_0 \cdots c_{n-1})$$

contain r copies and have length $N = nr$. Copy k of bit c_i occupies position $i + kn$ and step

$$t_{i,k} = \left\lfloor \frac{i + kn}{\beta} \right\rfloor + 1, \qquad S = \left\lceil \frac{nr}{\beta} \right\rceil.$$

Watermark Verification ignores erased copies and takes a majority over the surviving copies, rejecting when none survives or when a tie occurs.

Proposition 3 (Robustness to prefix truncation)**.** *Let $\alpha \in [0, 1]$. Suppose only the first $S' = \lfloor \alpha S \rfloor$ steps of an otherwise uncorrupted forensic behavior sequence are retained. If*

$$\alpha \geq \frac{\lceil n/\beta \rceil}{S} \approx \frac{1}{r}, \tag{14}$$

then Watermark Verification returns 1 *with probability* 1.

Proof. The retained prefix contains the first $S'\beta$ encoded bits. All positions $0, \ldots, n-1$, which form the first complete copy of c, are retained when $S'\beta \geq n$, equivalently when $S' \geq \lceil n/\beta \rceil$. The assumed lower bound on α guarantees this condition. Every copyright bit therefore has at least one surviving copy, and all surviving copies are correct. Erasure-aware majority decoding returns $\hat{c} = c$, so verification succeeds. □

Remark 1. *For the default configuration $n = 104$, $\beta = 6$, and $r = 3$, we have $S = 52$ and the sufficient retained-prefix threshold is $\lceil 104/6 \rceil / 52 = 18/52 \approx 34.6\%$.*

Proposition 4 (Robustness to random behavior substitution)**.** *Let $\rho \in [0, 1]$ and assume $n \geq \beta$. Independently at each step, replace the authentic behavior with probability ρ by a behavior sampled uniformly from $\mathcal{B}_t$. Then*

$$\Pr\left[\text{WatermarkVerify}(T^*, \tau', c, k_{\text{det}}, k_{\text{prv}}, k_{\text{emb}}, r, \ell) = 1\right] \geq 1 - n p_{\text{bit}}(\rho), \tag{15}$$

where, for $X \sim \text{Bin}(r, \rho/2)$,

$$p_{\text{bit}}(\rho) := \Pr\left[X \geq \left\lceil \frac{r}{2} \right\rceil\right]. \tag{16}$$

Proof. The r copies of c_i occur at positions $i, i + n, \ldots, i + (r-1)n$. Since $n \geq \beta$, these positions lie in distinct steps. Their substitution events and replacement draws are therefore independent under the stated attack model.

An unsubstituted copy is correct. Conditional on substitution, the replacement behavior is uniform in $\mathcal{B}_t$ and hence its index under the fixed permutation π_t is uniform in $\{0, \ldots, 2^\beta - 1\}$. Each decoded bit is then wrong with probability $1/2$, giving the unconditional per-copy error probability $\rho/2$. Consequently, the number of wrong copies of c_i follows $\text{Bin}(r, \rho/2)$, and majority decoding fails for this bit with probability $p_{\text{bit}}(\rho)$. A union bound over the n copyright bits gives a total failure probability at most $n p_{\text{bit}}(\rho)$, proving the result. □

Remark 2. *For $r = 3$,*

$$p_{\text{bit}}(\rho) = 3\left(\frac{\rho}{2}\right)^2 \left(1 - \frac{\rho}{2}\right) + \left(\frac{\rho}{2}\right)^3.$$

This equals approximately 0.007*,* 0.028*, and* 0.156 *at $\rho =$* 0.1*,* 0.2*, and* 0.5*, respectively. The union bound may be loose because different copyright bits can share step-level substitution events; no cross-bit independence is required for the bound.*