Cluster-Aware Attacks on Graph Watermarks

Alexander Nemecek
Case Western Reserve University
Cleveland, Ohio, USA
ajn98@case.edu

Emre Yilmaz
University of Houston-Downtown
Houston, Texas, USA
yilmaze@uhd.edu

Erman Ayday
Case Western Reserve University
Cleveland, Ohio, USA
exa208@case.edu

Abstract—Data from domains such as social networks, healthcare, finance, and cybersecurity can be represented as graph-structured information. Given the sensitive nature of this data and their frequent distribution among collaborators, ensuring secure and attributable sharing is essential. *Graph watermarking* enables attribution by embedding user-specific signatures into graph-structured data. While prior work has addressed random perturbation attacks, the threat posed by adversaries leveraging structural properties through community detection remains unexplored. In this work, we introduce a cluster-aware threat model in which adversaries apply community-guided modifications to evade detection. We propose two novel attack strategies and evaluate them on real-world social network graphs. Our results show that cluster-aware attacks can reduce attribution accuracy by up to 80% more than random baselines under equivalent perturbation budgets on sparse graphs. To mitigate this threat, we propose a lightweight embedding enhancement that distributes watermark nodes across graph communities. This approach improves attribution accuracy by up to 60% under attack on dense graphs, without increasing runtime or structural distortion. Our findings underscore the importance of cluster-topological awareness in both watermarking design and adversarial modeling.

I. INTRODUCTION

Many datasets are structured as graphs, representing relationships in various domains, including social networks [1], [2], biomedical research [3], and cryptographic applications [4], [5]. These graphs often contain sensitive information, making their secure storage, sharing, and analysis a critical concern. Researchers and organizations frequently share graph-structured data with trusted parties for collaborative analysis and real-world applications. However, the sensitive nature of this data raises significant security risks, including unauthorized access and data leakage among distributed entities. To mitigate these threats, data owners must ensure their datasets remain protected against adversarial modifications and that any unauthorized redistribution can be detected.

Watermarking, the process of embedding a detectable signature within an object, has been a widely studied area, including imaging [6], [7], audio [8]–[10], software [11], [12], databases [13]–[15], and machine learning models [16]–[19]. Graph-structured data is no exception; in a graph watermarking, a subgraph is embedded as a signature within the original graph, allowing for verification by extracting the watermark during detection [20]. While graph watermarking presents a promising solution for protecting against unauthorized access and redistribution, it remains a significantly understudied area,

notably in undirected graphs [20]–[22]. Moreover, existing watermarking schemes primarily consider random edge flipping attacks as the main adversarial threat model, failing to consider a more sophisticated adversary who can leverage the inherent community structure of graphs to maximize the effectiveness of an attack.

Most real-world graphs exhibit community structures [23], where nodes naturally form densely connected subgroups (clusters). An adversary who recognizes and understands these structures can develop targeted attack strategies that selectively modify edges within or between clusters to compromise graph watermark integrity more effectively than random edge modifications. However, existing literature does not explicitly consider adversaries that exploit community structures or employ clustering-based strategies when attempting to compromise graph watermarks. In Table I, we summarize the threat models described in existing graph watermarking literature.

The primary goal of this work is to address the notable gap in existing literature by introducing a clustering-aware threat model for graph watermarking. Our empirical analysis demonstrates that an adversary who considers community structure, rather than relying on random edge modifications, can more effectively evade detection, particularly when detection relies on exact subgraph extraction. To address this, we propose an alternative similarity-based detection strategy that remains consistent with the principles of watermark integrity, but tolerates more structural perturbations. To validate this claim, we focus on the earliest and most structured watermarking approach, introduced by Zhao et al. (2015) [20], which is regarded as the first practical system for embedding subgraphs in large graph-based networks. This watermarking scheme features a well-defined embedding-extraction methodology, making it a logical baseline for evaluating novel attack strategies.

Our key contributions can be summarized as follows:
1) We introduce a novel cluster-aware threat model for graph watermarking, where adversaries exploit community structure to add or remove edges. These attacks maintain distortion of random flipping but are significantly more effective at evading detection.
2) We develop a structure-based similarity approach for attribution, avoiding reliance on subgraph extraction.
3) We propose a robust watermark embedding strategy that distributes nodes across clusters to mitigate cluster-aware attacks without modifying the detection pipeline.

Paper (Author)	Random Edge Flipping	Collusion Attacks	Graph Anonymization	Clustering-Aware Attacks
Zhao et al. (2015) [20]	✓	✓	✓	✗
Eppstein et al. (2016) [21]	✓	✗	✓	✗
Bourrée et al. (2025) [22]	✓	✓	✗	✗

TABLE I: Overview of adversarial threat models addressed in existing graph watermarking literature.

4) We evaluate our approach on two real-world graphs, showing that cluster-aware attacks can reduce detection accuracy by up to 80% compared to random edge flipping while our mitigation improves accuracy by 30-60% under strong perturbations.

As graph-based data becomes increasingly critical to both the academic community and industry, ensuring the ability to trace and protect such data against evolving attack strategies is paramount. Our findings highlight that relying solely on random bit-flipping attack robustness is insufficient and that detection schemes must adapt to account for structure-aware adversarial behavior without relying purely on brittle exact-matching techniques. To the best of our knowledge, this work represents the first practical expansion of the graph watermarking threat model, demonstrating key vulnerabilities and introducing new detection metrics. By addressing these gaps, we aim to establish a more robust benchmark for future graph watermarking designs.

II. Background and Related Work

In this section, we review prior related work on graph watermarking and graph clustering methodologies.

A. Graph Watermarks

Graph watermarking is a subfield of digital watermarking that focuses on embedding signatures within graph-structured data. The goal of graph watermarking is to enable ownership verification and unauthorized redistribution detection while preserving the utility of the underlying graph. Formally, watermarking techniques embed a signature into an object, in this case, a graph G, producing a watermarked graph G'. The embedded signature is designed to be robust to modifications of G' and verifiable through a detection process that extracts and verifies its presence from a potentially modified graph $\hat{G}$.

Embedding watermarks in graph-structured data poses unique challenges. Graphs are inherently relational and exhibit complex structural properties, such as sparsity, community structure, and scale-free degree distributions. These characteristics constrain how signatures can be embedded without significantly distorting the graph's statistical properties or usability. As a result, graph watermarking schemes must carefully balance robustness, imperceptibility, and computational efficiency. Despite the increasing importance of graph-structured data, research on graph watermarking remains relatively sparse. To the best of our knowledge, only three practical watermarking schemes have been proposed in the literature: Zhao et al. (2015) [20], Eppstein et al. (2016) [21], and Bourrée et al. (2025) [22].

The earliest and most widely recognized graph watermarking scheme was proposed by Zhao et al. [20]. Their work

introduced the first practical method for embedding unique, user-specific watermarks into large graph datasets. The main idea of their scheme is to embed a randomly generated subgraph watermark W into the original graph G, resulting in a watermarked graph G'. The embedding process is performed in-band by selecting a subgraph S within G and modifying its structure to match the watermark graph W. This design preserves graph connectivity and limits structural distortion by avoiding the attachment of external subgraphs. The extraction process relies on regenerating W using a secret key and searching for its existence within a potentially modified graph $\hat{G}$ through guided subgraph matching. To evaluate the robustness of the watermark, Zhao et al. consider both a single-attacker model and a collusion attack model. The single-attacker model assumes an adversary who randomly modifies edges in the watermarked graph, while a collusion attack model involves multiple watermarked copies being compared to identify discrepancies and remove the watermark. However, their scheme assumes that adversaries act randomly or use naive collusion strategies. Notably, the threat model does not account for attackers who exploit the community structure or clustering patterns inherent in many real-world graphs. This would leave the watermark vulnerable to more sophisticated structural attacks.

Building on the work of Zhao et al., Eppstein et al. [21] proposed a framework and algorithmic foundations for graph watermarking. Their works extends Zhao et al.'s system-level approach by providing theoretical models and formal security guarantees regarding the feasibility and robustness of graph watermarking. Their scheme introduces an embedding process that modifies selected edges within the original graph G to encode a watermark identifier, resulting in a watermarked graph G'. Unlike Zhao et al., Eppstein et al. extend the embedding process in two random graph models, the Erdös-Rényi model[1] and a random power-law graph model[2], allowing them to theoretically analyze conditions under which watermarking is feasible and computationally secure. Specifically, they utilize structural properties of high- and medium-degree vertices, flipping edges between them to encode watermark bits while minimizing distortion. The extraction process labels vertices based on degree information and adjacency patterns, enabling the identification of the embedded watermark from a possibly modified graph $\hat{G}$.

The main improvement over Zhao et al.'s watermarking scheme is that Eppstein et al. formalize the notion of adversar-

---

[1]The Erdös-Rényi model generates graphs by connecting node pairs with a fixed probability, resulting in homogeneous degree distributions.

[2]Power-law graph models generate graphs with heterogeneous degree distributions, where a few nodes have high degrees and many nodes have low degrees, mimicking real-world networks.

ial advantage and provide provable security guarantees under specific random graph assumptions. They additionally introduce distance-based similarly measures to quantify adversarial modifications and evaluate the robustness of their scheme against arbitrary and random edge-flipping adversaries. However, similar to prior work, their threat model is limited to generic edge-flipping attacks and does not consider adversaries who can exploit higher-order structural information.

The most recent contribution to graph watermarking is the FFG scheme proposed by Bourrée et al. [22]. Unlike prior schemes which rely on specific subgraph matching or isomorphism techniques, FFG adapts image watermarking techniques to the graph domain. Specifically, the scheme treats the adjacency matrix of graph G as an image and embeds a watermark in the spectral domain using a Fourier transform-based approach. The embedding process first generates a watermark key sampled from a Gaussian distribution, which is inserted into the low-frequency components of the Fourier transform of G's adjacency matrix. After modifying these components, the inverse Fourier transform is applied, and the resulting matrix is binarized and symmetrized to produce the watermarked graph G'. The extraction is performed by comparing the spectral difference between a suspected graph $\hat{G}$ and the original graph, using a similarity threshold to determine the presence of the watermark.

FFG surpasses prior work in terms of computational efficiency. While Zhao et al. and Epstein et al. rely on NP-complete subgraph matching operations during extraction, FFG leverages spectral analyses with a time complexity of $O(N^2 \log N)$, making it scalable to graphs with millions of nodes. Additionally, Bourrée et al. empirically demonstrate that FFG achieves comparable or improved robustness to random edge-flipping attacks relative to prior schemes. However, similar to the prior work, the threat model considered in FFG is limited to random edge modifications and does not address adversaries who exploit structural properties.

B. Clustering in Graphs

Clustering refers to the process of partitioning a set of objects into groups based on a similarity metric. In the context of graphs, clustering or community detection, aims to identify densely connected subgroups of vertices (or nodes) that are sparsely connected to the rest of the graph. Given a graph $G = (V, E)$ where V is the set of vertices and E is the set of edges, clustering divides V into disjoint subsets $C = \{C_1, C_2, \ldots, C_k\}$ such that vertices within each cluster C_i are more densely connected to one another than to vertices outside the cluster.

Community structures are an inherent property of many real-world graphs [23], and identifying these communities allows for graph analysis. A widely used metric for evaluating the quality of a clustering is modularity [24], which measures the density of edges within clusters compared to a random graph with the same degree distribution. Modularity is defined as:

$$Q = \frac{1}{2m} \sum_{i,j} \left(A_{ij} - \frac{d_i d_j}{2m} \right) \delta(C_i, C_j) \tag{1}$$

where A_{ij} is the adjacency matrix of G, d_i and d_j are the degrees of vertices i and j, $m = |E|$ is the total number of edges, and $\delta(C_i, C_j)$ is 1 if vertices i and j belong to the same cluster, and 0 otherwise.

While clustering techniques provide insights into the structural properties of graphs, they can also inform adversarial strategies and expose vulnerabilities in graph-based systems. One prominent application of clustering in adversarial contexts is in graph de-anonymization attacks. For example, in social networks, data can be anonymized by removing names or obfuscating user IDs. However, an adversary can analyze community structures, such as friend groups or common interests, and cross-reference them with auxiliary datasets to re-identify users [25]. Additionally, clustering has been leveraged in adversarial graph manipulation attacks where adversaries have targeted graph neural networks (GNNs) by modifying community structures to poison the graph and degrade GNN performance [26]. These techniques involve designing targeted perturbations that modify specific regions of the graph based on communities. This enables the adversary to maximize disruption while minimizing the distortion of the attack.

As adversaries may exploit clustering to guide attacks, there are many algorithmic strategies available for discovering community structure in graphs. Each algorithm differs in its approach to clustering and reliance on user-specified parameters. These algorithms can be categorized based on whether they require explicit hyperparameter tuning. Clustering methods, such as spectral clustering [27] or Louvain/Leiden algorithms [28], [29], require the user to specify parameters such as the number of clusters or a resolution parameter. The choice of these parameters can significantly influence the clustering output and requires optimization of an objective function, such as modularity. The effectiveness therefore depends on the appropriate parameter selection, which can be nontrivial in an adversarial setting.

Other algorithms, such as greedy modularity maximization [24] and label propagation [30], operate without the need for hyperparameter tuning. In greedy modularity maximization, the algorithm directly optimizes the modularity score to detect communities, while label propagation assigns community labels based on iterative majority voting from neighboring nodes. The absence of tunable parameters in these algorithms makes them more attractive for adversarial use, as an attacker does not require prior knowledge of the graph's structural properties to apply them effectively. In our threat model, we leverage these parameter-free clustering algorithms to demonstrate that an attacker can conduct a successful attack without the need for fine-grained tuning or auxiliary information.

III. BASELINE GRAPH WATERMARKING

To validate the effectiveness of our proposed threat model and mitigation strategy, we focus on the earliest practical

graph watermarking scheme introduced by Zhao et al. [20]. This method embeds a user-specific watermark subgraph into a larger graph while minimizing structural distortion. The framework is composed of two key watermark components: embedding and extraction. To our knowledge, it represents the first practical implementation of watermarking techniques for graph-structured data.

A. Watermark Embedding

The embedding process consists of four main steps: **(I)** generating a random generator seed Ω_i, **(II)** generating the watermark graph W_i, **(III)** selecting the placement of W_i within the original graph G, and **(IV)** embedding W_i into a subgraph S of G.

(I) Generating a random generator seed Ω_i. A random seed Ω_i is derived from user i's RSA key pair $\langle K_{\mathrm{pub}}^i, K_{\mathrm{priv}}^i \rangle$ and a graph-specific key K^G held by the data owner. The framework begins by sending a timestamp T to the user, who then signs it using their private key K_{priv}^i to produce the signature $K_{\mathrm{priv}}^i(T)$. This signature is then verified using the user's public key K_{pub}^i. Once verified, the data owner combines $K_{\mathrm{priv}}^i(T)$ with K^G to generate Ω_i. This setup ensures neither the user or the data owner can independently compute Ω_i.

(II) Generating the watermark graph W_i. The watermark graph W_i is generated as an Erdös-Rényi random graph $G(k, p)$ using Ω_i as the random seed. The graph consists of k nodes and includes each potential edge independently with probability p. To ensure uniqueness and compactness of the watermark, Zhao et al. configure $p = \frac{1}{2}$ and define the node count as $k \geq (2 + \delta) \log_q n$, where $n = |V|$ is the number of nodes in the original graph G, $q = \frac{1}{\max(p, 1-p)}$, and $\delta > 0$ is a small constant. This choice of parameters minimizes both the number of nodes and the average edge count, thereby reducing distortion and improving robustness.

(III) Selecting the watermark placement in G. A subgraph S of G is selected to host the watermark. Specifically, k nodes are chosen from G based on their local structure rather than metadata, to mitigate node ID anonymization. Each node is assigned a *Node Structure Description* (NSD), defined as a sorted list of its neighbors' degrees. For example, a node with neighbors of degrees 1, 7, 4, and 2 would have an NSD of [1, 2, 4, 7]. These NSDs are hashed using a secure hash function (e.g., SHA-1), and Ω_i is used to generate k target hash values. Nodes whose hashed NSDs match these values are selected. To avoid collisions, nodes are deterministically ordered (e.g., by original ID), and selection proceeds. This results in an ordered node set $X = \{x_1, x_2, ..., x_k\}$, and the corresponding subgraph $S = G[X]$.

(IV) Embedding the watermark graph W_i into S. The final step embeds W_i into the selected subgraph $S = G[X]$ by mapping its nodes $\{v_1, ..., v_k\}$ to the nodes $\{x_1, ..., x_k\}$ in S, establishing a one-to-one correspondence $f : v_i \mapsto x_i$. Each edge in W_i is then embedded into S using an XOR-based edge-flipping operation. For each possible edge (v_i, v_j), if an edge exists in W_i, the corresponding edge (x_i, x_j) in S is flipped (i.e., added if absent or removed if present). If no edge exists in W_i, the corresponding edge in S remains unchanged. This operation encodes the structure of W_i into S while minimizing distortion to the rest of the graph. To ensure subgraph connectivity, edges are also explicitly added between consecutive nodes in the ordered set X, forming a path $(x_1, x_2), (x_2, x_3), ..., (x_{k-1}, x_k)$. The resulting subgraph, denoted S^{W_i}, replaces S in the original graph, producing the final watermarked graph G^{W_i}. Before distribution, the graph is anonymized by randomly reassigning node IDs to further obfuscate the watermark.

B. Watermark Extraction

To determine whether a user-specific watermark subgraph W_i is embedded in a target graph G', the extraction process includes three main steps: **(I)** regenerating the watermark, **(II)** identifying candidate watermark nodes in G', and **(III)** detecting the embedded subgraph S^{W_i} within G'.

(I) Regenerating the watermark. Given the original graph G, the graph key K^G, and the user's signed timestamp $K_{\mathrm{priv}}^i(T)$, the data owner reconstructs the random seed Ω_i used during the embedding. Using this seed, the owner regenerates the watermark graph W_i, identifies the k ordered node set $X = \{x_1, x_2, \ldots, x_k\}$ used in the original embedding, computes their corresponding NSD labels, and reconstructs the subgraph S^{W_i} that was embedded in the watermarked graph G^{W_i}.

(II) Identifying candidate nodes in G'. Using the NSD labels of the k nodes in X, the data owner searches G' for all nodes whose NSD labels match those of $x_j \in X$. This results in a candidate set C_j for each node x_j which contain all possible matches in G'. The candidate set may be large due to multiple nodes in G' sharing the same NSD label. To reduce the search space, the algorithm applies a structural pruning step where for every pair of nodes x_m and x_n connected in S^{W_i}, the corresponding candidate sets C_m and C_n are refined by eliminating any node in C_m that is not adjacent to any node in C_n, and vice versa. This pruning process is repeated iteratively, resulting in a reduced and structure-consistent set of candidates $\{C_1, C_2, \ldots, C_k\}$ on G'.

(III) Detecting the embedded subgraph. With the refined candidate sets, the algorithm performs a recursive search to determine whether the subgraph S^{W_i} exists in G'. A list $Y = \{y_1, y_2, \ldots, y_m\}$ tracks partial mappings, where each y_j corresponds to a tentative match for x_j. The process recursively explores combinations of candidates from the sets C_j, checking whether a subgraph isomorphic to S^{W_i} can be reconstructed. If such a match is found, the watermark is considered detected; otherwise, the graph is either unwatermarked or has been altered beyond detection.

C. Evaluation Scope

To define the scope of our evaluation, we summarize aspects of Zhao et al.'s threat model and experimental design. In addition to addressing random perturbation and collusion-based attacks, Zhao et al. propose a series of enhancements to improve watermark robustness. These include modifications

to the extraction process to tolerate structural noise, such as approximate NSD label matching using a threshold for similarity and approximate subgraph matching that tolerates a bounded edge difference between the extracted and original watermark subgraphs.

While these improvements enhance robustness against random modifications, they introduce ambiguity in the interpretation of extraction outcomes by relaxing matching criteria. Threshold-based matching of nodes and subgraphs can lead to uncertain attribution outcomes in graphs with repeating local structures. In contrast, our evaluation avoids redundancy and heuristic thresholds to isolate how cluster-aware attacks impact detection robustness.

While Zhao et al.'s method relies on guided subgraph matching for watermark recovery, we find that even small perturbations (as little as 1% of edge modifications) can lead to extraction failure, consistent with their findings. As a result, we adopt an alternative detection approach based on dK-2 similarity, which compares the structural distribution of degree pairs between the leaked graph and each of the individually watermarked graphs. This enables robust user attribution without relying on approximate subgraph matching or threshold-based heuristics. Our evaluation focuses on assessing how community-aware perturbations affect this dK-2-based detection accuracy under realistic adversarial conditions. Our goal is to highlight a precise vulnerability in existing graph watermarking schemes that is not addressed by Zhao et al.'s threat model or robustness extensions.

IV. CLUSTER-AWARE THREAT MODEL

Building on the baseline watermarking approach presented in Section III, we now consider a more capable and structurally aware adversary. Prior evaluations, including those in Zhao et al. [20], primarily focus on random edge perturbation or collusion-based attacks. These models assume adversaries act with limited insight into the graph's topology. In contrast, our work introduces a novel threat model in which an attacker leverages the graph's inherent community structure to launch more targeted and effective attacks.

Prior work has shown that watermarking schemes can be relatively robust against random perturbation attacks, particularly in a single-attacker scenario. While some works extend this threat model to include more complex adversaries, such as colluding users or those targeting structural properties (i.e., vertex density and degree distributions) [21], there is no focus on attackers that explicitly exploit a graph's community structure to guide their modifications. To the best of our knowledge, this is the first study to model a clustering-aware adversary, who actively leverages structural communities to selectively add or remove edges in a way that degrades watermark integrity. This allows the attacker to strategically perturb edges within or between communities to disrupt the watermark while preserving overall graph utility.

A. Adversarial Assumptions

Here, we define the capabilities, knowledge, and objectives of the adversary. The setting assumes a data owner who maintains sensitive graph-structured data and embeds a user-specific watermark before distributing individualized watermarked graphs to a number of recipients. One of these recipients acts maliciously by leaking their copy of the graph to an external party. We refer to this user as the leaker, and the remaining recipients as non-leakers. We assume a single-attacker model and do not consider collusion between multiple recipients. This choice reflects a more conservative adversary with limited access, where they are unable to compare their version of the graph with others to isolate or reverse-engineer the watermark. While collusion attacks are a valid threat model explored in prior work [20], our focus is on cluster-aware single-copy strategies, which have been comparatively unexplored.

Capabilities. The adversary has full access to the watermarked graph G' distributed to them. They do not possess the original graph G, the watermark generation key K^G, or the user-specific key used to generate the signature. The attacker cannot regenerate the embedded watermark or directly identify the subgraph in which it was placed. However, they are free to perform structural analysis and modifications on G' prior to leaking it. We assume the attacker is capable of running unsupervised community detection algorithms on G' and that the watermarking process does not fully distort the underlying community structure.

Knowledge. The attacker does not know for certain whether the graph has been watermarked, nor do they know which part of the graph contains a watermark. Since we do not consider colluding attackers, the adversary cannot compare their copy with others to confirm differences. This work focuses on the single-copy threat model, but our framework could be extended in future work to address colluding adversaries by simulating shared graph comparisons. Instead, the attacker must assume the graph might be watermarked and act preemptively. Their strategy is based on the structural properties of the graph, such as community modularity and local density, which are derived directly from G' itself. They operate under the assumption that meaningful community structure remains intact after watermark embedding.

Goals. The adversary's primary goal is to distort or remove the embedded watermark W_i to prevent successful extraction and identification by the data owner. A secondary but important objective is to preserve the graph's overall utility. The attacker aims to introduce minimal distortion, avoiding significant changes to the graph's structure or visual layout that might degrade usability.

Having defined the assumptions and capabilities of our clustering-aware adversary, we now describe the specific attack strategies such an adversary can employ to compromise watermark integrity.

V. Proposed Cluster-Aware Attack Strategy

We describe the proposed attack strategy of an adversary who exploits the inherent community structure present in real-world graph data. The attacker begins by applying an unsupervised, parameter-free community detection algorithm to the watermarked graph G'. We assume that the watermark embedding process preserves the underlying community structure of the graph, which is necessary both for maintaining utility and for enabling adversarial analysis. Given that the attacker operates without access to the detection mechanism or feedback from the watermarking system, we restrict the attack to use parameter-free clustering algorithms. This reflects a realistic adversarial model, where reliance on hyperparameter tuning or privileged information is infeasible.

Once communities are identified, the attacker selectively adds or removes edges in the graph. Edge addition involves selecting two nodes and adding an edge between them if one does not already exist, while edge deletion removes an existing edge between a selected pair of nodes. The decision to add or remove an edge is based on whether it lies within a community (intra-cluster) or between communities (inter-cluster). Intra-cluster modifications affect structural cohesion by making communities overly dense (via addition) or sparse (via deletion), while inter-cluster perturbations blur boundaries between communities, reducing modularity and making structural patterns less distinct. We define two combined attack strategies that exploit both of these dimensions:

1) **Strategy I**: Intra-cluster addition and inter-cluster removal, densifying communities while breaking clear boundaries between them.
2) **Strategy II**: Intra-cluster removal and inter-cluster addition, weakening internal community structure while injecting noise between communities.

A. Strategy I: Intra-Add/Inter-Remove

The intra-add/inter-remove attack strategy focuses on densifying communities while weakening the structural separation between them. The goal of this attack is to increase the density within clusters such that it might introduce structural ambiguity and remove edges between clusters to blur inter-community boundaries which may interfere with watermarking schemes that rely on modular structure for detection.

The attack is conducted by first partitioning the graph into communities using a chosen clustering algorithm. In this work, we use parameter-free clustering algorithms to avoid the need to fine-tune hyperparameters that are apparent in non-parameter-free clustering algorithms. Given a number of flips, for each edge, the attacker randomly decides whether to perform an intra-cluster addition or an inter-cluster removal. For an intra-cluster addition, a cluster is selected at random. Within that chosen cluster, two distinct nodes that are not yet connected are selected and an edge is added between them. For inter-cluster removal, an existing edge connecting two nodes from different clusters is selected at random. The corresponding edge is then removed from the graph. The

---

Algorithm 1 Intra-Add/Inter-Remove Attack

1: **Input:** Graph $G = (V, E)$, clustering map $\mathcal{C}$, number of flips n
2: $G' \leftarrow$ copy of G
3: $F \leftarrow 0$
4: **while** $F < n$ **do**
5:     Sample $\alpha \sim \mathcal{U}(0,1)$
6:     **if** $\alpha < 0.5$ **then**
7:       Select random cluster c
8:       Choose $u, v \in \mathcal{C}^{-1}(c)$ such that $(u, v) \notin E(G')$
9:       **if** such u, v exist **then**
10:         Add edge (u, v) to G'
11:         $F \leftarrow F + 1$
12:       **end if**
13:     **else**
14:       Pick $(u, v) \in E(G')$ such that $\mathcal{C}(u) \neq \mathcal{C}(v)$
15:       **if** such (u, v) exists **then**
16:         Remove edge (u, v) from G'
17:         $F \leftarrow F + 1$
18:       **end if**
19:     **end if**
20: **end while**
21: **return** G'

---

attacker continues flipping edges until the predefined budget (e.g., total number of flips) is exhausted. We describe the attack in Algorithm 1.

By increasing intra-cluster density, the attacker creates more locally similar neighborhoods, which can reduce the uniqueness of the watermark nodes. Additionally, by removing inter-cluster edges, the attacker flattens the modularity of the graph. Flattening the modularity results in community boundaries becoming less distinguishable and thus potentially obscuring the watermark.

B. Strategy II: Intra-Remove/Inter-Add

This strategy takes an opposite approach where the intra-remove/inter-add attack strategy weakens internal cluster cohesion while injecting structural noise across communities. The goal of an attacker is to disrupt the natural topology of clusters while reducing the graph's community structure. Both approaches are likely to undermine the assumptions of structure-based watermarking schemes which depend on community coherence.

Similar to its counterpart, this attack first clusters the graph using a chosen clustering algorithm. We again only use parameter-free clustering approaches to mitigate the need for fine-tuning hyperparameters. Given a certain number of flips, for each flip, the attack randomly decides between intra- or inter-cluster perturbation. For intra-cluster removal, an existing edge between two nodes in the same cluster is randomly selected and removed. For inter-cluster addition, two nodes from different clusters are selected such that they are not currently connected and an edge is then added between them.

This process continues until the attack exhausts their edge modification budget. We describe this attack in Algorithm 2.

Algorithm 2 Intra-Remove/Inter-Add Attack

1: **Input:** Graph $G = (V, E)$, clustering map $\mathcal{C}$, number of flips n
2: $G' \leftarrow$ copy of G
3: $F \leftarrow 0$
4: **while** $F < n$ **do**
5:     Sample $\alpha \sim \mathcal{U}(0,1)$
6:     **if** $\alpha < 0.5$ **then**
7:         Pick $(u, v) \in E(G')$ such that $\mathcal{C}(u) = \mathcal{C}(v)$
8:         **if** such (u, v) exists **then**
9:             Remove edge (u, v) from G'
10:             $F \leftarrow F + 1$
11:         **end if**
12:     **else**
13:         Pick clusters $c \neq c'$
14:         Choose $u \in \mathcal{C}^{-1}(c)$, $v \in \mathcal{C}^{-1}(c')$ such that $(u, v) \notin E(G')$
15:         **if** such u, v exist **then**
16:             Add edge (u, v) to G'
17:             $F \leftarrow F + 1$
18:         **end if**
19:     **end if**
20: **end while**
21: **return** G'

This attack strategy disrupts the internal structure of communities by deleting intra-cluster edges, which can directly affect the nodes within the watermark subgraph. At the same time, inter-cluster additions create artificial cross-cluster connections, effectively reducing modularity and increasing the likelihood that the watermark becomes structurally indistinct.

C. Adversarial Objective

The objective of the adversary is to disrupt the watermarking scheme by modifying the watermarked graph G' such that the watermark cannot be reliably attributed to the leaker. Unlike prior work, which evaluates attack success based on failure of watermark extraction, we measure the adversary's success based on a reduction in detection accuracy using our similarity-based attribution scheme. A successful attack results in the leaked graph having higher similarity to another (non-leaking) recipient's watermarked graph than to the actual leaker's version, thereby misleading the data owner's attribution process.

To remain stealthy, the adversary is constrained by a perturbation budget, expressed as a percentage of total edges flipped. We evaluate attacks under varying budget levels to understand the relationship between attack strength and attribution accuracy. The attack is designed to exploit structural properties of the graph in a way that degrades attribution accuracy while preserving the graph's overall usability and statistical integrity.

VI. Experimental Evaluation

In this section, we describe the experimental setup used to evaluate our cluster-aware attack strategies. We first introduce the graph datasets used in our study and outline the parameters for watermark embedding. We then detail our dK-2 similarity-based detection mechanism, describe the configuration of each attack strategy, and define the metrics used to assess attack effectiveness. The results of these evaluations are presented in the following section (Section VII).

A. Datasets

We evaluate our attacks on two real-world graph datasets from the SNAP network dataset collection [31], both representing large-scale social networks with differing structural characteristics. These datasets were selected to capture variation in graph size, density, and clustering behavior, which are important factors that influence both the feasibility of watermark embedding and the effectiveness of structure-aware adversarial attacks.

The two graphs used are as follows: *Facebook (Social Circles)* [32] is a social graph derived from ego networks collected via a Facebook app. Nodes represent users, and edges represent friendship ties. The graph is relatively dense and exhibits strong clustering behavior, making it a realistic setting for structure-aware attacks. *LastFM Asia Social Network* [33] is a sparse social graph of LastFM users in Asian countries, constructed from mutual follower relationships. This dataset presents a complementary structure with lower average degree and clustering, enabling evaluation under more challenging conditions for watermarking. For clarity, we refer to the *Facebook (Social Circles)* dataset as *Facebook* and the *LastFM Asia Social Network* dataset as *LastFM* throughout the remainder of the paper. Table II summarizes the key structural properties of each dataset, including node and edge counts, average degree, clustering coefficient, graph density, and number of connected components.

To ensure each graph is a valid host for watermark embedding, we follow the feasibility criteria introduced by Zhao et al. [20]. In their framework, a watermark is modeled as a random Erdös-Rényi subgraph S^{W_i} with edge probability $p = 0.5$ and size $k = (2 + \delta) \log_2 n$, where n is the number of nodes in the original graph. This results in an expected watermark node degree of $(k + 1)/2$ and an average subgraph density of $\frac{\binom{k}{2} + k - 1}{2}$. For a graph G to be considered suitable for watermark embedding, two structural conditions must be met:

- **Node Degree Criterion**: The expected watermark node degree, $(k + 1)/2$, must lie within the range of node degrees observed in the host graph. That is, $N_{\min}(G) \leq (k + 1)/2 \leq N_{\max}(G)$.
- **Subgraph Density Criterion**: The expected watermark subgraph density must fall within the density range of k-node subgraphs in the host graph that have at least $(k + 1)/2$. That is, $D_{\min}(k) \leq \frac{\binom{k}{2} + k - 1}{2} \leq D_{\max}(k)^2$.

Graph	# of Nodes	# of Edges	Avg. Deg.	Clustering Coef.	Density	Connected Components
Facebook	4039	88234	43.69	0.6056	0.0108	1
LastFM	7624	27806	7.29	0.2194	0.0010	1

TABLE II: Structural properties of the *Facebook* and *LastFM* graphs, including node and edge counts, average degree, clustering coefficient, graph density, and the number of connected components.

Graph	k	Node Degree Criterion		Subgraph Density Criterion		Suitability
		$(k+1)/2$	$[N_{min}(G), N_{max}(G)]$	Watermark	$[D_{min}(k), D_{max}(k)]$	
Facebook	28	15	[1, 1045]	202.5	[32, 378]	**Yes**
LastFM	30	16	[1, 216]	232.0	[33, 352]	**Yes**

TABLE III: Evaluation of dataset suitability for watermark embedding based on Zhao et al.'s [20] criteria, including node degree and subgraph density thresholds for each selected graph.

We apply these criteria to both datasets and summarize the results in Table III. Our analysis confirms that each graph satisfies the node degree and subgraph density conditions, and is therefore appropriate for watermark embedding under Zhao et al.'s framework.

B. Watermarking Parameters

For the watermark embedding process, we embed a single watermark per graph, rather than multiple redundant watermarks, in order to isolate the effectiveness of our cluster-aware attack strategies compared to random perturbation. We adopt the same watermark parameters used by Zhao et al., setting $\delta = 0.3$ and $p = 0.5$. This results in watermark sizes of $k = 28$ for *Facebook* and $k = 30$ for *LastFM*, satisfying the watermark feasibility condition outlined in the previous section. Following the embedding process, all graphs were anonymized by randomly relabeling node identifiers. While our embedding procedure follows Zhao et al.'s in full, we differentiate from their extraction method by using a dK-2 similarity-based detection mechanism.

C. Detection via dK-2 Similarity

Previous works [20], [21] have used the dK-2 deviation as a structural distortion metric to evaluate how much an adversarial attack perturbs the watermarked graph. We adopt this same formulation not only for measuring distortion, but also as a detection and attribution mechanism. Specifically, we propose a similarity-based detection strategy based on the dK-2 series that compares a leaked, perturbed graph against a known set of watermarked versions.

The dK-2 series [34] characterizes the structural signature of a graph by capturing its joint degree distribution. For a given graph G, the dK-2 series is defined as the normalized count of degree pairs across all edges. That is, for every edge $(u, v) \in E$, where $d_u = \deg(u)$ and $d_v = \deg(v)$, we increment a count associated with the tuple $(\min(d_u, d_v), \max(d_u, d_v))$. This process yields a histogram of degree pair frequencies, which is then normalized by the total number of edges to obtain a probability distribution:

$$\text{dK-2}_G(i, j) = \frac{1}{|E|} \cdot |\{(u, v) \in E : (\min(\deg(u), \deg(v)),$$
$$\max(\deg(u), \deg(v))) = (i, j)\}| \quad (2)$$

We use the unordered joint degree distribution (JDD), where each undirected edge contributes once to the bin $(\min(d_u, d_v), \max(d_u, d_v))$, ensuring the resulting distribution sums to 1. This differs from the ordered JDD convention, in which off-diagonal bins receive contributions from both (d_u, d_v) and (d_v, d_u), resulting in values that are exactly twice those in our representation.

To compare two graphs G_1 and G_2, we compute the Euclidean distance between their dK-2 series and apply an exponential decay to produce a similarity score:

$$D(i, j) = (\text{dK-2}_{G_1}(i, j) - \text{dK-2}_{G_2}(i, j))^2 \quad (3)$$

$$\text{sim}(G_1, G_2) = \exp\left(-\sqrt{\frac{1}{|\mathcal{K}|} \sum_{(i,j) \in \mathcal{K}} D(i, j)}\right) \quad (4)$$

where $\mathcal{K}$ is the union of all (i, j) degree-pair bins that appear in either G_1 or G_2. This score lies in the interval $(0, 1]$, with higher values indicating stronger structural similarity.

In our setting, we simulate a watermarking scenario with 10 distinct parties, each receiving a unique watermarked version of the graph. One party is designated as the leaker, and their copy is perturbed using a clustering-aware or random attack. To identify the leaker, we compute the dK-2 similarity between the leaked graph and each of the 10 original watermarked graphs. The graph with the highest similarity score is identified as the most likely source. We follow the 10-party setting used in prior watermarking work [20], [21], which balances attribution granularity and computational cost. While increasing the number of parties could reduce similarity resolution, and increase detection complexity, this configuration remains practical and consistent with established protocols.

We choose dK-2 similarity over strict watermark extraction for two reasons. First, even minor perturbations (as low as 1% edge modifications) are known to break exact subgraph extraction [20], making it unreliable under realistic adversarial settings. Second, dK-2 similarity is both efficient to compute and sensitive to structural changes that arise from our clustering-based attacks. Since these attacks are designed to preserve global utility while subtly degrading local community structure, a statistical distributional approach like dK-2 is well-suited to capture these subtle deviations without relying on subgraph isomorphism.

LastFM (Greedy Modularity) LastFM (Label Propagation) Facebook (Greedy Modularity) Facebook (Label Propagation)
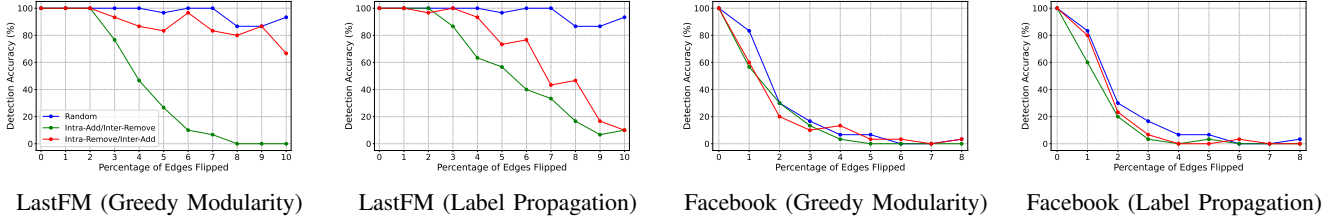
Fig. 1: Attribution accuracy of the dK-2 detection scheme under increasing perturbation. Cluster-aware attacks consistently outperform the random baseline, with greater impact seen in sparser graphs (*LastFM*).

LastFM (Greedy Modularity) LastFM (Label Propagation) Facebook (Greedy Modularity) Facebook (Label Propagation)
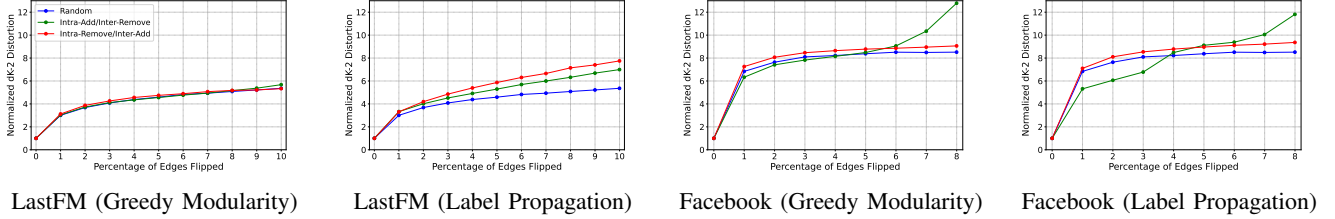
Fig. 2: dK-2 distortion introduced by each attack strategy. Distortion increases with edge flips, but cluster-aware attacks evade detection more effectively while maintaining equal or lower structural distortion than random attacks.

D. Attack Configuration

To evaluate the effectiveness of our cluster-aware attack strategy, we compare it against a random edge flipping baseline. We vary the total number of edge flips from 1% to either 8% or 10%, depending on the dataset and how quickly attacks reduce detection accuracy to near zero. This threshold-based cap ensures that we avoid unnecessary evaluation once attacks are clearly effective. Each attack trial is independently initialized. For each flip percentage, we generate a fresh perturbed graph starting from the original, rather than incrementally applying modifications. This introduces greater randomness and avoids bias from cumulative distortion. Consistent with prior studies [20], [21], our setup simulates watermarking across 10 distinct parties. However, we increase the number of trials per flip percentage from 10 (as used in prior work) to 30, in order to obtain more statistically stable results across randomized trials.

A key design choice in our cluster-aware threat model is the use of a clustering algorithm to guide intra- and inter-cluster edge modifications. We exclusively use parameter-free clustering methods to reflect a black-box attacker who lacks access to optimal hyperparameters. Specifically, we evaluate two algorithms: *Label Propagation*, a fast, unsupervised technique that assigns node labels through iterative majority voting among neighbors, and *Greedy Modularity*, which constructs communities by greedily optimizing modularity (as defined in Equation 1). These algorithms are commonly used in unsupervised settings and impose no manual tuning burden on the attacker. While an attacker could, in theory, evaluate dK-2 similarity against their own perturbed graph, we assume no collusion or access to other users' watermarked versions. As a result, the attacker operates under a single-copy, structure-only

threat model without feedback from the detection process.

E. Evaluation Metrics

We evaluate the effectiveness of our cluster-aware attack strategies in comparison to the random edge flipping baseline using three metrics: detection accuracy, structural distortion, and runtime performance.

Detection Accuracy. We measure the accuracy of our dK-2 similarity-based detection method, as defined in Equations 3 and 4. For each trial, the leaked graph is compared to the 10 individually watermarked graphs, and the graph with the highest similarity score is selected as the presumed source. Detection is considered successful if this prediction matches the true leaker. We report average detection accuracy across 30 randomized trials per flip percentage to quantify how quickly different attack strategies degrade attribution performance.

Structural Distortion. To evaluate how much the attacks perturb the original graph structure, we compute the dK-2 deviation between the perturbed graph and the original watermarked version. This metric, used in prior work [20], [21], calculates the Euclidean distance between the dK-2 series of the two graphs. Unlike the detection metric, this distortion score is reported as a raw distance without applying an exponential decay.

Runtime. We measure the average runtime of each attack strategy to assess their computational cost. This includes both the random edge flipping and the two cluster-aware strategies across all datasets. Runtime is averaged over the same 30 trials used in the evaluation of detection and distortion.

VII. RESULTS

We now highlight the results of our evaluation, comparing the performance of cluster-aware and random attack strategies.

Graph	Random			Label Propagation						Greedy Modularity					
				IA/IR			IR/IA			IA/IR			IR/IA		
	1%	3%	5%	1%	3%	5%	1%	3%	5%	1%	3%	5%	1%	3%	5%
Facebook	0.77	1.59	2.45	6.26	19.00	73.61	5.22	15.00	24.92	5.21	15.65	26.42	5.36	15.31	25.36
LastFM	0.42	0.93	1.43	0.72	1.83	2.97	0.57	1.37	2.16	0.56	1.46	2.39	0.55	1.36	2.13

TABLE IV: Average runtime (seconds) for each attack strategy. Runtime increases with graph size and flip percentage, with cluster-aware attacks being more computationally intensive than random edge flipping.

A. Attribution Accuracy

To evaluate how each attack strategy affects leaker attribution under our dK-2 similarity-based scheme, we identify the party whose watermarked graph has the highest dK-2 similarity to the leaked graph. We report attribution accuracy as the proportion of trials (out of 30) in which the true leaker is correctly identified based on dK-2 similarity.

On the *LastFM* dataset, the cluster-aware attacks outperform the random baseline in degrading detection accuracy. The intra-add/inter-remove strategy is more effective, causing sharper declines under both clustering algorithms, with *Greedy Modularity* yielding the most rapid drop. The intra-remove/inter-add attack also reduces attribution accuracy, performing comparably to intra-add/inter-remove under *Label Propagation*, but is less effective under *Greedy Modularity*, though it still outperforms random flipping in most cases.

Results on the *Facebook* dataset are more tightly grouped. While all attacks lead to a rapid drop in accuracy, cluster-aware strategies degrade performance slightly faster than random flipping during early perturbation stages. However, attribution accuracy for all methods converges to near zero as the number of perturbed edges increases.

These trends can be partially explained by the differences in the structural characteristics of the two graphs, summarized in Table II. *Facebook*'s higher density and clustering coefficient make its watermark structure more robust at baseline but also cause all attacks to converge quickly in their impact. In contrast, *LastFM*'s sparse structure and lower clustering coefficient result in greater sensitivity to targeted perturbations. Cluster-aware attacks are effective here, as they disrupt weakly cohesive clusters, which leads to early breakdowns in detection. These findings suggest that graph topology plays a critical role in determining watermark resilience. Sparse graphs with weak community cohesion are particularly vulnerable to cluster-aware attacks, even under low perturbation budgets. Figure 1 illustrates these trends, showing the attribution accuracy across both datasets and clustering methods as edge flip percentages increase.

B. dK-2 Distortion

We evaluate the impact of each attack strategy by computing the dK-2 deviation between the original watermarked graph and its perturbed version. This metric quantifies how edge modifications affect the global structure of the graph. As expected, all attack strategies introduce increasing distortion as the percentage of flipped edges increases. Across both datasets, cluster-aware attacks achieve greater reductions in attribution accuracy without incurring significantly more dK-2 deviation than the random baseline.

On *LastFM*, which is sparser and exhibits weaker clustering, all strategies produce similar distortion levels, but the cluster-aware variants are more effective at evading detection. On *Facebook*, which has a higher edge count and stronger community structure, distortion scales with the number of flips across all strategies. The intra-add/inter-remove strategy occasionally results in slightly higher distortion at larger flip percentages, though this occurs after detection has already collapsed.

These results reinforce the efficiency of cluster-aware attacks by degrading detection performance more quickly while introducing comparable, overall structural disruption. This efficiency is due to the attacks' ability to selectively perturb meaningful regions of the graph, such as community boundaries or internally fragile clusters. Figure 2 provides a detailed comparison of dK-2 distortion across attack strategies.

C. Attack Runtime

To assess the computational cost across different graphs, clustering methods, and perturbation levels, we evaluate the average runtime for each attack strategy, described in Table IV. As expected, runtime increases with both graph size and the percentage of flipped edges.

Cluster-aware attacks generally take longer than random edge flipping due to the additional overhead of targeted edge selection. Between the two clustering algorithms, *Label Propagation* is typically faster than *Greedy Modularity* due to its iterative level-passing approach. However, an exception occurs on *Facebook* under the intra-add/inter-remove strategy, where *Label Propagation* incurs higher runtime. We attribute this to the increased search space and collision handling required when adding edges within densely packed communities. These tightly connected clusters create more candidate pairs, increasing the cost of maintaining graph consistency during modification.

Despite this, all cluster-aware attacks remain computationally practical. Even the highest observed runtime (approximately 73.6 seconds on *Facebook* at 5% edge flips) falls well within the bounds of a realistic offline attack. This further supports the feasibility of community-guided perturbations in practical adversarial settings.

VIII. MITIGATION STRATEGIES

Zhao et al. [20] proposed several robustness enhancements to their graph watermarking scheme. While effective against random perturbations, these defenses introduce ambiguity in attribution, for example, detecting only 3 out of 5 embedded

watermark subgraphs may lead to uncertain source identification.

In contrast, we propose a lightweight modification to the embedding process that enhances robustness against clustering-aware attacks without requiring changes to the extraction logic or increasing system complexity. Our approach distributes the watermark across structural communities in the graph, improving resilience to targeted perturbations, particularly in larger, more densely connected graphs. This strategy remains fully compatible with existing detection methods and requires no changes to the extraction procedure.

A. Robust Embedding

As outlined in Section III, Zhao et al.'s embedding process consists of four main steps. Our modification targets **Step (III): Selecting the watermark placement in** G. In the original scheme, nodes are chosen based on local structural features via hashed NSDs. While this approach is resilient to node anonymization, it does not guarantee distribution of watermark nodes (i.e., the k existing nodes selected for subgraph modification) across structurally diverse regions of the graph.

In our revised approach, the data owner first applies a clustering algorithm G and selects watermark nodes from across the resulting communities. Using the seed Ω_i, the node set is deterministically shuffled, one node is selected per cluster until k nodes are chosen. If the number of clusters is fewer than k, the remaining nodes are randomly drawn from the rest of the graph. Because the process is seeded with Ω_i, it is fully deterministic and supports reproducible embedding and extraction. Full pseudocode is provided in Appendix B.

The key insight of this strategy is that it creates an intentional mismatch between how watermark nodes are placed and how an attacker identifies structural regions for perturbation. We assume a black-box adversary with no access to the detection mechanism or other watermarked graphs, and thus no opportunity to tune clustering parameters or validate attack effectiveness. In contrast, the data owner can employ a parameterized clustering method (e.g., Leiden with a custom resolution) during embedding. This asymmetry introduces uncertainty for the attacker, where even if they guess the clustering algorithm, they are unlikely to match the embedding parameters exactly. This misalignment makes it harder to isolate or degrade the watermark through structural manipulation.

B. Mitigation Evaluation

We evaluate the effectiveness of our mitigation strategy using the same experimental setup described in Section VI. We measure attribution accuracy to assess leaker identification performance under our cluster-aware threat model, dK-2 deviation to quantify structural distortion caused by the attacks, and runtime to evaluate computational efficiency. All evaluation parameters remain consistent with prior experiments: we use the same graphs, watermarking parameters ($\delta = 0.3$, $p = 0.5$), and evaluation protocol (30 trials over 10 watermarked graphs

with one randomly selected leaker). Edge flip percentages are also matched to the earlier settings for each graph dataset.

To embed the watermark under our modified node selection strategy, the data owner selects a clustering algorithm to partition the graph prior to node selection. For this evaluation, we use the *Leiden* algorithm with a resolution parameter of 1.2. This choice reflects a realistic security assumption where, unlike the attacker, the data owner can choose a parameterized clustering method. This asymmetry introduces an additional layer of defense, as the attacker would need to infer not only the clustering algorithm used during embedding, but also its internal parameterization. For consistency with earlier evaluations, the attacker continues to use the same parameter-free clustering algorithms to guide intra- and inter-cluster perturbations.

Additionally, we compare the runtime and distortion of our embedding and extraction mechanisms to those of Zhao et al. to validate that our strategy introduces no unintended overhead. We measure average runtime over 100 trails for both schemes, evaluate the dK-2 deviation between the original and watermarked graphs, and assess extrication reliability across 100 watermarked and 100 non-watermarked graphs. In all cases, our approach maintains comparable runtime and structural integrity, and achieves 100% accuracy in distinguishing between watermarked and non-watermarked graphs.

Attribution Accuracy. Our mitigation strategy demonstrates improved attribution robustness across both datasets when compared to the baseline embedding scheme in Figure 3. In *LastFM*, we observe that our method may exhibit an earlier decline in detection accuracy under certain attack types, particularly intra-add/inter-remove. However, this decline is more gradual, and attribution performance stabilizes at higher perturbation levels, where Zhao et al.'s method continues to degrade. This trend suggests that distributing watermark nodes across clusters improves long-term resilience, as it becomes harder for the attacker to fully suppress the watermark with inducing broader structural distortion.

That said, we note limitations in specific *LastFM* scenarios. Under the intra-remove/inter-add attack using *Greedy Modularity*, our method underperforms relative to the baseline. Additionally, in a few cases, accuracy under our method occasionally falls below that of random perturbation at higher flip budgets. We attribute both effects to the sparse and fragmented nature of the *LastFM* graph, which may cause our embedding to unintentionally align with community structures targeted by the attacker. These results suggest that while our method improves robustness overall, its effectiveness can be sensitive to the underlying graph topology and the specific clustering strategy used for embedding.

In contrast, on the *Facebook* dataset, our approach consistently maintains higher detection accuracy under all attack strategies and clustering algorithms. The graph's higher edge density and stronger community structure allow our cluster-aware embedding to benefit more from structural dispersion, making it more difficult for an adversary to suppress the watermark without significant impact on overall graph connectivity.
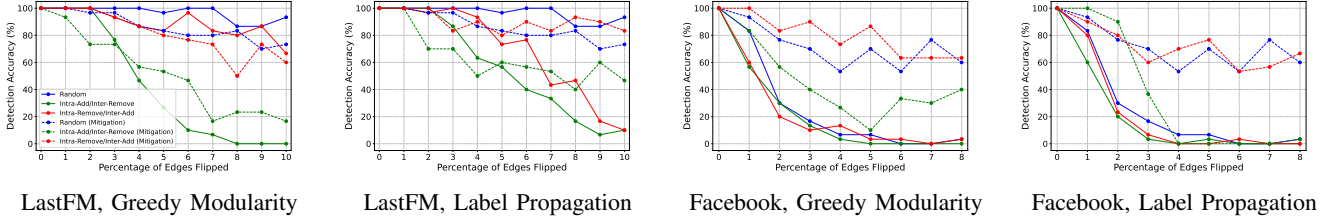
| LastFM, Greedy Modularity | LastFM, Label Propagation | Facebook, Greedy Modularity | Facebook, Label Propagation |

Fig. 3: Attribution accuracy under Zhao et al.'s embedding and our mitigation strategy. Our method improves robustness across datasets and clustering algorithms, maintaining higher detection accuracy under stronger attacks.
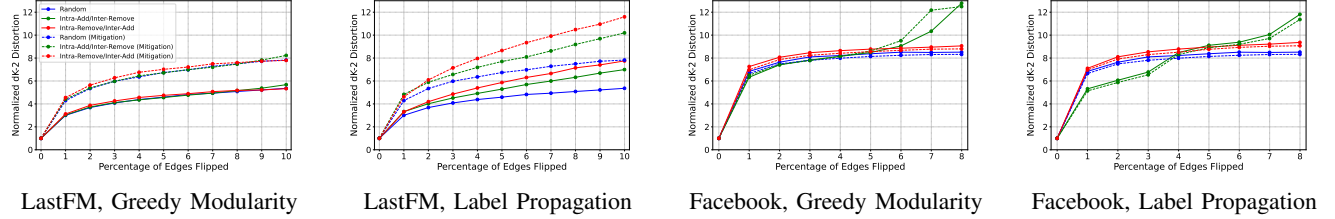

| LastFM, Greedy Modularity | LastFM, Label Propagation | Facebook, Greedy Modularity | Facebook, Label Propagation |

Fig. 4: dK-2 distortion under Zhao et al.'s embedding and our mitigation strategy. Both methods follow similar distortion trends, with our approach achieving better evasion without added structural disruption.

Together, these results indicate that spreading the watermark across communities improves resilience to clustering-aware attacks by forcing the adversary to introduce broader structural changes. Our method maintains higher robustness at larger attack budgets, demonstrating its effectiveness in preserving attribution integrity under increasingly severe adversarial conditions.

dK-2 Distortion. We evaluate structural impact by measuring dK-2 deviation between the original and perturbed watermarked graphs. As expected, all attack strategies exhibit increasing distortion with larger perturbation budgets, and our embedding strategy follows the same overall distortion trajectory as Zhao et al.'s baseline across both datasets. Our results are described in Figure 4.

The only notable difference occurs in the *LastFM* dataset, where our method introduces slightly higher distortion at lower perturbation levels (1–2% edge flips). This early increase aligns with the sharper initial drop in attribution accuracy observed under specific attack strategies. Since our embedding spreads watermark nodes across structural communities, even a small number of edge modifications may affect the integrity of the dispersed subgraph more significantly in sparse graphs. Beyond this early stage, distortion remains comparable between the two methods, and both converge toward similar distortion trajectories as the perturbation budget increases.

Runtime. The runtime required to perform attacks remains unchanged from the evaluation in Section VI, as our mitigation strategy does not modify the attack configuration or detection process. Full attack runtime results are included in Appendix C.

We also evaluate computational efficiency by comparing watermark embedding and extraction runtimes between our method and Zhao et al.'s approach. As shown in Table V, our

approach reduces both embedding and extraction time across datasets. This improvement is due to the streamlined node selection process that eliminates the overhead associated with NSD hashing and collision handling in the original scheme.

Method	Zhao et al.		Ours	
	Facebook	LastFM	Facebook	LastFM
Embedding	0.547	0.273	0.446	0.206
Extraction	0.211	0.099	0.111	0.054

TABLE V: Runtime comparison (seconds) between Zhao et al. and our method for watermark embedding and extraction.

Initial Distortion. To evaluate the structural impact of watermark embedding, we measure the dK-2 deviation between the original graph and the watermarked graph under both our method and Zhao et al.'s scheme. Since both approaches embed a single watermark of size k, the resulting structural deviation remains comparable across methods. As shown in Table VI, these results confirm that our embedding strategy preserves the statistical properties of the host graph while offering enhanced resilience against clustering-aware attacks.

Graph	Zhao et al.	Ours
Facebook	0.000006	0.000006
LastFM	0.000027	0.000025

TABLE VI: dK-2 distortion introduced by watermark embedding under Zhao et al. and our methods.

IX. CONCLUSION

Graph data is widely used across domains where maintaining attribution is critical. Graph watermarking enables this by embedding identifiable signatures into shared graph structures,

but it remains an understudied area, particularly under cluster-aware threat models, which have not been previously explored. In this work, we demonstrate that adversaries who exploit the inherent community structure of real-world graphs can degrade detection performance more effectively than through random perturbations. To address this, we propose a lightweight and easily integrable modification to existing watermark embedding schemes that improves robustness without increasing system complexity. Our findings emphasize the need to account for topological awareness in both watermark design and adversarial modeling. As watermarking schemes evolve, so too will attacker strategies. Future work should explore a broader range of graph types, adversarial capabilities, and embedding techniques, particularly under mismatched clustering assumptions between data owners and attackers. We discuss these directions further in Appendix A.

REFERENCES

[1] C. Wilson, A. Sala, K. P. Puttaswamy, and B. Y. Zhao, "Beyond social graphs: User interactions in online social networks and their implications," *ACM Transactions on the Web (TWEB)*, vol. 6, no. 4, pp. 1–31, 2012.

[2] J. Scott, "Social network analysis: developments, advances, and prospects," *Social network analysis and mining*, vol. 1, pp. 21–26, 2011.

[3] M. M. Li, K. Huang, and M. Zitnik, "Graph representation learning in biomedicine and healthcare," *Nature Biomedical Engineering*, vol. 6, no. 12, pp. 1353–1369, 2022.

[4] P. Amudha, A. C. Sagayaraj, and A. S. Sheela, "An application of graph theory in cryptography," *International Journal of Pure and Applied Mathematics*, vol. 119, no. 13, pp. 375–383, 2018.

[5] J. Song, P. Zhang, Q. Qu, Y. Bai, Y. Gu, and G. Yu, "Why blockchain needs graph: A survey on studies, scenarios, and solutions," *Journal of Parallel and Distributed Computing*, vol. 180, p. 104730, 2023.

[6] N. Nikolaidis and I. Pitas, "Robust image watermarking in the spatial domain," *Signal processing*, vol. 66, no. 3, pp. 385–403, 1998.

[7] A. K. Singh, B. Kumar, G. Singh, and A. Mohan, *Medical image watermarking*. Springer, 2017.

[8] P. Bassia, I. Pitas, and N. Nikolaidis, "Robust audio watermarking in the time domain," *IEEE Transactions on multimedia*, vol. 3, no. 2, pp. 232–241, 2001.

[9] K. Khaldi and A.-O. Boudraa, "Audio watermarking via emd," *IEEE transactions on audio, speech, and language processing*, vol. 21, no. 3, pp. 675–680, 2012.

[10] A. N. Lemma, J. Aprea, W. Oomen, and L. van de Kerkhof, "A temporal domain audio watermarking technique," *IEEE transactions on signal processing*, vol. 51, no. 4, pp. 1088–1097, 2003.

[11] C. S. Collberg, C. Thomborson, and G. M. Townsend, "Dynamic graph-based software fingerprinting," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 29, no. 6, pp. 35–es, 2007.

[12] F. Liu, B. Lu, and X. Luo, "A chaos-based robust software watermarking," in *International Conference on Information Security Practice and Experience*. Springer, 2006, pp. 355–366.

[13] T. Ji, E. Ayday, E. Yilmaz, and P. Li, "Robust fingerprinting of genomic databases," *Bioinformatics*, vol. 38, no. Supplement_1, pp. i143–i152, 2022.

[14] T. Ji, E. Ayday, E. Yilmaz, M. Li, and P. Li, "Privacy-preserving database fingerprinting," in *NDSS symposium*, vol. 2023, 2023, pp. 10–14 722.

[15] Z. Ren, H. Fang, J. Zhang, Z. Ma, R. Lin, W. Zhang, and N. Yu, "A robust database watermarking scheme that preserves statistical characteristics," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 6, pp. 2329–2342, 2023.

[16] A. Nemecek, Y. Jiang, and E. Ayday, "Topic-based watermarks for large language models," *arXiv preprint arXiv:2404.02138*, 2024.

[17] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, "Embedding watermarks into deep neural networks," in *Proceedings of the 2017 ACM on international conference on multimedia retrieval*, 2017, pp. 269–277.

[18] S. Lounici, M. Njeh, O. Ermis, M. Önen, and S. Trabelsi, "Yes we can: Watermarking machine learning models beyond classification," in *2021 IEEE 34th Computer Security Foundations Symposium (CSF)*. IEEE, 2021, pp. 1–14.

[19] X. You, Y. Jiang, J. Xu, M. Zhang, and M. Yang, "Gnnguard: A fingerprinting framework for verifying ownerships of graph neural networks," in *The Web Conference 2024*, 2024.

[20] X. Zhao, Q. Liu, H. Zheng, and B. Y. Zhao, "Towards graph watermarks," in *Proceedings of the 2015 ACM on Conference on Online Social Networks*, 2015, pp. 101–112.

[21] D. Eppstein, M. T. Goodrich, J. Lam, N. Mamano, M. Mitzenmacher, and M. Torres, "Models and algorithms for graph watermarking," in *Information Security: 19th International Conference, ISC 2016, Honolulu, HI, USA, September 3-6, 2016. Proceedings 19*. Springer, 2016, pp. 283–301.

[22] J. G. Bourrée, A.-M. Kermarrec, E. L. Merrer, and O. Safsafi, "Fast in-spectrum graph watermarks," *arXiv preprint arXiv:2502.04182*, 2025.

[23] H. Cherifi, G. Palla, B. K. Szymanski, and X. Lu, "On community structure in complex networks: challenges and opportunities," *Applied Network Science*, vol. 4, no. 1, pp. 1–35, 2019.

[24] M. E. Newman, "Modularity and community structure in networks," *Proceedings of the national academy of sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[25] A. Narayanan and V. Shmatikov, "How to break anonymity of the netflix prize dataset," *arXiv preprint cs/0610105*, 2006.

[26] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song, "Adversarial attack on graph structured data," in *International conference on machine learning*. PMLR, 2018, pp. 1115–1124.

[27] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 14, 2001.

[28] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[29] V. A. Traag, L. Waltman, and N. J. Van Eck, "From louvain to leiden: guaranteeing well-connected communities," *Scientific reports*, vol. 9, no. 1, pp. 1–12, 2019.

[30] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," *ProQuest number: information to all users*, 2002.

[31] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, Jun. 2014.

[32] J. Leskovec and J. Mcauley, "Learning to discover social circles in ego networks," *Advances in neural information processing systems*, vol. 25, 2012.

[33] B. Rozemberczki and R. Sarkar, "Characteristic Functions on Graphs: Birds of a Feather, from Statistical Descriptors to Parametric Models," in *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*. ACM, 2020, p. 1325–1334.

[34] A. Sala, L. Cao, C. Wilson, R. Zablit, H. Zheng, and B. Y. Zhao, "Measurement-calibrated graph models for social network experiments," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 861–870.

A. Discussion

Graph Types. Our evaluation demonstrates that the effectiveness of our cluster-aware threat model varies across graphs with differing structural properties, such as sparsity versus density. While we anticipate similar trends in detection accuracy across larger graphs with comparable properties, empirical validation is needed to fully assess the scalability of our threat model (e.g., graphs with tens of millions of nodes and edges). We expect our detection strategy to maintain its effectiveness in larger graphs. However, the average distortion per flip percentage will likely increase, as the absolute number of edge modifications grows proportionally with graph size. For our mitigation strategy, we similarly expect detection performance to remain robust, but the extraction process may incur higher computational overhead due to the expanded search space.

Adversarial Capabilities. Our evaluation is based on the single-watermark setting to isolate the effectiveness of cluster-aware attacks. However, extending the analysis to include colluding adversaries would strengthen the evaluation. In scenarios where multiple recipients leak anonymized watermarked graphs, colluding attackers can attempt to compare their graphs to identify watermark regions. As shown in Zhao et al., successful collusion requires deanonymization before meaningful comparison can occur. Their results indicate that while extracting a single watermark is feasible for a colluding attacker, the success rate diminishes significantly when multiple watermark subgraphs are embedded (e.g., 2 to 5 subgraphs). In contrast, our similarity-based detection avoids explicit extraction and is less vulnerable to such comparisons, unless the attacker has access to both the detection pipeline and multiple watermarked graphs. In that case, localized attacks would still be possible. Understanding how our mitigation strategy holds under collusion remains an important direction for future work, particularly when watermark nodes are distributed across communities.

With respect to our mitigation strategy, we introduce an asymmetry between the data owner's and attacker's clustering assumptions. Specifically, the data owner selects a clustering method (e.g., *Leiden*) that incorporates tunable parameters. This contrasts with the attacker, who relies on parameter-free methods such as *Greedy Modularity* or *Label Propagation*. The inclusion of clustering parameters creates additional uncertainty for the attacker. For example, if the attacker chooses among three possible algorithms, the probability of matching the data owner's method drops to one-third. Moreover, methods like *Leiden* introduce resolution parameters, adding another layer of complexity. Even if the attacker selects the correct algorithm, they must also guess the right parameter setting. In future work, we aim to explore this further by evaluating detection accuracy under varying attacker–defender clustering alignments, to better understand how clustering mismatch affects both attack success and defense robustness.

Embedding Strategies. Our embedding method, described in Appendix B, empirically provides effective mitigation against cluster-aware attacks. As summarized in Table I, two other notable schemes have been proposed in prior literature. We aim to incorporate these approaches into a comprehensive evaluation framework to assess whether they exhibit similar vulnerabilities under our threat model. In particular, additional embedding strategies could explore hierarchical group-based watermarking or adaptive node placement that reacts to graph topology metrics such as betweenness or centrality. Since prior schemes do not explicitly consider cluster-aware adversaries, we hypothesize that their robustness would degrade similarly compared to random edge flipping, and that our mitigation strategy would generalize to improve resilience across these methods.

B. Robust Node Selection for Watermark Embedding

As described in Section VIII, our mitigation strategy modifies Step (III) of the original watermark embedding process by Zhao et al. [20], replacing NSD-based node selection with a community-aware approach. The following pseudocode outlines our deterministic method for selecting k watermark nodes across clusters. This process ensures that nodes are sampled from different communities identified via a clustering algorithm (e.g., Leiden), increasing the structural dispersion of the watermark and improving its robustness under clustering-aware attacks.

Algorithm 3 Select Nodes Across Clusters

1: **Input:** Graph $G = (V, E)$, randomness seed Ω_i, number of nodes k, clustering map $\mathcal{C}$
2: Set random seed using Ω_i
3: Initialize mapping: cluster → list of nodes
4: **for** each node v in G **do**
5:      Let $c \leftarrow \mathcal{C}(v)$
6:      Add v to cluster_to_nodes[c]
7: **end for**
8: Initialize empty list: selected_nodes
9: Let $\mathcal{K} \leftarrow$ list of all cluster IDs in random order
10: **for** each cluster c in $\mathcal{K}$ **do**
11:      **if** length of selected_nodes $\geq k$ **then**
12:          **break**
13:      **end if**
14:      Let candidates $\leftarrow$ sorted nodes in cluster c
15:      Let $i \leftarrow \Omega_i$ mod |candidates|
16:      Add candidates[i] to selected_nodes
17: **end for**
18: **if** length of selected_nodes $< k$ **then**
19:      Let remaining $\leftarrow$ nodes in G not in selected_nodes
20:      Shuffle remaining
21:      Add first $k - $ |selected_nodes| nodes from remaining to selected_nodes
22: **end if**
23: **return** selected_nodes

C. Attack Runtime Mitigation Evaluation

Although our mitigation strategy modifies the watermark embedding step, it does not alter the attack process or detection method. All clustering-aware attacks were executed under the same settings as described in Section VI, using identical edge flip percentages, clustering algorithms, and dK-2 similarity detection. For completeness, we report the runtime measurements of each attack configuration across datasets.

Graph	Random		
	1%	3%	5%
Facebook	0.74	1.55	2.38
LastFM	0.40	0.87	1.34

TABLE VII: Average runtime (seconds) of the random edge-flipping attack on both datasets across different flip percentages for our mitigation technique.

Graph	IA/IR (Label Prop)			IR/IA (Label Prop)		
	1%	3%	5%	1%	3%	5%
Facebook	4.77	14.90	54.83	3.96	11.55	18.80
LastFM	0.68	1.75	2.86	0.56	1.37	2.17

TABLE VIII: Average runtime (seconds) of cluster-aware attacks using *Label Propagation*, across both attack strategies (IA/IR and IR/IA) and flip percentages for our mitigation technique.

Graph	IA/IR (Greedy Mod)			IR/IA (Greedy Mod)		
	1%	3%	5%	1%	3%	5%
Facebook	3.97	11.70	19.69	3.99	11.70	18.86
LastFM	0.56	1.45	2.46	0.55	1.36	2.17

TABLE IX: Average runtime (seconds) of cluster-aware attacks using *Greedy Modularity*, across both attack strategies (IA/IR and IR/IA) and flip percentages for our mitigation technique.

As expected, the runtime remains consistent with prior evaluations, confirming that our embedding changes do not introduce any additional attack-time overhead.