Dandelion: Redesigning the Bitcoin Network for Anonymity

Shaileshh Bojja
Venkatakrishnan
University of Illinois at
Urbana-Champaign
bjjvnkt2@illinois.edu

Giulia Fanti
University of Illinois at
Urbana-Champaign
fanti@illinois.edu

Pramod Viswanath
University of Illinois at
Urbana-Champaign
pramodv@illinois.edu

ABSTRACT

Bitcoin and other cryptocurrencies have surged in popularity over the last decade. Although Bitcoin does not claim to provide anonymity for its users, it enjoys a public perception of being a 'privacy-preserving' financial system. In reality, cryptocurrencies publish users' entire transaction histories in plaintext, albeit under a pseudonym; this is required for transaction validation. Therefore, if a user's pseudonym can be linked to their human identity, the privacy fallout can be significant. Recently, researchers have demonstrated deanonymization attacks that exploit weaknesses in the Bitcoin network's peer-to-peer (P2P) networking protocols. In particular, the P2P network currently forwards content in a structured way that allows observers to deanonymize users. In this work, we redesign the P2P network from first principles with the goal of providing strong, provable anonymity guarantees. We propose a simple networking policy called DANDELION, which achieves nearly-optimal anonymity guarantees at minimal cost to the network's utility. We also provide a practical implementation of DANDELION for deployment.

1. INTRODUCTION

Cryptocurrencies are digital currencies that provide cryptographic verification of transactions. Bitcoin is the best-known example of a cryptocurrency [34]. In recent years, cryptocurrencies have transitioned from an academic research topic to a multi-billion dollar industry [13].

Cryptocurrencies exhibit two key properties: egalitarianism and transparency. In this context, *egalitarianism* means that no single party wields disproportionate power over the network's operation. This diffusion of power is achieved by asking other network nodes (e.g., other Bitcoin users) to validate transactions, instead of the traditional method of using a centralized authority for this purpose. Moreover, all transactions and communications are managed over a fully-distributed, peer-to-peer (P2P) network.

Cryptocurrencies are *transparent* in the sense that all transactions are verified and recorded with cryptographic integrity guarantees; this prevents fraudulent activity like double-spending of money. Transparency is achieved through a combination of clever cryptographic protocols and the publication of transactions in a ledger known as a *blockchain*. This blockchain serves as a public record of every financial transaction in the network.

A property that Bitcoin does *not* provide is anonymity. Each user is identified in the network by a public, cryptographic key. If one were to link such a key to its owner's human identity, the owner's entire financial history could be learned from the public blockchain. In practice, it is possible to link public keys to identities through a number of channels, including the very networking protocols on which Bitcoin is built [7]. This is a massive privacy violation, and can be downright dangerous for deanonymized users.

Bitcoin is often associated with anonymity or privacy in the public eye, despite explicit statements to the contrary in the original Bitcoin paper [34]. People may therefore use Bitcoin without considering the potential privacy repercussions [2]. Moreover, this problem is not unique to Bitcoin; many spinoff cryptocurrencies (known as *altcoins*) use similar technologies, and therefore suffer from the same lack of anonymity in their P2P networks.

The objective of this paper is to redesign the Bitcoin networking stack from *first principles* to *prevent network-facilitated deanonymization* of users. Critically, this redesign must not reduce the network's reliability or performance. Although the networking stack is only one avenue for deanonymization attacks, it is an avenue that is powerful, poorly-understood, and often-ignored. To better convey the problem, we begin with a brief primer on Bitcoin and its networking stack.

1.1 Bitcoin Primer

Bitcoin represents each user and each unit of Bitcoin currency by a public-private key pair. A user "possesses" a coin by knowing its private key. Any time a user Alice wishes to transfer her coin m to Bob, she generates a signed *transaction* message, which states that Alice (denoted by her public key) transmitted m (denoted by its public key) to Bob (denoted by his public key). This transaction message is broadcast to all active Bitcoin nodes, at which point *miners*, or nodes who choose to help validate transactions, race to append the transaction to a global ledger known as the *blockchain*. Specifically, each miner aggregates a group of transaction messages into a *block*, or list, and then completes a computational proof-of-work for the block; the first miner to complete a proof-of-work appends their block to

the blockchain and reaps a reward of newly-minted bitcoins and transaction fees.

1.1.1 Bitcoin message propagation

This paper focuses on one key step in the pipeline: broadcasting transactions to other nodes. The broadcasting process is critical because it affects which nodes can reap a transaction's mining reward (by virtue of the delivery delays to different nodes), and it also affects the global consistency of the network (e.g., if only a subset of the users receive a given transaction).

To understand the mechanics of broadcasting, note that cryptocurrencies can be abstracted into two layers: the application layer and the network layer. The application layer handles tasks like transaction management, blockchain processing, and mining. Nodes are identified by their public keys in the application layer. The network layer handles communication between nodes, which occurs over a P2P network of inter-node TCP connections. In the network layer, nodes are identified by their IP addresses. As we shall see momentarily, a node's IP address and public key should remain unlinkable for privacy reasons.

Bitcoin's peer-to-peer broadcast of transactions and blocks is based on flooding information along links in the P2P network. When a node learns of a new transaction or block, it passes the message to its neighbors who have not yet seen the message with an independent, exponential delay. The process continues recursively until all reachable peers receive the message. This broadcast protocol is commonly known as a *diffusion process*; it forms the basis of Bitcoin's global, eventually consistent log and is therefore of utmost importance to its correct and fair operation.

1.1.2 Desirable Network Properties

Bitcoin's network layer should exhibit two principal properties: low latency and anonymity.

Low latency means that the maximum time for a message to reach all network nodes should be bounded and small. Latency matters because if the network fails to deliver messages within a predictable time bound, the network risks reaching an inconsistent state.

Anonymity means that the adversary should be unable to link transaction messages (and hence, the associated public keys) to the IP address that originated a transaction. Recall that every transaction made by a public key is listed in plaintext in the blockchain. Therefore, if a public key can be linked to an IP address, the adversary can link all of that user's transactions. In some cases, the IP address could even be used to learn a node operator's human identity. Thus, deanonymization attacks can result in a user's entire banking history being revealed. Cryptocurrency users are typically recommended to choose fresh public keys and "mix" their coins with others to obscure their transaction history [43, 11] (in practice, few users do so [27, 31]). However, these techniques are useless if the IP address of the source of the transaction can be recovered.

1.1.3 How the Current Network Fails

In recent years, security researchers have demonstrated multiple deanonymization attacks on the Bitcoin P2P network. These attacks typically use a "supernode" that connects to active Bitcoin nodes and listens to the transaction traffic relayed by honest nodes [28, 7, 8]. Because nodes diffuse transactions symmetrically over the network, researchers were able to link Bitcoin users' public keys to their IP addresses with an accuracy of up to 30% [7]. Moreover, the source estimators used in these papers are simple, and exploit only minimal knowledge of the P2P graph structure and the structured randomness of diffusion. We hypothesize that even higher accuracies may be possible with more sophisticated estimation tools.

These attacks demonstrate that Bitcoin's networking stack is inadequate for protecting users' anonymity. Moreover, the Bitcoin networking codebase is copied almost directly in other cryptocurrencies, so the problem pervades the ecosystem. To some extent, this is to be expected: Bitcoin's networking stack was taken directly from Satoshi Nakamoto's original network implementation, which appears to have been a product of expediency. However, in the decade since Bitcoin's release, the networking stack and its anonymity properties have not been systematically studied.

1.2 Problem Statement and Contributions

We aim to address the Bitcoin P2P network's poor anonymity properties through a ground-up redesign of the networking stack. We seek a network management policy that exhibits two properties: (a) strong anonymity against an adversarial group of colluding nodes (which are a fraction p of the total network size), and (b) low broadcasting latency. We define these notions formally in Section 2. Critically, these networking protocols should be *lightweight* and provide *statistical anonymity guarantees against computationally-unbounded adversaries*. Lightweight statistical solutions are complementary to cryptographic solutions, which aim to provide worst-case anonymity guarantees, usually in the face of computationally bounded adversaries. Lightweight anonymization methods lower the barrier to adoption since a more efficient, faster protocol leads to a better user experience and also places less burden on developers to significantly modify existing code; their study is also of basic scientific and engineering interest. Such is the goal of this paper.

Part of the novelty of our work is that the Bitcoin P2P networking stack has not been modeled in any detailed way (much less analyzed theoretically), to the best of our knowledge. In addition to modeling this complex, real-world networking system, our contributions are threefold:

(1) Fundamental anonymity bounds. The act of user deanonymization can be thought of as classifying transactions to source nodes. Precision and recall are natural performance metrics. Recall is simply the probability of detection, a common anonymity metric that captures completeness of the estimator, whereas precision captures the exactness. We define these terms precisely in Section 2.4.

Given a networking protocol, the adversary has a region of feasible (recall, precision) operating points, which are achieved by varying the source classification algorithm. We give fundamental bounds on the best precision and recall achieved by the adversary for any networking protocol, as illustrated in Figure 1; here p refers to the ratio of colluding nodes to the total number of nodes in the network. We show that a (recall, precision) point is feasible only if it lies between the red and blue lines in Figure 1. Moreover, every networking protocol yields an achievable (recall, precision) region to the adversary that intersects with the shaded region (a) in Figure 1 in at least one point.

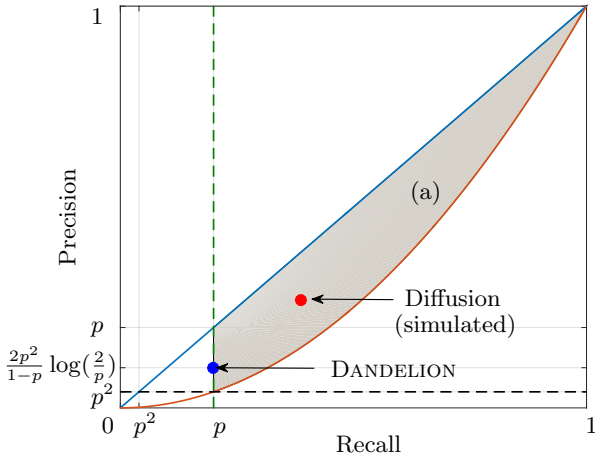**(2) Optimal algorithm.** We propose a simple network-

Figure 1: Bounds on the precision and recall of any networking protocol, plotted for $p = 0.2$. DANDELION has strong anonymity properties, achieving a precision-recall region close to the fundamental lower bounds.

ing protocol called DANDELION, whose achievable precision-recall region is nearly optimal, in the sense that it is contained in the achievable region of (nearly) every other possible networking protocol.

DANDELION consists of two phases. In the first phase, each transaction is propagated on a random line; that is, each relay passes the message to exactly one (random) node for a random number of hops. In the second phase, the message is broadcast as fast as possible using diffusion. DANDELION has two key constraints: (a) in the first phase, all transactions from all sources should propagate over the *same* line, and (b) the adversary should not be able to learn the structure of the line beyond the adversarial nodes' immediate neighbors.

The point labeled 'DANDELION' in Figure 1 is the Pareto frontier of DANDELION's achievable precision-recall region. We compare this to the achievable region for diffusion. The point labeled 'Diffusion' was obtained by simulating a diffusion process on a snapshot of the Bitcoin server network from 2015 [32], and using a suboptimal source classifier. Because of this, the achievable region must contain the plotted point, but may be larger. Not only is the region for diffusion a superset of the one for DANDELION, but DANDELION's region is nearly as small as possible. We revisit Figure 1 in greater detail in Sections 3 and 5.

(3) Practical implementation. We outline the practical challenges associated with implementing DANDELION. In particular, constructing the graph for DANDELION in a distributed fashion, and enforcing the assumption that the adversary cannot learn the graph, are nontrivial. We therefore propose simple heuristics for addressing these challenges in practical implementations.

Paper Structure. We begin by discussing Bitcoin's P2P networking stack and our problem of interest, which we model in Section 2. We then present fundamental bounds on our anonymity metric in Section 3; these bounds are used for comparison with various networking policies later in the paper. In Section 4, we present some first-order solutions, and explain why they do not work. We present our main result, DANDELION, in Section 5. Section 6 discusses the systems

challenges of implementing DANDELION, and proposes some simple, heuristic solutions. We discuss the relation between DANDELIONand prior related work in Section 7, and conclude with some open problems in Section 8.

2. SYSTEM MODEL

We model three critical aspects of Bitcoin's P2P network: the network topology, the message propagation protocol, and the deanonymizing adversary's capabilities. These models are based on existing protocols and observed behavior.

2.1 P2P Network Model

The Bitcoin P2P network contains two classes of nodes: servers and clients. Clients are nodes that do not accept incoming TCP connections (e.g., nodes behind NAT), whereas servers do accept incoming connections. We focus in this work on servers because (a) they are more permanent in the network, and (b) it is straightforward to generalize server-oriented anonymity solutions to also protect clients.

We model the P2P network as a graph $G(V, E)$, where V is the set of all server nodes and E is the set of edges, or connections, between them. For a node v, $\Gamma(v)$ denotes the set of v's neighbors in G. Similarly for a set of nodes U, $\Gamma(U)$ denotes the set of all neighborhood sets of the nodes in U. To model the graph's topology, we first discuss Bitcoin's network management protocols.

Each node in the Bitcoin P2P network has an *address manager*—a list of other nodes' contact information represented as a (IP address, port) pair, along with a time estimate of when that node was last active. When a server first joins the network, its address manager is empty, but the node can learn a random set of active addresses by contacting a hard-coded DNS server. During normal network operation, nodes periodically relay entries from their address managers, which helps spread information regarding active peers. We model address managers by assuming that each node possesses the contact information for *every* other Bitcoin server. In practice, address managers actually contain a random sample of population IP addresses.

Each server is allowed to establish up to eight outgoing connections to nodes in the server's address manager. An *outgoing connection* from Alice to Bob is one that is initiated by Alice. However, these TCP connections are bidirectional once established. We therefore model the subgraph of server-to-server connections as a random 16-regular graph. In practice, the degree distribution is not quite uniform—we revisit this issue in Section 6.

2.2 Transaction Model

As explained in Section 2.3, the network is partitioned into honest nodes and colluding, adversarial nodes, who attempt to deanonymize users. In this work, we assume that all honest nodes generate one transaction in the time period of interest. In practice, servers generate transactions at different rates; however, all transactions by a single node are identified by the node's public key (as long as the node does not generate fresh keys). Therefore, we treat multiple transactions from the same node as a single transaction to be deanonymized. We also assume the exact time when each server starts broadcasting its transaction is unknown to the adversary. A typical transaction can take up to 60 seconds to propagate through the Bitcoin network [16], so estimating its time of origin at a useful granularity of a second or

sub-second can be difficult. $\mathcal{X}$ is the set of all transaction messages from honest servers. X_v is the transaction message originating from honest server v and $\mathbf{X}$ is a vector containing the ground truth mapping between source nodes v and transactions X_v. We model the mapping between servers and transaction messages as being drawn uniformly from the set of all such mappings.

Spreading Model. Once a Bitcoin transaction is complete, the source broadcasts the transaction message over the network. The protocol for broadcasting transactions should ensure low transaction latency, in order to provide network consistency and fairness.

Bitcoin currently uses a *diffusion* propagation mechanism to broadcast transactions, in which each transaction source or relay passes the transaction to the node's neighbors with independent, exponential delays. Once a node has received a particular transaction, the node does not accept future relays of the transaction. This diffusion spreading serves as a baseline for our algorithmic improvements. It has good latency properties due to its exponential spreading [5].

More generally, in this work, we consider spreading policies that are symmetric in the neighbor node IDs; that is, a forwarding node does not use the IP address values (or other metadata) of its neighbors to influence its forwarding decisions. This holds for diffusion spreading, but we constrain our proposed solutions to also satisfy the same property.

2.3 Adversarial Model

We consider an adversary whose goal is to deanonymize users by linking their transactions (and hence, their public keys) to their IP addresses. In particular, we are interested in defending against *botnets*—large sets of malware-infected hosts that are controlled remotely, often without the host owners' knowledge [47]. Botnets are a commonly-studied adversarial model for various Bitcoin attacks [3], largely because they are easy to access, cheap, and pervasive in the Bitcoin network [37]. While botnets can have many uses, we wish to defend against a botnet that aims to deanonymize users.

We model the botnet adversary as a set of adversarial, colluding "spy" nodes that participate in the Bitcoin network as if they were honest nodes (i.e., honest-but-curious). We denote honest nodes by V_H and adversarial nodes by V_A. For a parameter p, we assume a fixed number of adversarial nodes ($|V_A| = np$) and honest nodes ($\tilde{n} = |V_H| = (1-p)n$). The adversarial nodes are dispersed uniformly at random in the network; this reflects the botnet's ability to obtain IP addresses uniformly across the IP address space. However, for a given topology, the actual locations of the honest/adversarial nodes are random. We further assume that all nodes know the complete list of active IP addresses, and honest nodes cannot distinguish between an adversarial and honest IP address.

Whenever a transaction is broadcast over the network, the adversarial nodes log the timestamps and the honest neighbors from which they receive the transaction. We assume a continuous-time system, in which simultaneous transmissions do not occur. For each honest server node v, we let S_v denote the set of (transaction, receiving spy node, timestamp) tuples $(x, u, T_u(x))$ such that transaction x was forwarded by honest node v to adversary $u \in V_A$ at time $T_u(x)$ (Fig. 9); $\mathbf{S}$ is the vector of all S_v's. We shall see in Section 5 that the honest server who *first* delivers a given transaction

$$|V_H| = 7$$
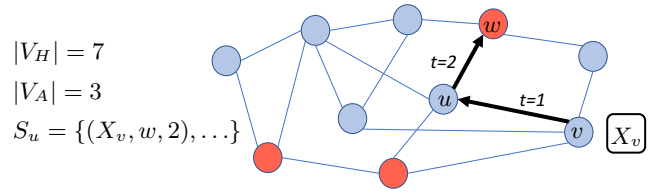$$|V_A| = 3$$
$$S_u = \{(X_v, w, 2), \ldots\}$$

Figure 2: Red nodes are adversarial spies; blue nodes are honest. Message X_v reaches the spy w at time $t = 2$.

to the adversary plays a special role.

In addition to the transaction timestamps, the adversaries can also learn the network structure G, partially or completely, over time. The extent of such knowledge depends on the dynamism of the network, and will be made clear in the context of the specific networks being considered. For example, if the network is static over an extended period of time then adversaries can learn the entire graph G. On the other hand, in a fast changing network, the adversaries have knowledge of only their local neighborhood $\Gamma(V_A)$. For ease of exposition, let us, for now, use $\boldsymbol{\Gamma}$ to denote the adversary's knowledge of the graph.

Once the timestamps have been collected, the adversarial nodes collude to infer the transaction source. The adversary uses its observations $\mathbf{O} = (\mathbf{S}, \boldsymbol{\Gamma})$ to output a mapping between transactions and honest servers; we let $\mathbb{M}(X_v) \in V_H$ denote the server associated with transaction X_v in the adversary's mapping. This mapping is chosen to maximize the adversary's deanonymization payoff, defined in Section 2.4.

2.4 Anonymity Metric

A common metric for measuring a broadcasting scheme's anonymity is **probability of detection**. For a fixed transaction and estimator, probability of detection is defined as

$$\mathbb{P}_{\mathbb{M},G}(\text{detection}) = \frac{\sum_{v \in V_H} \mathbb{P}(\mathbb{M}(X_v) = v)}{\tilde{n}}, \qquad (1)$$

or the probability that the estimator outputs the correct source of a single transaction, computed over all transaction sources $v \in V_H$, mappings between sources and transactions $\mathbf{X}$, realizations of the message propagation trajectory, and graph realizations G (if the graph is random). While probability of detection considers a single source, our problem considers the joint deanonymization of transactions from distinct sources. In this case, probability of detection inherently captures the recall, or completeness, of an estimator. We propose to augment this metric by also studying precision, which captures the exactness of an estimator.

Precision and recall are performance metrics commonly used in information retrieval for binary classification. Suppose we have n data items, each associated with a class: 0 or 1. We are given a classifier that labels each data item as either a 0 or a 1, without access to the ground truth. We designate one of these classes (e.g. class 1) 'positive'. For a given classifier output on a single item, a *true positive* means the item was correctly assigned to class 1, and a *true negative* means the item was correctly assigned to class 0. A *false positive* means a 0 item was incorrectly classified as a 1, and a *false negative* means a 1 item was incorrectly classified as a 0. If we run this classifier on all n data items,

4

precision and recall are defined as follows:

$$\textbf{Precision} \quad = \quad \frac{|\text{True Positives}|}{|\text{True Positives}| + |\text{False Positives}|}$$

$$\textbf{Recall} \quad = \quad \frac{|\text{True Positives}|}{|\text{True Positives}| + |\text{False Negatives}|}$$

where $|\cdot|$ denotes the cardinality of a set, and 'True Positives' denotes the set of all data items whose classification output was a true positive (and so forth).

Precision can be interpreted as the probability that a randomly-selected item with label 1 is correct, whereas recall can be interpreted as the probability that a randomly-selected data item from class 1 is correctly classified. Adapting this terminology to our problem, we have a multiclass classification problem; each server is a class, and each transaction is to be classified. For a given server v and mapping M, the precision $D_\text{M}(v)$ comparing class v to all other classes is computed as[1]

$$D_\text{M}(v) = \frac{\mathbb{1}\{\text{M}(X_v) = v\}}{\sum_{w \in V_H} \mathbb{1}\{\text{M}(X_w) = v\}}, \quad (2)$$

and the recall is computed as

$$R_\text{M}(v) = \mathbb{1}\{\text{M}(X_v) = v\} \quad (3)$$

where $\mathbb{1}\{\cdot\}$ denotes the indicator function. In multiclass classification settings, precision and recall are often aggregated through *macro-averaging*, which consists of averaging precision/recall across classes. This approach is typically used when the number of items in each class is equal [44], as in our problem. We therefore average the precision and recall over all servers and take expectation, giving an expected macro-averaged precision of $\mathbb{E}[D_\text{M}] = \frac{1}{\tilde{n}} \sum_{v \in V_H} \mathbb{E}[D_\text{M}(v)]$ and recall of $\mathbb{E}[R_\text{M}] = \frac{1}{\tilde{n}} \sum_{v \in V_H} \mathbb{E}[R_\text{M}(v)]$.

We now explain why probability of detection does not capture the distinction between precision and recall. First, note that the expected per-node recall is identical to the probability of detection. Now consider two estimators: in the first, the adversary's strategy is to assign all $\tilde{n}$ transactions to one randomly-selected server v. In the second, the adversary creates a random matching between the $\tilde{n}$ transactions and honest servers. Both estimators have a probability of detection (i.e., expected per-node recall) of $1/\tilde{n}$. However, the first estimator has an expected per-node precision of $1/\tilde{n}^2$, while the second has an expected per-node precision of $1/\tilde{n}$. Operationally, this can be interpreted as a difference in plausible deniability: the implicated node v in the first case can deny being the source of any given transaction, because it could not have generated all $\tilde{n}$ transactions. If a node is correctly implicated in the second estimator, it has no plausible deniability. Probability of detection alone does not capture this difference, and is therefore insufficient as a standalone metric.

In this work, we quantify anonymity through a combination of **expected macro-averaged precision** (or "precision" for short) and **expected macro-averaged recall** (or "recall", or probability of detection). Higher precision and recall favor the adversary. For a mapping strategy M let D_M and R_M denote the average precision and recall, respectively, obtained in a realization. Our metrics of interest,

then, are the overall expected precision $\textbf{D}_\text{M} = \mathbb{E}[D_\text{M}]$ and recall $\textbf{R}_\text{M} = \mathbb{E}[R_\text{M}]$. This expectation is taken over four random variables: the graph realization G (which can be random in general), the mapping between servers and messages $\textbf{X}$, the observed timestamp and topological information $\textbf{O}$, and the adversary's mapping strategy M. Similarly let $D_\text{M}(v)$ and $\textbf{D}_\text{M}(v)$ denote the instantaneous and expected precisions at a server $v \in V_H$, and let $R_\text{M}(v)$ and $\textbf{R}_\text{M}(v)$ denote the instantaneous and expected recalls. Let $\textbf{D}_\text{OPT}$ and $\textbf{R}_\text{OPT}$ denote the precision and recall, respectively, of the precision-maximizing and recall-maximizing mapping strategies, respectively. The optimal precision is not necessarily achieved by the same mapping strategy as the optimal recall. The adversary is computationally unbounded.

2.5 Problem Statement

As network designers, we control two aspects of the network: the graph creation/maintenance strategy and the spreading protocol. Our goal is to choose a graph-selection strategy and a spreading protocol that simultaneously give low average latency, precision, and recall guarantees. We restrict ourselves to the following model of graph generation: For a fixed topology τ, we assume that the nodes are equally likely to assume each possible label ordering in τ. Moving forward, $G(V, E)$ will describe the resulting, labeled graph.

Let $\mathcal{T}$ denote the set of all graph topologies over n nodes, and Σ the set of graph-independent spreading strategies. The adversary controls only the estimation algorithm for mapping transactions to nodes. Given a topology $\tau \in \mathcal{T}$ and a spreading strategy $\sigma \in \Sigma$, let $\mathcal{M}_{\tau, \sigma}$ denote the set of mapping strategies that map $\tilde{n}$ transactions to $\tilde{n}$ servers, with all knowledge derived from the topology and the spreading strategy. If τ and σ are clear from context, we simply use $\mathcal{M}$ to denote the space of mapping strategies. We define the *detection region* for τ and σ as the set of achievable precision and recall operating points:

$$\Omega(\tau, \sigma) = \{(D, R) \mid \exists\ \text{M} \in \mathcal{M}_{\tau, \sigma},\ D = \textbf{D}_\text{M}, R = \textbf{R}_\text{M}\}.$$

Note that the detection region always contains the origin. The adversary's goal is to find estimators that achieve the boundary points of the region, whereas our goal is to make the detection region as small as possible.

Problem: Characterize fundamental, protocol-independent bounds on the detection region. Further, identify a (τ^*, σ^*) pair whose detection region is a subset of the detection region of every graph-generation and spreading strategy:

$$\Omega(\tau^*, \sigma^*) = \bigcap_{\sigma \in \Sigma, \tau \in \mathcal{T}} \Omega(\tau, \sigma). \quad (4)$$

It is unclear a priori if such a strategy pair exists. In this work, we show a simple networking policy that closely approximates condition (4).

3. ANONYMITY METRIC PROPERTIES

Precision and recall are not generally used as anonymity metrics, since most anonymity systems provide per-user anonymity guarantees [17, 20, 41, 18]. We instead want guarantees against a stronger adversary that jointly deanonymizes multiple users. The goal of this section is to give intuition about precision and recall as metrics, and to provide fundamental bounds on both.

Our problem differs from traditional classification in that there is only one data item (transaction) per class (server).

---

[1]Following convention we define $D_\text{M}(v) = 0$ if both the numerator and denominator are 0 in Equation (2).

This restricts the set of achievable macro-averaged precision-recall points in a somewhat unconventional way. We first explain how precision and recall are typically used, and then prove fundamental bounds that illustrate the ways in which our problem differs from traditional classification problems.

Precision-Recall Curves. Most binary classifiers have an internal parameter (e.g., a threshold) that can be varied to give the classifier different precision and recall characteristics. Sweeping this parameter typically yields a tradeoff between precision and recall. While this tradeoff has been studied theoretically [38], it is most often illustrated empirically for a given classifier, through curves like Figure 3 (right). Notably, a classifier can achieve high recall (≈ 1) at the expense of precision by assigning all data elements to the positive category, or high precision (≈ 1) at the expense of recall by classifying only data elements that are clearly true positives. Hence the precision-recall points $(0, 1)$ and $(1, 0)$ are typically achievable in practice.

Unlike traditional precision-recall curves, we are not interested in the curve for a single estimator; we want to identify the achievable detection region across *all* estimators. Moreover, since ours is a multi-class classification problem, we consider *macro-averaged* precision and recall. With macro-averaging, increasing the recall (resp. precision) for one class will often reduce the recall (resp. precision) for another. Therefore, it is unclear what the precision-recall tradeoff will look like, or even if the boundary points $(0, 1)$ and $(1, 0)$ are achievable. The following theorem restricts the set of feasible, macro-averaged precision-recall points for *any* estimator the adversary employs.

THEOREM 1. *Any mapping policy* $\mathtt{M} \in \mathcal{M}_{\tau,\sigma}$ *on a network with topology* $\tau \in \mathcal{T}$ *and spreading strategy* $\sigma \in \Sigma$ *has a precision and recall that are bounded as*

$$\mathbf{D}_{\mathtt{M}} \overset{(a)}{\leq} \mathbf{R}_{\mathtt{M}} \overset{(b)}{\leq} \sqrt{\mathbf{D}_{\mathtt{M}}}. \tag{5}$$

(Proof in Section A.1.1)

This theorem follows from the definition of macro-averaged precision and recall; it implies that not only are corner points $(0, 1)$ and $(1, 0)$ unachievable, but every estimator's detection region must lie between the blue and red lines in Fig. 3 (left). Given this constraint, a natural question is whether there exist precision and recall points that can always be achieved, regardless of the networking protocol. We demonstrate the existence of such points by analyzing a simple estimator.

Lower Bounds. Computing lower bounds on precision and recall is challenging because the adversary's knowledge can vary depending on the networking policy. However, the so-called *first-spy estimator* (which is used in practical attacks like [7]) relies only on the adversary's knowledge of its local network neighborhood. The adversaries we consider will always have access to this information. The first-spy estimator outputs the first honest node to send a given message to any of the adversarial nodes. We start by showing that the first-spy estimator always achieves a precision and recall of at least p^2 and p, respectively, where p is the fraction of spies. This in turn implies that the maximum precision and recall over all estimators are individually lower-bounded by p^2 and p, respectively.
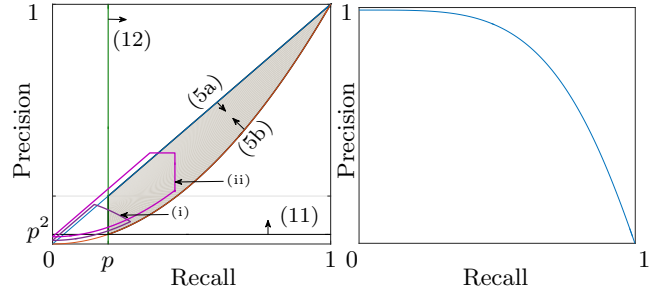

Figure 3: Bounds on the precision-recall detection region for any networking policy (left). Each bound is labeled with the corresponding equation number from Section 3. Example of a typical precision-recall curve (right).

THEOREM 2. *The optimal precision and recall on a network with a fraction p of adversaries and any spreading policy are lower bounded as*

$$\mathbf{D}_{\mathtt{OPT}} \geq p^2 \tag{6}$$
$$\mathbf{R}_{\mathtt{OPT}} \geq p. \tag{7}$$

(Proof in Section A.1.2)

This theorem implies that for any networking policy, the detection region must include at least one point in the shaded region of Figure 3 (left). Note that if an estimator can achieve a given (recall, precision) point, then it can also achieve points with elementwise lower precision and recall by choosing to discard observed information. The purple curves labeled (i) and (ii) outline the boundaries of two examples of feasible detection regions, staggered for visibility.

Optimizing Estimators. Given these constraints on the detection region, we want to understand what estimators achieve the maximum precision and recall, respectively. For a given network specification, precision and recall might be maximized by different estimators; if this is the case, then the detection region will have a nontrivial Pareto frontier, like curve (i) in Figure 3 (left). On the other hand, if the same estimator maximizes precision and recall, the detection region's Pareto frontier will be a single point, like curve (ii).

We start by proving that in order to maximize precision, the adversary should use a maximum-weight matching estimator, where the weights depend on the information observed by the adversary, such as graph structure and timestamps.

THEOREM 3 (PRECISION-OPTIMAL ESTIMATOR). *The precision-optimizing estimator for an adversary with observations* $\mathbf{O} = (\mathbf{S}, \mathbf{\Gamma})$, *is achieved by a matching over the bipartite graph* $(V_H, \mathcal{X})$. *Moreover, such a matching is a maximum-weight matching for edge weights* $\mathbb{P}(X_v = x|\mathbf{O})$ *on each edge* $(v, x) \in V_H \times \mathcal{X}$ *of the graph.*

(Proof in Section A.1.3)

Theorem 3 gives a corollary used in Section 4 for bounding the performance of various networking protocols.

COROLLARY 1. *The optimal expected payoff at a server v, under observations* $\mathbf{O} = (\mathbf{S}, \mathbf{\Gamma})$ *for the adversaries, is upper bounded as*

$$\mathbb{E}[D_{\mathtt{OPT}}(v)|\mathbf{O}] \leq \max_{x \in \mathcal{X}} \mathbb{P}(X_v = x|\mathbf{O}). \tag{8}$$

(Proof in Section A.1.4)

Computing the probabilities in Corollary 1 may be challenging, depending on how much information the adversary has. Nonetheless, if the adversary can approximate these probabilities with some accuracy (e.g., if it knows the underlying graph G), there exist polynomial-time algorithms for computing max-weight matchings [21, 6].

The precision-optimal maximum-weight matching does not necessarily maximize recall. Notice that for any matching, its precision and recall are equal, due to the definitions of precision and recall. The following theorem characterizes a recall-optimal estimator, which assigns each message x to any server v for which $\mathbb{P}(X_v = x | \mathbf{O})$ is maximized.

THEOREM 4 (RECALL-OPTIMAL ESTIMATOR). *The recall-optimizing estimator for an adversary with observations $\mathbf{O} = (\mathbf{S}, \mathbf{\Gamma})$, is a mapping that assigns each transaction $x \in \mathcal{X}$ to any server $v^* \in \arg\max_{v \in V_H} \mathbb{P}(X_v = x | \mathbf{O})$.*

(Proof in Section A.1.5)

The first-spy estimator is an instance of a recall-optimal estimator for spreading models in which the exit node to the first-spy is the most likely source. Moreover, Theorem 4 implies that a precision-optimal, maximum-weight matching is only recall-optimal if it also maps each message to its most likely source, elementwise. For example, if k servers are equally likely sources for k messages, then the precision-optimal matching estimator is also recall-optimal.

Summary. This section provides fundamental limits on both precision and recall, as well as detailing estimators that optimize precision (Thm 3) and recall (Thm. 4), respectively. These fundamental limits and estimators will be useful benchmarks as we analyze the precision-recall regions for networking policies in Sections 4 and 5.

4. BASELINE ALGORITHMS

With the fundamental bounds from Section 3, we now tackle our main problem: designing a networking policy with a minimal detection region. A key message of our work is that statistical anonymity requires *mixing* of messages: users should spread their own messages and those of their peers in a way that is difficult to distinguish. Degree of mixing depends on three key properties of a networking policy: (1) the spreading protocol, (2) the topology of the network, and (3) the dynamicity of the network (i.e., how often the P2P graph changes). For example, the current Bitcoin network uses diffusion spreading over a static, roughly 16-regular topology. This policy has poor mixing—i.e., a large detection region—because different nodes have unique spreading patterns and can therefore be deanonymized.

In this section, we first identify a taxonomy of networking policies, based on the properties above. We then systematically evaluate the anonymity of various first-order, natural networking policies from this taxonomy. We show that most of these baseline policies have poor anonymity guarantees, and we extract rules of thumb for improving a policy's anonymity. These rules of thumb will build the groundwork for our main result, DANDELION, presented in Section 5.

4.1 Taxonomy of Networking Policies

Our taxonomy has three axes: spreading protocol, topology, and dynamicity. We consider multiple categories along each axis.

Spreading protocol. The space of spreading protocols is vast. In this work, we consider a few natural, first-order spreading policies, and also propose a new protocol called dandelion spreading.

Perhaps the most natural spreading strategy is **flooding**, where messages are propagated with a fixed delay to all neighbors. A slightly refined version is **diffusion**, which adds independent randomness to the transmission delays of flooding. Diffusion is explained in Section 2.2. Flooding and diffusion reflect the current status quo in the Bitcoin network.

Given that our goal is to provide anonymity, another natural strategy is to forward a message to a randomly-chosen node, which then runs diffusion or flooding. We call this spreading protocol **diffusion-by-proxy**.

Finally, we propose in this paper a new protocol called **dandelion spreading**. Dandelion spreading forwards each message on a randomly-selected line before diffusing it to the rest of the network. Since dandelion spreading is a comparatively new protocol (not a first-order baseline), we defer a detailed discussion to Section 5.

Topology. We are interested in topologies that are simultaneously simple to construct, analyzable, and good for anonymity. We therefore limit ourselves to a set of canonical graph models: lines, trees, d-regular graphs, and complete graphs. These categories are not mutually exclusive; lines are a special case of both trees and regular graphs (we consider lines and cycles interchangeably), and complete graphs are a special case of regular graphs.

Dynamicity. Many network-based deanonymization attacks use partial or full knowledge of the connectivity graph between nodes [45]. We assume that the network can change the graph at varying rates to control the adversary's ability to learn it. We consider two extremes on this spectrum: static graphs and dynamic graphs. In static graphs, the network never changes the graph, so the adversary learns it fully over time. In dynamic graphs, the graph is changed at a rate such that the adversary only knows its local neighborhood at any given point in time.

In the remainder of this section, we first explore the regions of our taxonomy by studying three baseline networking policies: flooding, diffusion, and diffusion-by-proxy. Although none of these baselines has satisfactory anonymity guarantees, the associated analysis provides valuable intuition that helps us design better policies in Section 5.

4.2 Flooding

To model flooding, we assume that messages propagate along each graph edge with a deterministic delay, and nodes forward incoming messages to their neighbors with a constant delay. On undirected topologies, flooding has poor source-hiding due to symmetry and the deterministic spreading scheme. However, it is unclear if directed topologies fare better. We begin by showing that flooding has poor performance on directed, static, d-regular graphs.

PROPOSITION 1. *The expected precision of flooding on a static d-regular graph is at least $\mathbf{D}_{\texttt{OPT}} \geq (1 - (1 - p)^d) \geq p$.*

(Proof in Section A.2.1)

Flooding performs poorly on static regular graphs because each honest node has a unique spreading "timestamp signature", and the adversary can predict these signatures. That

7

is, if node v is the source, then the adversarial nodes receive all messages from v in a deterministic timing pattern. Moreover, the adversary can *predict* this pattern from the structure of the graph, due to the fixed nature of flooding.

This reasoning suggests that if the adversary does not know the graph, it cannot predict nodes' spreading patterns, and therefore cannot deanonymize nodes. However, the following proposition shows that even when the graph is dynamic, the adversary can achieve a high precision.

PROPOSITION 2. *Flooding precision for dynamic d-regular graphs is bounded as $\mathbf{D}_{\text{OPT}} \geq cp$ for some constant $c > 0$ independent of p.*

(Proof sketch in Section A.2.2)

This result highlights that even if the adversary cannot predict the exact timestamp pattern for a given node, it can infer certain statistical properties of the pattern that are sufficient for deanonymization. In short, as long as the topology allows messages to flood in more than one direction, the adversary can use the statistics of observed timestamp signatures to infer the source of a message.

Lesson: <mark>Do not flood content in multiple directions on the graph at the same rate.</mark>

4.3 Diffusion

Diffusion is a natural successor to flooding; instead of using deterministic delays, it uses random ones. By introducing uncertainty into the adversary's timing estimates, diffusion reduces the adversary's overall precision and recall. However, much research in recent years has shown that the source of a diffusion process can nonetheless be identified reliably [45, 46, 19, 48, 30, 39, 36, 51]. Although there are no theoretical results on the precision or recall under our particular adversarial model, several heuristic estimation algorithms are able to identify the source of a diffusion process on many classes of graphs [36, 51]. Moreover, theoretical results exist on other adversarial models [45, 46, 48]. All of these results rely on the intuition that diffusion spreads content symmetrically. Because of this, the source node appears at the center of the adversary's observed spreading pattern, and can be identified. Diffusion is therefore not a satisfactory solution to this problem.

Lesson: <mark>Random forwarding delays are not powerful enough to provide anonymity against spreading protocols that spread content symmetrically.</mark>

4.4 Diffusion-by-Proxy

The takeaway message from diffusion and flooding is that symmetry of spreading leads to deanonymization. To counter this, we must break the symmetry of diffusion. A natural strategy for breaking symmetry about the source is to ask someone else to spread the message. That is, for every transaction, the source node chooses a peer uniformly at random from the pool of all nodes. It transmits the message to that node, who then broadcasts the message. More generally, the network could forward each message a few hops (each hop choosing a new node at random) before diffusing it. We call this approach diffusion-by-proxy, and it is conceptually equivalent to <mark>propagating over a line that changes for every transmission</mark>. Diffusion-by-proxy might seem like it should have low precision because the graph is so dynamic, but that intuition turns out to be false.

PROPOSITION 3. *The expected first-spy precision of diffusion-by-proxy is bounded as*

$$\mathbf{D}_{\text{FS}} \geq \frac{p}{1-p}(1 - e^{p-1}). \tag{9}$$

(Proof in Section A.2.3)

Intuitively, this statement holds because each node delivers its own message to the adversary with probability p, and few other nodes report to the adversary over the same edge. So even though diffusion-by-proxy breaks the symmetry of diffusion, it also provides many paths for messages to reach the adversary. Since there are many total paths to the adversary, each path sees (relatively) less traffic, which in turn reduces the amount of mixing that happens. A simple countermeasure is to reduce the number of paths over which messages can flow.

Lesson: <mark>There is anonymity in numbers; dense graphs achieve poor mixing because they do not constrain messages to flow over the same paths.</mark>

5. MAIN RESULT: DANDELION

The baseline spreading protocols from Section 4 provide us with a key guideline for building more anonymous networking policies: spread asymmetrically over a sparse graph. In this vein, we propose a new protocol: **dandelion spreading**. While the basic intuition of dandelion spreading is used in several point-to-point anonymous communication systems [17, 41], it has not been formally studied in the context of anonymous broadcast messaging.

Algorithm 1: DANDELION SPREADING. $\mathcal{N}_{out}(G, v)$ denotes the out-neighbors of node v on directed graph G.

Input: Message X_v, source v, anonymity graph G, spreading graph H, parameter $q \in (0, 1)$
anonPhase ← True
head ← v
recipients ← $\{v\}$
while *anonPhase* **do**
  /* forward message to random node   */
  target ∼ Unif($\mathcal{N}_{out}(G, \text{head})$)
  recipients ← recipients $\cup \{X_v\}$ from head to target
  head ← target
  $u \sim \text{Unif}([0, 1])$
  if $u \leq q$ **then**
    | anonPhase ← False
  end
end
/* Run diffusion over H from 'head'   */
DIFFUSION(X_v, head, H)

Dandelion spreading consists of an anonymity phase and a spreading phase (Algorithm 1). <mark>In the anonymity phase, the protocol spreads the message over a randomly-selected line for a random number of hops; in the spreading phase, the message is broadcast using diffusion until the whole network receives the message.</mark> In general, the two phases can occur over different graphs. In this work, we will design a (possibly time-varying) graph G over which the anonymity phase occurs, and we will assume the spreading phase occurs over the current Bitcoin P2P network H. The name 'dandelion spreading' reflects the spreading pattern's resemblance to a dandelion seed head (Figure 4).
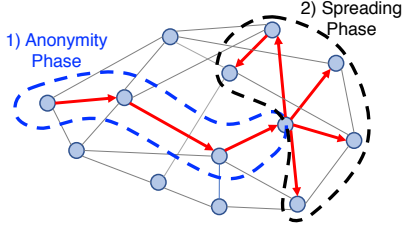
Figure 4: Dandelion spreading forwards a message in a line over the graph, then broadcasts it using diffusion. Here both phases occur over the same graph, i.e., $H = G$.

The two-phase nature of dandelion spreading allows us to separately design networking policies that optimize anonymity and latency. This separated architecture is not necessarily optimal in terms of a latency-anonymity tradeoff; exploring that tradeoff is an interesting direction for future work. However, diffusion is known to have good spreading properties [5], but poor anonymity properties [45]. Therefore, we combine it with an anonymity phase of constant duration (in an order sense), such that the average latency is increased by a small, bounded factor. We subsequently assume that the spreading phase can be fully deanonymized; i.e., the node that launches the diffusion process can be identified. As such, we only need to analyze the precision and recall of the anonymity phase. This assumption does not weaken our anonymity guarantees since it gives the adversary more power.

A key observation for this analysis is that the anonymity phase of dandelion spreading largely *removes the need for exact timestamps*. For honest server v, let $S'_v \subseteq S_v$ denote a trimmed down version of S_v, in which we retain only those transaction log tuples $(x, u, T_u(x))$ that correspond to the first time transaction x was received by an adversary from any honest node. That is, we only keep a tuple if u was the first spy to see message x, and x was delivered to u by honest exit node v. As before, let $\mathbf{S}'$ denote the vector of all S'_v's. Then, with dandelion spreading, it holds that $\mathbf{X} - (\mathbf{S}', \mathbf{\Gamma}) - \mathbf{S}$ forms a Markov chain. Therefore it is sufficient to use only the first observation information $\mathbf{S}'$ instead of $\mathbf{S}$ for computing transaction likelihoods. In fact, the sufficient statistic $\mathbf{S}'$ can be further simplified by ignoring the timestamp coordinate $T_u(x)$ in the tuples. This is possible due to our assumption that the transactions' originating times are unknown *a priori* to the adversary, which removes the observed timestamps from any temporal reference frame. Hence, in the remainder of this paper, with a slight abuse of notation, we use S_v, for honest server v, to denote the set of message tuples (x, u) such that (i) u was the first adversarial node to receive x and (ii) u received x from v. $\mathbf{S}$ denotes the vector of S_v's.

Note that a similar argument does not hold for spreading mechanisms like flooding or diffusion, in which multiple independent timestamps across different nodes (i.e., not just the first observation of a message) are used to compute likelihoods. The diversity of such observations allows the estimator to compare timestamps across nodes, thus making them useful for detection.

We begin by showing that the maximum recall for dandelion spreading over any connected topology is p, the lower bound from Theorem 2.

THEOREM 5. *The expected maximum recall for dandelion spreading on any connected graph of n nodes with a fraction p of adversaries is $\mathbf{R}_{\text{OPT}} = p$.*

(Proof in Section A.3.1)

The reason for this result is that dandelion spreading propagates content unidirectionally over a line. This lack of symmetry makes the first-spy estimator—which has a recall of p—optimal. Theorem 5 result implies that as we explore various topologies of dandelion spreading, we only need to analyze and minimize their precision. We do so for three topologies of the graph G: static trees, dynamic trees, and dynamic lines. Each topology provides intuition about how to achieve anonymity. We find that dynamic lines achieve nearly-optimal average precision and recall.

5.1 Static Trees

Recall that our goal is to mix messages from different users; in this sense, trees are a natural topology to study. That is, consider a rooted, directed d-regular tree, with each edge directed toward the parent node. Dandelion spreading respects the directedness of the graph, so during the anonymity phase, each node passes all messages to its parent node (i.e. toward the root). Nodes near the root are therefore able to mix their own messages with exponentially-many other messages from users beneath them in the tree. However nodes near the leaves of the tree have few nodes beneath them, and therefore experience minimal mixing. This fundamental asymmetry results in a high average precision.

PROPOSITION 4. *The expected precision under a matching estimator* MAT *on any tree is given by $\mathbf{D}_{\text{MAT}} = p$.*

(Proof in Section A.3.2)

Intuitively, when the graph is known, the adversary can partition nodes into *wards*, or sets of honest nodes that share the same first spy. Each ward contributes equally to the adversary's precision, so we would like to minimize the number of wards. On trees, the expected number of wards is $p\tilde{n}$, most of which consist of a single leaf with an adversarial parent node. This gives an overall precision of p.

Although a precision of p is an improvement over Bitcoin's current networking policy, we would like to achieve a precision close to the lower bound of p^2 (Theorem 2). We therefore consider topologies with fewer wards on average.

Lesson: Use topologies in which it is difficult for the adversary to partition nodes into wards.

5.2 Dynamic Trees

The adversary was able to partition the nodes of a static tree into wards largely because the graph was known. A natural question is whether dynamic trees have the same problem, since most of the graph is hidden, except the adversary's local neighborhood.

A perfect d-ary tree is a rooted tree in which each node has either d children or no children, and all leaves are at the same depth. Again, we assume each edge in such a tree is directed toward the parent node. We find that dandelion spreading on perfect d-ary trees has an expected precision similar to that of static trees.

PROPOSITION 5. *The expected precision of the first-spy estimator on a perfect d-ary tree, $d \geq 2$, can be bounded as $\mathbf{D}_{\text{FS}} \geq p/2$.*

(Proof in Section A.3.3)

Since the graph is now dynamic, the adversary cannot explicitly determine every ward like it could in the static case. However, the first-spy estimator naturally identifies wards that consist of a single honest leaf. Statistically, there are many such wards on trees that are not lines, so we obtain similar guarantees to the static case. This implies that the problem with trees is mainly the fact that they have many leaves.

Lesson: <mark>A dynamic graph does not mitigate the negative impact of leaf nodes.</mark>

5.3 Dynamic Lines: Dandelion

Next, we study dynamic line graphs. Lines are 2-regular trees, but unlike higher degree trees, they do not suffer from the asymmetry problems associated with leaves. However, line graphs seem to lack the strong mixing properties of higher-degree graphs. Nonetheless, we show near-optimal precision for this class of graphs. This happens because despite the moderate mixing on lines, the number of honest nodes visible to the adversary is also small. As such, the adversary cannot accurately partition nodes into wards, which reduces the overall precision. Note that this would not hold in the static case, since the adversary could identify the wards exactly.
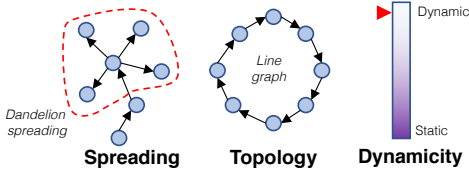
Figure 5: The DANDELION networking policy: (1) dandelion spreading, (2) a line topology, (3) a dynamic graph.

We use the name DANDELION to refer to a full networking policy (Figure 5): dandelion spreading over dynamic lines (i.e., 2-regular graphs with out-degree 1). We begin by showing that DANDELION has near-optimal precision.

THEOREM 6. *The expected precision of* DANDELION *(i.e., dandelion spreading on a dynamic line graph) with n nodes and a fraction $p < 1/3$ of adversaries, is upper bounded by*

$$\mathbf{D}_{\text{OPT}} \leq \frac{2p^2}{1-p} \log\left(\frac{2}{p}\right) + O\left(\frac{1}{n}\right). \tag{10}$$

(Proof in Section A.3.4)

This result states that for small p, the expected maximum precision is within a logarithmic factor of our lower bound of p^2. The stated bound has loose constants for improved readability; a tighter expression is included in the proof. The proof depends heavily on the fact that the adversary cannot reliably assign nodes to wards outside of its local neighborhood on the graph. As such, it is forced to use estimators that would give suboptimal precision in the static case, like variants of the first-spy estimator.

Figure 6 illustrates DANDELION's detection region compared to those of other benchmark policies. The points for diffusion and flooding are generated through simulation over a snapshot of the Bitcoin server graph from 2015 [32]. Since dandelion spreading has optimally-low recall (Theorem 5),
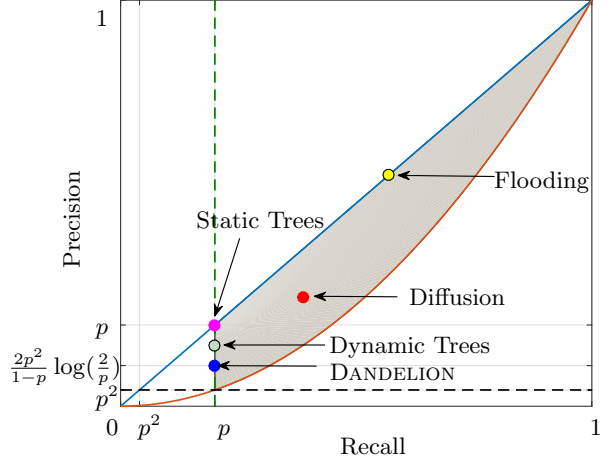
Figure 6: Detection regions for studied networking policies, $p = 0.2$. DANDELION has a detection region close to the fundamental lower bounds.

the Pareto frontier for DANDELION is exactly the plotted point. The other policies are analyzed using possibly-suboptimal estimators, so their detection regions must at least contain the plotted points.

DANDELION satisfies the theoretical demands of our problem. However, implementing DANDELION in a distributed, robust fashion is nontrivial. In the next section, we discuss some challenges associated with implementing DANDELION and present a simple, distributed implementation with empirically good performance.

6. SYSTEMS ISSUES

Theoretically, DANDELION is simple and exhibits desirable anonymity properties. However, the implementation raises a number of practical considerations, like how to construct the underlying line graph and how to provide sufficient graph dynamicity. We discuss each of these challenges, and introduce practical, heuristic solutions for addressing them.

6.1 Constructing a line graph

<mark>In DANDELION, all nodes propagate their messages over the *same* line.</mark> To implement this, the network must build either a Hamiltonian circuit or a set of long, disjoint lines in a fully-distributed fashion. Constructing a Hamiltonian circuit is challenging in our case because it is not a one-time event; in order to provide dynamicity, the network must frequently construct a new random line. To ensure scalability, the algorithm for constructing such a line should be fully-distributed, lightweight, and asynchronous.

Traditional algorithms for computing Hamiltonian circuits are often computationally intensive and/or require centralized control [22, 25], but recent papers have studied lightweight, distributed alternatives [26, 29]. For instance, [29] first generates line fragments, then splices them together into a circuit. However, it relies on the nodes of each line fragment knowing the identities of the fragment's head and tail nodes. This could partially reveal the graph structure to the adversary, which would likely change our anonymity guarantees.

On the other hand, [26] builds up the circuit sequentially; a pair of nodes start as the circuit 'seeds'. Each node v who joins the circuit contacts a random node u from the

Algorithm 2: k-APPROXIMATE LINE Approximates a directed line graph in a fully-distributed fashion. Each node picks an edge from k options

---

Input: Set $V = \{v_1, v_2, \ldots, v_n\}$ of nodes; parameter k
Output: A connected, directed graph $G(V, E)$ with
average degree 2
for $v \leftarrow V$ **do**
    /* pick k random targets   */
    $u_i \sim \text{Unif}(V \setminus \{v\}), \quad \text{for } i \in \{1, \ldots, k\}$
    /* pick the smallest in-degree   */
    $u \leftarrow \arg\min_{u_i} \deg_{in}(u_i)$
    $E = E \cup (v \rightarrow u)$ /* make connection   */
end
return $G(V, E)$

Figure 7: Degree distribution of k-approximate lines (Algorithm 2) for various k. The fraction of leaves decreases as the number of edge choices k increases.

Figure 8: The k edge choices during graph creation (Algorithm 2) do not significantly reduce the precision of the first-spy estimator beyond $k = 1$.

partially-built circuit; u replies with the IP address of its outgoing neighbor w. Then v splices itself into the (u, w) edge, so the new ordering is $u \rightarrow v \rightarrow w$. This distributed protocol is a viable solution for constructing an exact line.

Another alternative is to use Bitcoin's current networking strategy to approximate a line. Currently, each Bitcoin server generates eight connections at random. We can approximate a line by asking each server to create one outgoing connection at random. We can further refine this protocol by having each server, prior to making a connection, contact k nodes and connect to the node with the smallest in-degree. Algorithm 2 specifies this algorithm for approximating a line. The protocol is fully distributed, but it is unclear how well it approximates a line.

Figure 7 illustrates the degree distribution of Algorithm 2's approximation of a line graph with 1,000 nodes, averaged over 1,000 trials, for different values of k. First, note that the average degree is two by construction. As k increases, the fraction of leaves decays, with the greatest reduction coming as we transition from $k = 1$ to $k = 2$. This empirical observation is supported by the following proposition:

PROPOSITION 6. *Suppose Algorithm 2 is used to construct a k-approximate line over n nodes. Let the empirical degree distribution of the resulting graph's nodes have support $(d_1, \ldots, d_m)$, where $d_1 < \ldots < d_m$. Then with probability $1 - o(1)$, the maximum degree d_m satisfies the following condition:*

$$d_m = \begin{cases} \frac{\log n}{\log \log n}(1 + o(1)) + \Theta(1) & \text{if } k = 1 \\ \frac{\log \log n}{\log k}(1 + o(1)) + \Theta(1) & \text{if } k > 1. \end{cases}$$

(Proof in Section A.4.1)

Here we are using maximum degree as a proxy for regularity (or number of leaves), but recall that the expected degree is fixed by construction. Therefore, if we can drive the maximum degree down to 2, the minimum degree must also be 2. Proposition 6 suggests that we can reap most of the precision gains of a more regular graph by connecting to one of $k = 2$ nodes with minimum in-degree, whereas larger k only improves the regularity by a factor logarithmic in k.

Sec. 5.2 showed that leaves increase the precision of a scheme because the leaf nodes' messages cannot be mixed with other messages. This suggests that DANDELION can achieve lower precision over k-approximate lines (Algorithm 2) by increasing k and decreasing the number of leaves. Figure 8 compares t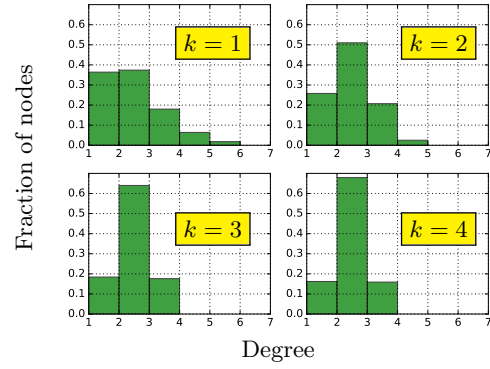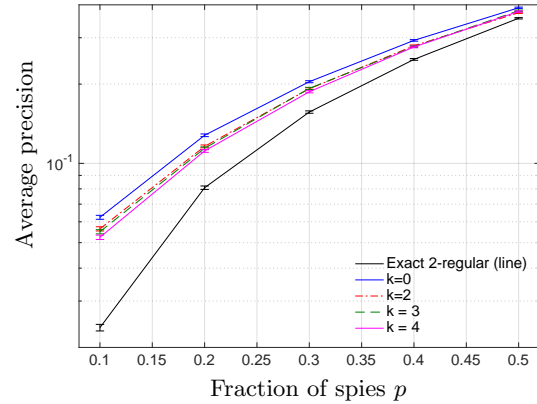he the first-spy estimator precision for exact lines (generated by [26]) and k-approximate lines (Algorithm 2). The figure shows that over k-approximate lines, average precision decreases as k increases (i.e., as the distribution becomes more regular), but the returns are diminishing in k. The most significant decrease in precision occurs as we transition from $k = 1$ to $k = 2$; higher values of k give marginal improvements. Moreover, the precision of k-approximate lines is significantly larger than that of exact lines, which could be obtained through the line-creation protocol in [26].

Algorithm 2 and [26] are both viable options for constructing a line. Although [26] has lower overall precision, it uses more fine-grained information—connection IPs rather than simple degree information. As such, [26] may be less robust to misbehaving nodes. Understanding this tradeoff, and developing alternatives that are resistant to adversarial misbehavior, are of practical interest.

6.2 Preventing graph leakage

Another challenge associated with DANDELION is that it assumes the graph G is a line whose structure is unknown to the adversary. However, lines can be learned over time. First, note that for any given adversarial node s_1 on a 2-regular digraph, s_1 can eventually learn the identities of the adversarial nodes immediately before and after it on the graph by sending probe messages. Now consider the fol-

lowing scenario: a message from an honest user propagates on the line, and passes s_1. At an honest node v between s_1 and the next adversarial node s_2 (see Figure 9), the message transitions into the spreading phase at and starts diffusing over the main P2P graph H. We assume that the adversary can reliably infer the diffusion source v. Since s_2 did not receive the message before the spreading phase began, and v was the source of the spreading phase, the adversary learns that v lies between s_1 and s_2. In this way, the adversary learns the internal, honest nodes of G at a rate proportional to the creation of new transactions; by learning this graph, the adversary's expected per-node precision grows to p.
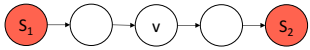

Figure 9: The adversary can easily learn line graphs.

This problem must be managed by changing the graph quickly enough that the adversary cannot learn it—i.e., on the timescale at which transactions are executed. As a ballpark estimate, the Bitcoin network currently sees about three transactions per second [10]. Botnets can be rented through online services; for \$200, one can rent a botnet of 1,000 US-based *zombies*, or corrupted hosts [15]. Since the current Bitcoin network consists of about 5,500 servers [9], this corresponds to $p \approx 0.15$; each ward would have about seven nodes on average, of which five are unknown to the adversary in the fully-dynamic setting. We conservatively assume that each transaction launches its spreading phase from a different honest node. If we want to ensure that the adversary never learns more than 40% of interior nodes, we should change the graph every 5500 transactions $\times \frac{5}{7} \times 0.4 \times \frac{1 \text{ sec}}{3 \text{ transactions}} \approx 9$ minutes. This is easy to enforce in a distributed fashion; every nine minutes, each node will tear down its connections and form new ones. Synchrony between nodes is not needed for this restructuring due to the fully-distributed line approximation protocol.

More powerful attackers can create botnets of tens of thousands of nodes, which would overwhelm DANDELION. In such scenarios, statistical solutions are no longer appropriate.

7. RELATED WORK

Three key facets of this work appear in the literature: practical anonymity attacks on Bitcoin, source detection analysis in diffusion processes, and the design of anonymous communication protocols.

Anonymity attacks on Bitcoin. There have been several attacks on the anonymity of Bitcoin, most of which harness the public nature of the blockchain [40, 35, 42]. Transaction patterns can be used to link user transactions over time, and in some cases identify the human owner of a public key.

More recently, authors have demonstrated deanonymization attacks on Bitcoin's networking stack. These attacks typically use the first-spy estimator, and achieve surprisingly high accuracies [28, 7, 8]. The Bitcoin community has responded to these attacks with ad hoc changes to its networking stack for improved anonymity [1].

Analysis of diffusion. A number of researchers have studied source detection on diffusion processes on graphs. These results show that for various classes of graphs and adversarial models, reliable deanonymization is possible [45, 46, 19, 48, 30, 39]. However, there has been a relative lack of theo-

retical results in the analysis of diffusion under a spy-based adversary like ours. Many of the results in this space propose effective heuristics that achieve high recall in practice [36, 51, 50]. These papers suggest that by using centrality information, adversaries may be able to launch stronger attacks than prior practical network attacks [7, 8].

Anonymous broadcasting. The topic of anonymous broadcasting has been studied for decades. The best-known example is dining cryptographer networks (DC nets), which enable a user to broadcast a message anonymously with information-theoretic guarantees [12]. DC nets are communication-intensive, which has prevented them from scaling beyond a few thousand nodes [49, 14, 23], and they are not well-suited to distributed systems like cryptocurrencies.

Another relevant topic is adaptive diffusion (AD) [18], which was recently proposed as an anonymous spreading protocol for broadcasting content over fixed graphs. AD shares some properties with DANDELION, such as symmetry-breaking. However, AD was designed for social networks, which do not require all nodes to receive every message. Indeed, AD can 'get stuck' on real graphs, meaning that some messages do not reach the entire network [18]. This property is unacceptable in cryptocurrencies: all nodes should receive all messages for fairness and consistency purposes.

Finally, the core idea of dandelion spreading—i.e., passing content through a set of proxies—has been used in numerous anonymity systems, mainly for point-to-point communication [17, 41]. However, existing systems have not connected dandelion spreading to any fundamental anonymity guarantees, and they typically assume a complete graph topology [17]. In contrast, we identify topologies over which dandelion spreading actually provides strong guarantees (i.e., not complete graphs). More fundamentally, our problem is focused on broadcasting over a network, which has different requirements and models than point-to-point messaging.

8. CONCLUSION

In this paper, we redesign the Bitcoin P2P stack to provide anonymity against distributed adversaries (e.g., botnets) who wish to link users to their transactions. We present the DANDELION networking policy, which achieves nearly-optimal anonymity guarantees with a simple, distributed implementation. Intuitively, DANDELION achieves these guarantees by mixing messages from different users on a graph that remains hidden from the adversary.

A few topics of interest during deployment are as follows: first, we have analyzed honest-but-curious adversarial nodes. In practice, botnet nodes may run malicious code that breaks protocol. In this case, anonymity can be negatively affected by nodes forwarding content inappropriately or misbehaving during graph construction. Hardening our protocols against such intrusions is critical. For example, noninteractive graph construction protocols offer some robustness by reducing opportunities for the adversary to lie in order to generate an advantageous anonymity graph. Alternatively, a system could use cryptographic proofs to ensure that nodes follow the graph construction protocol. Exploring these options is beyond the scope of this paper.

Second, we have paid less attention to the issue of message latency by assuming the two-phase architecture of dandelion spreading. As stated in Sec. 5, it is unclear if such an architecture is inherently optimal. Understanding the tradeoff between anonymity and latency is of fundamental interest.

9. REFERENCES

[1] Bitcoin core integration/staging tree. https://github.com/bitcoin/bitcoin.

[2] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun. Evaluating user privacy in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 34–51. Springer, 2013.

[3] M. Apostolaki, A. Zohar, and L. Vanbever. Hijacking bitcoin: Large-scale network attacks on cryptocurrencies. *arXiv preprint arXiv:1605.07524*, 2016.

[4] Y. Azar, A. Broder, A. Karlin, and E. Upfal. Balanced allocations. *SIAM Journal of Computing*, 29(1):180–200, 1999.

[5] M. Bartlett. Deterministic and stochastic models for recurrent epidemics. In *Proceedings of the third Berkeley symposium on mathematical statistics and probability*, volume 4, page 109, 1956.

[6] M. Bayati, D. Shah, and M. Sharma. Max-product for maximum weight matching: Convergence, correctness, and lp duality. *IEEE Transactions on Information Theory*, 54(3):1241–1251, 2008.

[7] A. Biryukov, D. Khovratovich, and I. Pustogarov. Deanonymisation of clients in bitcoin p2p network. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 15–29. ACM, 2014.

[8] A. Biryukov and I. Pustogarov. Bitcoin over tor isn't a good idea. In *2015 IEEE Symposium on Security and Privacy*, pages 122–134. IEEE, 2015.

[9] Bitnodes. Global bitcoin nodes distribution, 2016.

[10] Blockchain.info. Confirmed transactions per day, 2016.

[11] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten. Mixcoin: Anonymity for bitcoin with accountable mixes. In *International Conference on Financial Cryptography and Data Security*, pages 486–504. Springer, 2014.

[12] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of cryptology*, 1(1), 1988.

[13] CoinMarketCap. Cryptocurrency market capitalizations, 2016.

[14] H. Corrigan-Gibbs and B. Ford. Dissent: accountable anonymous group messaging. In *CCS*. ACM, 2010.

[15] D. Danchev. New underground service offers access to thousands of malware-infected hosts, 2013.

[16] C. Decker and R. Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*, pages 1–10. IEEE, 2013.

[17] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.

[18] G. Fanti, P. Kairouz, S. Oh, and P. Viswanath. Spy vs. spy: Rumor source obfuscation. In *SIGMETRICS Perform. Eval. Rev.*, volume 43, pages 271–284, 2015.

[19] V. Fioriti and M. Chinnici. Predicting the sources of an outbreak with a spectral technique. *arXiv:1211.2333*, 2012.

[20] M. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proc. CCS*. ACM, 2002.

[21] Z. Galil. Efficient algorithms for finding maximum matching in graphs. *ACM Computing Surveys (CSUR)*, 18(1):23–38, 1986.

[22] M. R. Garey, D. S. Johnson, and R. E. Tarjan. The planar hamiltonian circuit problem is np-complete. *SIAM Journal on Computing*, 5(4):704–714, 1976.

[23] P. Golle and A. Juels. Dining cryptographers revisited. In *Advances in Cryptology-Eurocrypt 2004*, 2004.

[24] N. L. Johnson and S. Kotz. Urn models and their application; an approach to modern discrete probability theory. 1977.

[25] D. Karger, R. Motwani, and G. Ramkumar. On approximating the longest path in a graph. *Algorithmica*, 18(1):82–98, 1997.

[26] J. Kim and R. Srikant. Peer-to-peer streaming over dynamic random hamilton cycles. In *Information Theory and Applications Workshop (ITA), 2012*, pages 415–419. IEEE, 2012.

[27] P. Koshy. *CoinSeer: A Telescope Into Bitcoin*. PhD thesis, The Pennsylvania State University, 2013.

[28] P. Koshy, D. Koshy, and P. McDaniel. An analysis of anonymity in bitcoin using p2p network traffic. In *International Conference on Financial Cryptography and Data Security*, pages 469–485. Springer, 2014.

[29] E. Levy, G. Louchard, and J. Petit. A distributed algorithm to find hamiltonian cycles in g(n, p) random graphs. In *Workshop on Combinatorial and Algorithmic aspects of networking*, pages 63–74. Springer, 2004.

[30] A. Y. Lokhov, M. Mézard, H. Ohta, and L. Zdeborová. Inferring the origin of an epidemic with dynamic message-passing algorithm. *arXiv preprint arXiv:1303.5315*, 2013.

[31] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage. A fistful of bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 127–140. ACM, 2013.

[32] A. Miller, J. Litton, A. Pachulski, N. Gupta, D. Levin, N. Spring, and B. Bhattacharjee. Discovering bitcoin?s public topology and influential nodes, 2015.

[33] M. Mitzenmacher. The power of two choices in randomized load balancing. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1094–1104, 2001.

[34] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[35] M. Ober, S. Katzenbeisser, and K. Hamacher. Structure and anonymity of the bitcoin transaction graph. *Future internet*, 5(2):237–250, 2013.

[36] P. C. Pinto, P. Thiran, and M. Vetterli. Locating the source of diffusion in large-scale networks. *Physical review letters*, 109(6):068702, 2012.

[37] D. Plohmann and E. Gerhards-Padilla. Case study of the miner botnet. In *2012 4th International Conference on Cyber Conflict (CYCON 2012)*, pages 1–16. IEEE, 2012.

[38] D. M. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.

[39] B. A. Prakash, J. Vreeken, and C. Faloutsos. Spotting culprits in epidemics: How many and which ones? In *ICDM*, volume 12, pages 11–20, 2012.

[40] F. Reid and M. Harrigan. An analysis of anonymity in the bitcoin system. In *Security and privacy in social networks*, pages 197–223. Springer, 2013.

[41] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security (TISSEC)*, 1(1):66–92, 1998.

[42] D. Ron and A. Shamir. Quantitative analysis of the full bitcoin transaction graph. In *International Conference on Financial Cryptography and Data Security*, pages 6–24. Springer, 2013.

[43] T. Ruffing, P. Moreno-Sanchez, and A. Kate. CoinShuffle: Practical decentralized coin mixing for bitcoin. In *European Symposium on Research in Computer Security*, pages 345–364. Springer, 2014.

[44] F. Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

[45] D. Shah and T. Zaman. Rumors in a network: Who's the culprit? *Information Theory, IEEE Transactions on*, 57:5163–5181, Aug 2011.

[46] D. Shah and T. Zaman. Rumor centrality: a universal source detector. In *ACM SIGMETRICS Performance Evaluation Review*, volume 40, pages 199–210. ACM, 2012.

[47] S. S. Silva, R. M. Silva, R. C. Pinto, and R. M. Salles. Botnets: A survey. *Computer Networks*, 57(2):378–403, 2013.

[48] Z. Wang, W. Dong, W. Zhang, and C. Tan. Rumor source detection with multiple observations: Fundamental limits and algorithms. In *ACM SIGMETRICS*, 2014.

[49] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson. Dissent in numbers: Making strong anonymity scale. In *OSDI*, pages 179–182, 2012.

[50] K. Zhu, Z. Chen, and L. Ying. Locating contagion sources in networks with partial timestamps. *arXiv preprint arXiv:1412.4141*, 2014.

[51] K. Zhu and L. Ying. A robust information source estimator with sparse observations. *arXiv preprint arXiv:1309.4846*, 2013.

APPENDIX

A. PROOFS

A.1 Section 3: Anonymity Metric Properties

A.1.1 Proof of Theorem 1

Consider any realization of the network, in which the messages $\mathcal{X}$ are mapped to the servers V_H according to mapping rule M. Then from the definition of precision and recall at any node v (Equations (2), (3)), we have

$$D_{\text{M}}(v) = \frac{\mathbb{1}\{\text{M}(X_v) = v\}}{\sum_{w \in V_H} \mathbb{1}\{\text{M}(X_w) = v\}}$$
$$\leq \mathbb{1}\{\text{M}(X_v) = v\} = R_{\text{M}}(v). \qquad (11)$$

Hence it follows that the macro-averaged precision D_{M} is at

most the macro-averaged recall R_{M}, implying $\mathbf{D_M} \leq \mathbf{R_M}$.

To prove inequality (b), let $V_{\text{M}} = \{v \in V_H : \text{M}(X_v) = v\}$ denote the set of servers whose corresponding messages are correctly mapped by M. Further, for each such node $v \in V_{\text{M}}$, let $I_v = \{x \in \mathcal{X} : \text{M}(x) = v, x \neq X_v\}$ denote all the messages (other than v's own message X_v) that are mapped to v. Then, by definition we have $R_{\text{M}} = |V_{\text{M}}|/\tilde{n}$ and

$$\tilde{n}D_{\text{M}} = \sum_{v \in V_H} D_{\text{M}}(v) = \sum_{v \in V_{\text{M}}} \frac{1}{|I_v| + 1}$$
$$\geq \frac{|V_{\text{M}}|^2}{\sum_{v \in V_{\text{M}}}(|I_v| + 1)} \geq \frac{|V_{\text{M}}|^2}{\tilde{n}} = \tilde{n}R_{\text{M}}^2 \qquad (12)$$

where Equation (12) follows from the arithmetic-mean harmonic-mean (A.M-H.M) inequality and $\sum_{v \in V_{\text{M}}}(|I_v|+1) \leq \tilde{n}$. Hence we have $R_{\text{M}} \leq \sqrt{D_{\text{M}}}$, which upon taking expectation and using Jensen's inequality, yields $\mathbf{R_M} \leq \sqrt{\mathbf{D_M}}$.

A.1.2 Proof of Theorem 2

Recall that for honest server v, the tuple $(x, u, T_u(x))$ is contained in S_v if v forwards message x to adversarial node u at time $T_u(x)$. Let us now define a related quantity $\bar{S}_v$ to denote the set of messages $x \in \mathcal{X}$ forwarded by v to some adversary such that x was not received by any adversarial node previously. This quantity $\bar{S}_v$ is useful in analyses involving the first-spy estimator. $\bar{\mathbf{S}}$ denotes the vector of all $\bar{S}_v$'s.

LEMMA 1. *If $v \in V_H$ is a honest server node in a network with a fraction p of adversaries, then the recall of the first-spy estimator is $\mathbf{R}_{\text{FS}}(v) = \mathbb{P}(X_v \in \bar{S}_v) \geq p$.*

PROOF. Let $U \in \Gamma(v)$ denote the node to which v first sends its message X_v. Then,

$$\mathbb{P}(U \in V_A) = \sum_{u \in V, u \neq v} \mathbb{P}(U = u)\mathbb{P}(U \in V_A | U = u)$$
$$= \sum_{u \in V, u \neq v} \frac{1}{n-1}\mathbb{P}(U \in V_A | U = u) = \frac{np}{n-1} \geq p, \qquad (13)$$

due to uniform distribution among the remaining nodes $V \setminus \{v\}$. Therefore we have,

$$\mathbb{P}(X_v \in \bar{S}_v) \geq \mathbb{P}(U \in V_A) \geq p. \qquad (14)$$

Thus v's message is contained in $\bar{S}_v$ with probability at least p. The case where v simultaneously broadcasts X_v to multiple nodes can also be similarly bounded as above, and hence the lemma follows. $\square$

To show (7), note that $\mathbf{R}_{\text{OPT}} \geq \mathbf{R}_{\text{FS}}(v) \geq p$, by Lemma 1. Next, we show that the first-spy estimator also has a precision of at least p^2 regardless of the topology or spreading scheme. Consider a random realization $\bar{\mathbf{S}}$, in which the adversaries observe a set of first-received messages $S_v \subseteq \mathcal{X}$ from each node $v \in V$. Now, supposing in these observations there exists a subset of t server nodes $\{v_1, v_2, \ldots, v_t\}$ whose own messages are included in the respective forwarded sets, i.e., $X_{v_i} \in \bar{S}_{v_i} \forall i = 1, 2, \ldots, t$. The macro-averaged precision in this case is

$$D_{\text{FS}} = \frac{1}{\tilde{n}} \sum_{i=1}^{t} \frac{1}{|\bar{S}_{v_i}|} \geq \frac{t^2}{\tilde{n} \sum_{i=1}^{\tilde{n}} |\bar{S}_{v_i}|} \geq \frac{t^2}{\tilde{n}^2}, \qquad (15)$$

where the first inequality above is due to the arithmetic-mean harmonic-mean (A.M-H.M) inequality, and the second

14

inequality is because the total number of messages is at most $\tilde{n}$. Equation (15) in turn implies that

$$\mathbb{E}[D_{\text{FS}}|T=t] \geq \frac{t^2}{\tilde{n}^2}. \tag{16}$$

The overall expected detection precision can then be bounded as

$$\mathbf{D}_{\text{FS}} = \mathbb{E}[D_{\text{FS}}] = \sum_{t=0}^{\tilde{n}} \mathbb{P}(T=t)\mathbb{E}[D_{\text{FS}}|T=t]$$

$$\geq \sum_{t=0}^{\tilde{n}} \mathbb{P}(T=t)\frac{t^2}{\tilde{n}^2} = \frac{\mathbb{E}[T^2]}{\tilde{n}^2} \geq \frac{\mathbb{E}[T]^2}{\tilde{n}^2}$$

$$= \frac{\mathbb{E}[\sum_{v \in V_H} \mathbf{1}_{X_v \in \bar{S}_v}]^2}{\tilde{n}^2} \geq \frac{(p\tilde{n})^2}{\tilde{n}^2} = p^2, \tag{17}$$

where the inequality in Equation (17) follows from Lemma 1. Finally by definition we have $\mathbf{D}_{\text{OPT}} \geq \mathbf{D}_{\text{FS}}$ and hence the theorem follows.

A.1.3 Proof of Theorem 3

Let us first prove that the optimal mapping must be a matching. Supposing otherwise, consider a mapping $\mathtt{M} \in \mathcal{M}$ that is not a matching. Then there exists a server v that is mapped to the most number of messages $\{x_1, x_2, \ldots, x_k\}$ ($k > 1$) in $\mathtt{M}$. This also implies there exists another node $u \in V_H$ such that no message is mapped to u. Now, the expected precision at server v is given by

$$\mathbb{E}[D_{\mathtt{M}}(v)|\mathbf{O}] = \frac{\sum_{i=1}^{k} \mathbb{P}(X_v = x_i|\mathbf{O})}{k}$$
$$\leq \max_{i \in \{1,\ldots,k\}} \mathbb{P}(X_v = x_i|\mathbf{O}). \tag{18}$$

On the other hand, the expected precision at u is zero. Now, consider an alternative mapping $\mathtt{M}' \in \mathcal{M}$ in which all messages $x \in \mathcal{X}$ are mapped to servers exactly as in $\mathtt{M}$ except for the message x_{i^*} where $i^* = \text{argmin}_{i \in \{1,\ldots,k\}} \mathbb{P}(X_v = x_i|\mathbf{O})$ which is mapped to server u. In this case, the expected precision at v becomes

$$\mathbb{E}[D_{\mathtt{M}'}(v)|\mathbf{O}] = \frac{\sum_{i=1, i \neq i^*}^{k} \mathbb{P}(X_v = x_i|\mathbf{O})}{k-1}$$
$$\geq \mathbb{E}[D_{\mathtt{M}}(v)|\mathbf{O}], \tag{19}$$

while the expected precision at u is

$$\mathbb{E}[D_{\mathtt{M}'}(u)|\mathbf{O}] = \mathbb{P}(X_u = x_{i^*}|\mathbf{O}) \geq 0. \tag{20}$$

As such the total expected precision at servers u and v is

$$\mathbb{E}[D_{\mathtt{M}'}(v) + D_{\mathtt{M}'}(u)|\mathbf{O}] \geq \mathbb{E}[D_{\mathtt{M}}(v) + D_{\mathtt{M}}(u)|\mathbf{O}]$$
$$\Rightarrow \mathbb{E}[D_{\mathtt{M}'}|\mathbf{O}] \geq \mathbb{E}[D_{\mathtt{M}}|\mathbf{O}]. \tag{21}$$

Thus we have constructed a new mapping $\mathtt{M}'$ whose expected precision is at least as much as $\mathtt{M}$ and in which the maximum number of messages mapped to any server is smaller by 1.[2] Continuing this process, we conclude that for any mapping $\mathtt{M} \in \mathcal{M}$ there exists another matching mapping $\mathtt{M}'$ such that $\mathbb{E}[D_{\mathtt{M}'}|\mathbf{O}] \geq \mathbb{E}[D_{\mathtt{M}}|\mathbf{O}]$. Thus the optimizing mapping is achieved by a matching.

Now, let $\mathcal{M}^*$ denote the set of all matchings in the bipartite graph $(V_H, \mathcal{X})$. By the first part of the theorem above,

---

[2]In case of ties, we repeat the above process to each of the servers until the maximum server degree reduces by one.

we can restrict our search to $\mathcal{M}^*$ for finding the optimal mapping. As such,

$$\mathbb{E}[D_{\text{OPT}}|\mathbf{O}] = \max_{\mathtt{M} \in \mathcal{M}^*} \mathbb{E}[D_{\mathtt{M}}|\mathbf{O}]$$
$$= \max_{\mathtt{M} \in \mathcal{M}^*} \sum_{(v,x) \in \mathtt{M}} \mathbb{P}(X_v = x|\mathbf{O}), \tag{22}$$

implying that the optimum is achieved by a maximum weight matching.

A.1.4 Proof of Corollary 1

Let $\mathtt{M} \in \mathcal{M}$ be any mapping under observations $\mathbf{O} = (\mathbf{S}, \mathbf{\Gamma})$. Consider a server v and let $\{x_1, x_2, \ldots, x_k\}$ be the set of messages that are mapped to v in $\mathtt{M}$. Then,

$$\mathbb{E}[D_{\mathtt{M}}(v)|\mathbf{O}] \leq \frac{\sum_{i=1}^{k} \mathbb{P}(X_v = x_i|\mathbf{O})}{k}$$
$$\leq \max_{i \in \{1,\ldots,k\}} \mathbb{P}(X_v = x_i|\mathbf{O})$$
$$\leq \max_{x \in \mathcal{X}} \mathbb{P}(X_v = x|\mathbf{O}). \tag{23}$$

Since the above Equation (23) holds for any mapping $\mathtt{M}$, it must hold for the optimal mapping as well.

A.1.5 Proof of Theorem 4

We want to prove that the optimal mapping must map each message $x \in \mathcal{X}$ to a server v that maximizes $\mathbb{P}(X_v = x|\mathbf{O})$. Supposing otherwise, let us consider a mapping $\mathtt{M} \in \mathcal{M}$ where there exists a server w that is mapped to a set of messages $\{x_1, x_2, \ldots, x_k\}$ ($k \geq 1$), where w.l.o.g. $w \notin \text{argmax}_{v \in V_H} \mathbb{P}(X_v = x_1|\mathbf{O})$. The expected recall at server w is given by

$$\mathbb{E}[R_{\mathtt{B}}(w)|\mathbf{O}] = \sum_{i=1}^{k} \mathbb{P}(X_w = x_i|\mathbf{O}).$$

Further, consider another node $u \in V_H$ such that $u \in \text{argmax}_{v \in V_H} \mathbb{P}(X_v = x_1|\mathbf{O})$. Suppose it is mapped to a different set of messages $\{y_1, \ldots, y_j\}$. The expected recall for node u is

$$\mathbb{E}[R_{\mathtt{B}}(u)|\mathbf{O}] = \sum_{i=1}^{j} \mathbb{P}(X_u = y_i|\mathbf{O}).$$

Now, consider an alternative mapping $\mathtt{M}' \in \mathcal{M}$ in which all messages $x \in \mathcal{X}$ are mapped to servers exactly as in $\mathtt{M}$ except for the message x_1, which is mapped to server u. In this case, the expected recall at w becomes

$$\mathbb{E}[R_{\mathtt{M}'}(w)|\mathbf{O}] = \sum_{i=2}^{k} \mathbb{P}(X_w = x_i|\mathbf{O})$$

while the expected recall at u is

$$\mathbb{E}[R_{\mathtt{M}'}(u)|\mathbf{O}] = \mathbb{E}[R_{\mathtt{M}}(u)|\mathbf{O}] + \mathbb{P}(X_u = x_1|\mathbf{O}). \tag{24}$$

As such the total expected precision at servers u and v is

$$\mathbb{E}[D_{\mathtt{M}'}(w) + D_{\mathtt{M}'}(u)|\mathbf{O}] \geq \mathbb{E}[D_{\mathtt{M}}(w) + D_{\mathtt{M}}(u)|\mathbf{O}]$$
$$\Rightarrow \mathbb{E}[D_{\mathtt{M}'}|\mathbf{O}] \geq \mathbb{E}[D_{\mathtt{M}}|\mathbf{O}]. \tag{25}$$

Thus we have constructed a new mapping $\mathtt{M}'$ whose expected precision is at least as much as $\mathtt{M}$ and in which the number of messages mapped to servers with sub-maximal likelihood is reduced by one. Continuing this process, we conclude that for any mapping $\mathtt{M} \in \mathcal{M}$ there exists another mapping

M' such that each message x is mapped to a server $v^* \in \arg\max_{v \in V_H} \mathbb{P}(X_v = x | \mathbf{O})$.

A.2 Section 4: Baseline Algorithms

A.2.1 Proof of Proposition 1

Consider the broadcasting experiment on a random realization G of the network topology. In the static case this topology is completely known to the adversary. As defined in Section A.1.2, for any honest server $v \in V_H$, let $\bar{S}_v$ denote the set of transactions $x \in \mathcal{X}$ such that x is directly forwarded by v to some adversary and x has not been received by any adversary previously. By our assumption on flooding, this means a server v's message is contained in $\bar{S}_u$ if and only if

(i) u is reachable from v,

(ii) u has an out-going edge to an adversary and

(iii) no other node $u' \in V_H$ that satisfies the previous two conditions is strictly closer to v than u.

Thus by looking at the graph G, the adversary can construct a bipartite graph $B(V_H, V_H)$ in which there is edge $(u, v) \in V_H \times V_H$ if and only if X_v will be contained in $\bar{S}_u$. Further for each $u \in V_H$, let $W_u = \{v \in V_H : (u, v) \in B, \nexists\, u' \neq u \text{ s.t. } (u', v) \in B\}$ denote the set of server nodes whose messages reach only the message set $\bar{S}_u$. Note that for servers $v \in V_H$ that have an out-going edge to an adversary, we must have $v \in W_v$.

Now, once the messages have been broadcast in G, consider the following mapping strategy M for the adversary. First, for each set $\bar{S}_v$ we compute a subset $\bar{S}_v' = \{x \in \bar{S}_v : x \notin \bar{S}_u \forall u \neq v\}$. Such a set $\bar{S}_v'$ corresponds to the messages that were delivered to the adversaries only by server v and no other server. Thus, the messages in $\bar{S}_v'$ must precisely belong to the servers in W_v. As such, the adversary's mapping strategy can be: (i) for each $v \in V_H$ pick a random matching mapping between $\bar{S}_v'$ and W_v and (ii) assign any remaining messages randomly to the remaining server nodes. Let $\mathcal{E}_v$ denote the event that v has an out-going edge to an adversary. The payoff can then be bounded as

$$\mathbb{E}[\tilde{n}D_{\texttt{M}}|G] \geq \mathbb{E}\left[\sum_{\substack{v \in V_H:\\|W_v| \geq 1}} \sum_{u \in W_v} \mathbb{1}\{\texttt{M}(X_u) = u\}\,\bigg|\, G\right]$$

$$= \sum_{\substack{v \in V_H:\\|W_v| \geq 1}} \sum_{u \in W_v} \mathbb{E}[\mathbb{1}\{\texttt{M}(X_u) = u\}|G]$$

$$= \sum_{\substack{v \in V_H:\\|W_v| \geq 1}} \sum_{u \in W_v} \frac{1}{|W_v|} = \sum_{v \in V_H} \mathbb{1}\{|W_v| \geq 1\}, \quad (26)$$

where Equation (26) follows because in a random matching any message in S_v' is likely to be assigned to its true server in W_v with probability $1/|W_v|$. Hence the total average precision is bounded by

$$\tilde{n}\mathbf{D}_{\texttt{M}} \geq \mathbb{E}\left[\sum_{v \in V_H} \mathbb{1}\{|W_v| \geq 1\}\right] \quad (27)$$

$$\geq \sum_{v \in V_H} \mathbb{P}(\mathcal{E}_v) = \tilde{n}(1 - (1 - p)^d) \quad (28)$$
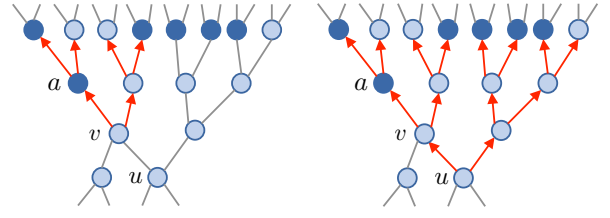
and we have the proposition.

Figure 10: Comparison of the number of adversarial nodes receiving a message in 2 rounds following first reception when the source is (i) directly connected to an adversary (left) and (ii) away from the adversary (right). The propagation of the message is shown in red; darkened nodes are adversarial.

A.2.2 Proof Sketch of Proposition 2

Before we begin the proof, first notice that in the dynamic setting the adversaries are directly connected to at most dpn (i.e., roughly a fraction p) honest servers, while the rest of the server locations are unknown to the adversary. Since the hidden servers can only be trivially de-anonymized, in order to obtain our claimed average precision of $O(p)$ each of the servers visible to the adversary must be de-anonymized with a high precision close to 1. Indeed, in the following we describe a simple mapping scheme M that achieves this high precision.

For a server $v \in V_H$, let $\mathcal{E}_v$ denote the event that at least one of v's out-going edges is connected to an adversary. Consider then, the spreading of v's message X_v in the graph under event $\mathcal{E}_v$. Since G is a random d-regular graph, using the result in [?], there is almost surely a regular tree of depth at least $\frac{1}{2}\log_{d-1} n$ rooted at v. For simplicity, let us consider $d = 4$ in which case there is a tree of depth at least $\frac{1}{4}\log n$ rooted at v almost surely. Thus v's message propagates along this tree, reaching two nodes (at least one of which is an adversary, due to $\mathcal{E}_v$) in the first round and subsequently reaching 2^i new nodes in the i-th round for each $i < \frac{1}{4}\log n$. Since a fraction p of the nodes are adversarial, this implies in the i-th round we expect roughly $p2^i$ adversarial nodes to receive X_v. On the other hand, if some other server u upstream of v had started broadcasting its message X_u, then more adversarial nodes would have received X_u in the i-th round following reception by the first adversarial node (see Figure 10).

The above observation then, naturally motivates a mapping M as described in Algorithm 3. In this strategy, the adversary simply counts the number of adversarial nodes that received a particular message $x \in \mathcal{X}$ at a time $\frac{1}{4}\log n - 1$ rounds after the message was first received by some adversarial node a. If this number is small ($< 2pn^{1/4}$) then we conclude the source of x to be the server v that sent the message to the first adversary. Otherwise the message is randomly assigned to an unassigned server at the end.

The algorithm works because if v were truly the source of x, then in the $(\frac{1}{4}\log n - 1)$-th round following reception by a, the number of adversarial nodes to receive x is less than $2pn^{1/4}$ with a probability at least $1 - 2^{-\log(4/e)pn^{1/4}}$ by the Chernoff bound. On the other hand if v were not the true source of x, then x was initially broadcast at a time at least 2 rounds before a received it. This implies at least $2pn^{1/4}$ adversarial nodes receive the message at a time $\frac{1}{4}\log n - 1$

16

Algorithm 3: Mapping algorithm under flooding for a dynamic 4-regular graph.

Input: Time-stamp $T_v(x)$ and sender $S_v(x)$ for each
       message $x \in \mathcal{X}$ received by adversary v for all
       $v \in V_A$.
Output: Mapping M from $\mathcal{X}$ to V_H
$I \leftarrow V_H, J \leftarrow \mathcal{X}$
for *each* $x \in \mathcal{X}$ **do**
    | $a_{\text{init}} \leftarrow \arg\min_{v \in V_A} T_v(x)$
    | $T_{\text{init}} \leftarrow T_{a_{\text{init}}}(x)$
    | $v_{\text{init}} \leftarrow S_{a_{\text{init}}}(x)$
    | $\eta \leftarrow |\{v \in V_A : T_v(x) = T_{\text{init}} + \frac{1}{4}\log n - 1\}|$
    | **if** $\eta < 2pn^{1/4}$ **then**
    | | $M(x) \leftarrow v_{\text{init}}$
    | | $I \leftarrow I\backslash\{v_{\text{init}}\}$
    | | $J \leftarrow J\backslash\{x\}$
    | **end**
end
Randomly assign messages in J to servers in I
return M

---

rounds following reception by a. Thus the total probability of error can be bounded by the union bound, to yield that whenever a server v's out-going edges are connected to at least one adversarial node, X_v is mapped to v with precision 1 with high probability. Such an event $\mathcal{E}_v$ happens with a probability at least p to conclude the Theorem.

A.2.3 *Proof of Proposition 3*

For any message X_u, let $\Pi_u = (\Pi_{1,u}, \Pi_{2,u}, \dots, \Pi_{L_u,u})$ be the path taken by a message from its source u $(= \Pi_{1,u})$ until it reaches an adversarial node $\Pi_{L_u,u}$ for the first time (L_u denotes the length of the path). Further for any two nodes $v, u \in V_H$, let $\mathcal{E}_{u,v}$ denote the event that u's message X_u reaches the adversary through server v, i.e., $\Pi_{1,u} = u, \Pi_{2,u} \notin V_A, \Pi_{3,u} \notin V_A, \dots, \Pi_{k-2,u} \notin V_A, \Pi_{k-1,u} = v$ and $\Pi_{k,u} \in V_A$. Then by counting over paths of all possible lengths, we can evaluate probability of $\mathcal{E}_{u,u}$ as

$$\mathbb{P}(\mathcal{E}_{u,u}) = \sum_{l \geq 2} \mathbb{P}(L_k = l, \mathcal{E}_{u,u})$$

$$= \left(\frac{np}{n}\right) + \sum_{l \geq 3} \left(\frac{\tilde{n}}{n}\right)^{l-3} \left(\frac{1}{n}\right)\left(\frac{np}{n}\right) = p + \frac{1}{n}. \quad (29)$$

Similarly, for $u \in V_H, u \neq v$,

$$\mathbb{P}(\mathcal{E}_{u,v}) = \sum_{l \geq 3}\left(\frac{\tilde{n}}{n}\right)^{l-3}\left(\frac{1}{n}\right)\left(\frac{np}{n}\right) = \frac{1}{n}. \quad (30)$$

Further, since the messages are all forwarded independently the set of events $\{\mathcal{E}_{v,u} : v \in V_H\}$ are mutually independent for each server $u \in V_H$. Hence the expected cost incurred at a server under the first-spy estimator can be written as

$$\mathbf{D}_{\text{FS}}(v) = \left(p + \frac{1}{n}\right)\mathbb{E}\left[\frac{1}{1 + Z_v}\right]$$

$$= \left(p + \frac{1}{n}\right)\frac{1}{\tilde{n}\frac{1}{n}}\left(1 - \left(1 - \frac{1}{n}\right)^{\tilde{n}}\right), \quad (31)$$

where $Z_v = \sum_{u \in V_H, u \neq v} \mathbb{1}\{\mathcal{E}_{u,v}\}$ is the number of messages, other than X_v, that reach the adversary through v and

$Z_v \sim \text{Binom}\left(\tilde{n} - 1, \frac{1}{n}\right)$ because of independence of messages and Equation (30). The last equation above can be further simplified to yield the bound

$$\mathbf{D}_{\text{FS}}(v) \geq \frac{p}{1-p}(1 - e^{p-1}), \quad (32)$$

which when averaged over all honest nodes $v \in V_H$ gives us the desired result.

A.3 Section 5: Main Result - Dandelion

A.3.1 *Proof of Theorem 5*

We first show that the first-spy estimator is recall-optimal for dandelion spreading, then that the first-spy estimator has an expected recall of p.

To show the first step, i.e., $\mathbf{R}_{\text{OPT}} = \mathbf{R}_{\text{FS}}$, Theorem 4 implies that we must show that for every message x, its exit node z (i.e., the node implicated by the first-spy estimator) maximizes $\mathbb{P}(X_v = x|\mathbf{O})$. For any message X_u, let $\Pi_u = (\Pi_{1,u}, \Pi_{2,u}, \dots, \Pi_{L_u,u})$ be the path taken by a message from its source u $(= \Pi_{1,u})$ until it reaches an adversarial node $\Pi_{L_u,u}$ for the first time (L_u denotes the length of the path). From the adversary's observation $\mathbf{S}$, $\Pi_{L_u-1,u}$ and $\Pi_{L_u,u}$ are fixed as the exit node z and the first spy for X_u, respectively. Due to the specification of dandelion spreading (Algorithm 1), the likelihood of this path, $\mathcal{L}(\Pi_u)$, is $\mathcal{L}(\Pi_u) = \prod_{i=1}^{L_u-1} \frac{1}{\deg(\Pi_{i,u})}$, where $\deg(v)$ denotes the out-degree of v. Assuming a uniform prior over candidate sources, we have $\mathbb{P}(X_v = x|\mathbf{O}) \propto \mathcal{L}(\Pi_v)$. Since each node is assumed to have an out-degree of at least 1, this likelihood is maximized by taking the shortest path possible. That is, the maximum-likelihood path over all paths originating at honest candidate sources gives $z \in \arg\max_{v \in V_H} \mathbb{P}(X_v = x|\mathbf{O})$. Hence the first-spy estimator is also a maximum-recall estimator.

Now we analyze the recall of the first-spy estimator. Let $\mathcal{P}_v$ denote the event that v's parent (i.e., the next node in the line) is adversarial. Then the expected recall is

$$\mathbf{R}_{\text{OPT}} = \mathbb{E}[R_{\text{FS}}|\mathbf{S}, G] = \frac{1}{\tilde{n}}\mathbb{E}\left[\sum_{v \in V_H} \mathbb{1}\{\mathcal{P}_v\}\right]$$

$$\Rightarrow \mathbf{R}_{\text{OPT}} = \frac{1}{\tilde{n}}\sum_{v \in V_H}\mathbb{P}(\mathcal{P}_v) = \frac{1}{\tilde{n}}\sum_{v \in V_H}\left(\frac{n-1}{n}\frac{np}{n-1}\right) = p.$$

A.3.2 *Proof of Proposition 4*

For any node v, let $W_v = \{u \in V_H : X_u \in S_v\}$ denote the ward under node v and let $\mathbf{W}$ denote the set of all wards. Note that in the tree topology, the wards W_v can be completely determined from knowledge of the graph G. Also, let $I(v)$ denote the node $u \in V_H$ such that $X_v \in S_u$. Then, the expected cost at a node v under observations $\mathbf{O} = (\mathbf{S}, G)$ by the adversary can be written as

$$\mathbb{E}[D_{\text{M}}(v)|\mathbf{S}, G] = \mathbb{E}[D_{\text{M}}(v)|\mathbf{S}, G, \mathbf{W}] = \frac{1}{|W_{I(v)}|}. \quad (33)$$

This follows because the matching estimator MAT assigns the messages in S_v to the nodes in W_v as a random matching, and hence the probability of a node receiving the correct message is $1/|W_v|$. Summing Equation (33) over all honest nodes, we have

$$\mathbb{E}[D_{\text{MAT}}|\mathbf{S}, G] = \sum_{v \in V_H}\frac{1}{|W_{I(v)}|} = |W|, \quad (34)$$

where $|W| = |\{v : W_v \neq \emptyset\}|$ denotes the number of non-empty wards, and $\emptyset$ denotes the null set. Now, let $\mathcal{P}_v$ denote the event that v's parent is adversarial. Since a ward under a node v is non-empty iff v's parent is adversarial, Equation (34) above becomes

$$\mathbb{E}[D_{\text{MAT}}|\mathbf{S}, G] = \frac{1}{\tilde{n}}\mathbb{E}\left[\sum_{v \in V_H} \mathbb{1}\{\mathcal{P}_v\}\right] \qquad (35)$$

$$\Rightarrow \mathbf{D}_{\text{MAT}} = \frac{1}{\tilde{n}}\sum_{v \in V_H} \mathbb{P}(\mathcal{P}_v) = \frac{1}{\tilde{n}}\sum_{v \in V_H}\left(\frac{n-1}{n}\frac{np}{n-1}\right) = p.$$

A.3.3   Proof of Proposition 5

First note that since the tree is dynamic, the adversary's observations consists of $\mathbf{O} = (\mathbf{S}, \Gamma(V_A))$, i.e., the transaction logs and the local neighborhood of adversarial nodes. Now for any honest node $v \in V_H$, let $\mathcal{E}_v$ denote the event that (i) v occurs at a position in G which is a leaf of the tree and (ii) v's parent is an adversary. Similarly let $\mathcal{I}_v$ denote the event that $v \in V_H$ occurs at the interior of the tree. We first show that whenever $\mathcal{E}_v$ happens, v is detected with certainty under the first-spy estimator, i.e.,

$$\mathbb{E}[D_{\text{FS}}(v)|\mathbf{S}, \Gamma(V_A), \mathcal{E}_v] = 1. \qquad (36)$$

This is because $D_{\text{FS}}(v) = \frac{\sum_{x \in S_v} \mathbb{1}\{X_v = x\}}{|S_v|}$ in the first-spy estimator and $S_v = \{X_v\}$ whenever $\mathcal{E}_v$ happens. As such,

$$\mathbb{E}[D_{\text{FS}}(v)|\mathbf{S}, \Gamma(V_A), \mathcal{E}_v] = \mathbb{E}[\mathbb{1}\{X_v = X_v\}|\mathbf{S}, \Gamma(V_A), \mathcal{E}_v] = 1$$
$$\Rightarrow \mathbb{E}[D_{\text{FS}}(v)|\mathcal{E}_v] = 1. \qquad (37)$$

Hence the expected payoff becomes

$$\mathbf{D}_{\text{FS}}(v) = \mathbb{P}(\mathcal{E}_v)\mathbb{E}[D_{\text{FS}}(v)|\mathcal{E}_v] + \mathbb{P}(\mathcal{I}_v)\mathbb{E}[D_{\text{FS}}(v)|\mathcal{I}_v] \qquad (38)$$

$$\geq \mathbb{P}(\mathcal{E}_v)\mathbb{E}[D_{\text{FS}}(v)|\mathcal{E}_v] = \frac{1}{2}\frac{np}{(n-1)} \geq \frac{p}{2}, \qquad (39)$$

since at least half of the nodes are leaves in a perfect d-ary tree. Summing over all honest nodes gives the result.

A.3.4   Proof of Theorem 6

As in the case of dynamic trees, the adversary's observations consists of $\mathbf{O} = (\mathbf{S}, \Gamma(V_A))$ in the dynamic line as well. The proof works by evaluating the cost incurred under various possibilities for the local neighborhood structure around a node in the network. For any honest server node $v \in V_H$, let $\mathcal{E}_v(i, j)$ denote the event that (i) i nodes preceding v are honest nodes, the $(i+1)$-th node preceding v is adversarial and (ii) j nodes succeeding v are honest nodes and the $(j+1)$-th node following v is adversarial. Also for ease of notation let $\mathcal{I}_v$ denote the event $\cup_{i>0, j>0}\mathcal{E}_v(i, j)$. Then the following lemmas hold true.

LEMMA 2. *On a line-graph, for any $i, j > 0$, we have*

$$\mathbb{E}[\max_{x \in \mathcal{X}} \mathbb{P}(X_v = x|\mathbf{S}, \Gamma(V_A), \mathcal{E}_v(i, 0))|\mathcal{E}_v(i, 0)] \leq \frac{1}{i+1}$$

$$\mathbb{E}[\max_{x \in \mathcal{X}} \mathbb{P}(X_v = x|\mathbf{S}, \Gamma(V_A), \mathcal{E}_v(0, j))|\mathcal{E}_v(0, j)] \leq \frac{1}{j+1}$$

$$\mathbb{E}[\max_{x \in \mathcal{X}} \mathbb{P}(X_v = x|\mathbf{S}, \Gamma(V_A), \mathcal{E}_v(0, 0))|\mathcal{E}_v(0, 0)] \leq 1$$

$$\mathbb{E}[\max_{x \in \mathcal{X}} \mathbb{P}(X_v = x|\mathbf{S}, \Gamma(V_A), \mathcal{I}_v)|\mathcal{I}_v] \leq \frac{1}{n(1-3p)}. \qquad (40)$$

PROOF. Consider a realization G of the network topology such that our desired event $\mathcal{E}_v(i, 0)$ happens. In such a graph

G, the node succeeding v is an adversarial node and the i nodes preceding v are honest. Let us denote this set of $i+1$ nodes – comprising of the i nodes preceding v and v itself – as W_v (i.e., the ward of v). Now, if the messages assigned to the nodes outside of W_v is denoted by $X(V_H \backslash W_v)$, then for any $x \in S_v$ we have $\mathbb{P}(X_v = x|G, \mathbf{S}, \Gamma(V_A), \mathcal{E}_v(i, 0), X(V_H \backslash W_v)$

$$= \frac{\mathbb{P}(X_v = x, \mathbf{S}, X(V_H \backslash W_v)|G, \Gamma(V_A), \mathcal{E}_v(i, 0))}{\sum_{x \in S_v} \mathbb{P}(X_v = x, \mathbf{S}, X(V_H \backslash W_v)|G, \Gamma(V_A), \mathcal{E}_v(i, 0))}$$

$$= \frac{\mathbb{P}(X_v = x, X(V_H \backslash W_v)|G, \Gamma(V_A), \mathcal{E}_v(i, 0))}{\sum_{x \in S_v} \mathbb{P}(X_v = x, X(V_H \backslash W_v)|G, \Gamma(V_A), \mathcal{E}_v(i, 0))}$$

$$= \frac{1}{i+1}, \qquad (41)$$

by using the fact that the allocation of messages $\mathbf{X}$ is independent of the graph structure $(G, \Gamma(V_A), \mathcal{E}_v(i, 0))$ and $\mathbb{P}(\mathbf{S}|X_v = x, X(V_H \backslash W_v), G, \Gamma(V_A), \mathcal{E}_v(i, 0)) = 1$ on a linegraph. Now, taking expectation on both sides of Equation (41) we get

$$\mathbb{P}(X_v = x|\mathbf{S}, \Gamma(V_A), \mathcal{E}_v(i, 0)) = \frac{1}{i+1} \ \forall x \in S_v$$

$$\Rightarrow \max_{x \in \mathcal{X}} \mathbb{P}(X_v = x|\mathbf{S}, \Gamma(V_A), \mathcal{E}_v(i, 0)) = \frac{1}{i+1} \text{ or}$$

$$\mathbb{E}[\max_{x \in \mathcal{X}} \mathbb{P}(X_v = x|\mathbf{S}, \Gamma(V_A), \mathcal{E}_v(i, 0))|\mathcal{E}_v(i, 0)] = \frac{1}{i+1}. \qquad (42)$$

By a similar argument as above, we can also show that

$$\mathbb{E}[\max_{x \in \mathcal{X}} \mathbb{P}(X_v = x|\mathbf{S}, \Gamma(V_A), \mathcal{E}_v(0, j))|\mathcal{E}_v(0, j)] = \frac{1}{j+1},$$

$$\mathbb{E}[\max_{x \in \mathcal{X}} \mathbb{P}(X_v = x|\mathbf{S}, \Gamma(V_A), \mathcal{E}_v(0, 0))|\mathcal{E}_v(0, 0)] = 1. \qquad (43)$$

Finally let us consider the case where v is an interior node, i.e., event $\mathcal{I}_v$ happens. As before, for a head-node u (an honest node whose successor is an adversarial node) let W_u denote the ward containing u. Notice that under observations $\mathbf{S}, \Gamma(V_A)$ the adversaries know (i) the head and tail nodes of each ward (from $\Gamma(V_A)$) and (ii) the size of each ward ($|W_u| = |S_u|$). Therefore if a message x is such that $x \in S_u$ for some u, then

$$\mathbb{P}(X_v = x|\mathbf{S}, \Gamma(V_A), \mathcal{I}_v) = \mathbb{P}(X_v = x, v \in W_u|\mathbf{S}, \Gamma(V_A), \mathcal{I}_v)$$
$$= \mathbb{P}(v \in W_u|\mathbf{S}, \Gamma(V_A), \mathcal{I}_v)\mathbb{P}(X_v = x|v \in W_u, \mathbf{S}, \Gamma(V_A), \mathcal{I}_v)$$
$$= \frac{|W_u| - 2}{|I|}\frac{1}{|W_u|} \leq \frac{1}{|I|} \leq \frac{1}{n(1-3p)}, \qquad (44)$$

where I denotes the set of all interior nodes and $|I| \geq n(1 - 3p)$ since each adversary is a neighbor to at most 2 honest server nodes. Hence we have

$$\mathbb{E}[\max_{x \in \mathcal{X}} \mathbb{P}(X_v = x|\mathbf{S}, \Gamma(V_A), \mathcal{I}_v)|\mathcal{I}_v] \leq \frac{1}{n(1-3p)}, \qquad (45)$$

concluding the proof. $\square$

LEMMA 3. *On a line-graph, for $i, j > 0$ we have*

$$\mathbb{P}(\mathcal{E}_v(i, 0)) \leq \left(p + \frac{1}{n}\right)^2\left(1 - p + \frac{2}{n}\right)^i \qquad (46)$$

$$\mathbb{P}(\mathcal{E}_v(0, j)) \leq \left(p + \frac{1}{n}\right)^2\left(1 - p + \frac{2}{n}\right)^j \qquad (47)$$

$$\mathbb{P}(\mathcal{E}_v(0, 0)) \leq (p + 1/n)^2 \qquad (48)$$

$$\mathbb{P}(\mathcal{I}_v) \leq (1 - p)^2. \qquad (49)$$

PROOF. First let us consider the event $\mathcal{E}_v(i,0)$ in which node v has an adversarial successor, i honest predecessor nodes and an adversarial $i+1$-th predecessor. Let Y_v denote the position of node v in the line graph. Then

$$\mathbb{P}(\mathcal{E}_v(i,0)) = \sum_{j=i+1}^{n} \mathbb{P}(Y_v = j)\mathbb{P}(\mathcal{E}_v(i,0)|Y_v = j), \quad (50)$$

since v needs to be at a position on the line graph where at least $i+1$ predecessors are feasible. Now, for $i+1 \leq j \leq n$, by a simple counting argument we have $\mathbb{P}(\mathcal{E}_v(i,0)|Y_v = j) =$

$$\left(\frac{np}{n-1}\right)\left(\frac{np-1}{n-2}\right)\left(\frac{\tilde{n}-1}{n-3}\right)\left(\frac{\tilde{n}-2}{n-4}\right)\cdots\left(\frac{\tilde{n}-i}{n-i-2}\right)$$
$$\leq \left(p+\frac{1}{n}\right)^2\left(1-p+\frac{2}{n}\right)^i$$

Combining the above inequality with Equation (50) we conclude that

$$\mathbb{P}(\mathcal{E}_v(i,0)) \leq \left(p+\frac{1}{n}\right)^2\left(1-p+\frac{2}{n}\right)^i \quad (51)$$

for $i > 0$. By essentially a similar counting as above, we can also obtain the remaining Equations (47), (48) and (49) from the Lemma. $\square$

LEMMA 4. *If $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_k$ is a set of mutually exclusive and exhaustive events, and $v \in V_H$ is any honest server node, then*

$$\mathbf{D}_{\text{OPT}}(v) \leq \sum_{i=1}^{k} \mathbb{P}(\mathcal{E}_i)\mathbb{E}[\max_{x\in\mathcal{X}}\mathbb{P}(X_v = x|\mathbf{S},\Gamma(V_A),\mathcal{E}_i)|\mathcal{E}_i]. \quad (52)$$

PROOF. The proof is straightforward and follows from Corollary 1. From Equation (8) we have

$$\mathbb{E}[D_{\text{OPT}}(v)|\mathbf{S},\Gamma(V_A)] \leq \max_{x\in\mathcal{X}}\mathbb{P}(X_v = x|\mathbf{S},\Gamma(V_A))$$

$$= \max_{x\in\mathcal{X}}\sum_{i=1}^{k}\mathbb{P}(\mathcal{E}_i|\mathbf{S},\Gamma(V_A))\mathbb{P}(X_v = x|\mathbf{S},\Gamma(V_A),\mathcal{E}_i)$$

$$\leq \sum_{i=1}^{k}\mathbb{P}(\mathcal{E}_i|\mathbf{S},\Gamma(V_A))\max_{x\in\mathcal{X}}\mathbb{P}(X_v = x|\mathbf{S},\Gamma(V_A),\mathcal{E}_i). \quad (53)$$

Taking expectation on both sides of the above equation, we get $\mathbf{D}_{\text{OPT}}(v) \leq \mathbb{E}[\max_{x\in\mathcal{X}}\mathbb{P}(X_v = x|\mathbf{S},\Gamma(V_A),\mathcal{E}_i)]$,

$$\Rightarrow \mathbf{D}_{\text{OPT}}(v) \leq \sum_{i=1}^{k}\mathbb{P}(\mathcal{E}_i)\mathbb{E}[\max_{x\in\mathcal{X}}\mathbb{P}(X_v = x|\mathbf{S},\Gamma(V_A),\mathcal{E}_i)|\mathcal{E}_i],$$

and thus proving the Lemma. $\square$

To complete the proof of the Theorem, let use Lemma 4 with $\mathcal{E}_v(i,0), \mathcal{E}_v(0,j), \mathcal{E}_v(0,0)$ and $\mathcal{E}_v$ for $i,j > 0$ as the set of mutually exclusive and exhaustive events. Then the expected payoff at v can be bounded as $\mathbf{D}_{\text{OPT}}(v) \leq$

$$\sum_{i>0}\mathbb{P}(\mathcal{E}_v(i,0))\mathbb{E}[\max_{x\in\mathcal{X}}\mathbb{P}(X_v = x|\mathbf{S},\Gamma(V_A),\mathcal{E}_v(i,0))|\mathcal{E}_v(i,0)]$$

$$+\sum_{j>0}\mathbb{P}(\mathcal{E}_v(0,j))\mathbb{E}[\max_{x\in\mathcal{X}}\mathbb{P}(X_v = x|\mathbf{S},\Gamma(V_A),\mathcal{E}_v(0,j))|\mathcal{E}_v(0,j)]$$

$$+\mathbb{P}(\mathcal{E}_v(0,0))\mathbb{E}[\max_{x\in\mathcal{X}}\mathbb{P}(X_v = x|\mathbf{S},\Gamma(V_A),\mathcal{E}_v(0,0))|\mathcal{E}_v(0,0)]$$

$$+\mathbb{P}(\mathcal{I}_v)\mathbb{E}[\max_{x\in\mathcal{X}}\mathbb{P}(X_v = x|\mathbf{S},\Gamma(V_A),\mathcal{I}_v)|\mathcal{I}_v],$$
$$(54)$$

where the values of the individual expectation and probability terms in the above Equation (54) have been computed in Lemmas 2 and 3 respectively. Using those bounds, we get

$$\mathbf{D}_{\text{OPT}}(v) \leq \sum_{i>0}\left(p+\frac{1}{n}\right)^2\left(1-p+\frac{2}{n}\right)^i\frac{1}{i+1}$$

$$+\sum_{j>0}\left(p+\frac{1}{n}\right)^2\left(1-p+\frac{2}{n}\right)^j\frac{1}{i+1}$$

$$+(p+1/n)^2 + (1-p)^2\frac{1}{n(1-3p)}$$

$$\leq \frac{2(p+\frac{1}{n})^2}{(1-p+\frac{2}{n})}\log\left(\frac{1}{p-\frac{2}{n}}\right) + \frac{(1-p)^2}{n(1-3p)}$$

$$\leq \frac{2p^2}{1-p}\log\left(\frac{2}{p}\right) + O\left(\frac{1}{n}\right). \quad (55)$$

Finally averaging the expected payoff $\mathbf{D}_{\text{OPT}}(v)$ over each of the $\tilde{n}$ honest server nodes $v \in V_H$, we get the desired result.

A.4 Section 6: Systems Issues

A.4.1 Proof of Proposition 6

We map the problem of constructing a line (i.e., a 2-regular digraph) to one of assigning balls to bins. Suppose each ball represents an outgoing connection, and each bin represents a server who may accept that outgoing connection. There are n balls and n bins; for the sake of simplicity, we assume that a server can establish a connection to itself, so all n bins are available to each ball. Then the maximum degree d_m of the degree distribution is linearly related to the maximum number of balls in any bin h_m. That is, $d_m = 1+h_m$. When $k = 1$, each ball is assigned to a bin uniformly. The quantity h_m has been studied extensively in this case, and the result for $k = 1$ in Proposition 6 is well-known [24]. When $k > 1$, Algorithm 2 exploits the 'power of two choices' paradigm. Power of two choices states that by picking the minimum-degree node among two choices, the maximum in-degree $h_m = \frac{\log\log n}{\log 2}(1+o(1))+\Theta(1)$ with high probability [4]. This is an exponential reduction in maximum degree compared to when $k = 1$. More generally, for arbitrary $k > 1$, the maximum degree is $\frac{\log\log n}{\log k}(1+o(1))+\Theta(1)$. This result is due to Azar et al. [4] and is well-studied in subsequent literature [33].