TwistedPair: Towards Practical Anonymity in the Bitcoin P2P Network

Abstract—Recent work has demonstrated significant anonymity vulnerabilities in Bitcoin’s networking stack. In particular, the current mechanism for broadcasting Bitcoin transactions allows third-party observers to link transactions to the IP addresses that originated them. This lays the groundwork for low-cost, large-scale deanonymization attacks. In this work, we present TwistedPair, a first-principles, theoretically-justified defense against large-scale deanonymization attacks. TwistedPair is lightweight, scalable, and completely interoperable with the existing Bitcoin network. We evaluate TwistedPair through experiments on Bitcoin’s mainnet to demonstrate its interoperability with the current network, as well as low broadcast latency overhead.

I. INTRODUCTION

Anonymity is an important property for a financial system, especially given the often-sensitive nature of transactions [15]. Unfortunately, the anonymity protections in Bitcoin and similar cryptocurrencies can be fragile. This is largely because Bitcoin users are identified by cryptographic pseudonyms (a user can have multiple pseudonyms). When a user Alice wishes to transfer funds to another user Bob, she generates a transaction message that includes Alice’s pseudonym, the quantity of funds transferred, the prior transaction from which these funds are drawn, and a reference to Bob’s pseudonym [35]. The system-wide sequence of transactions is recorded in a public, append-only ledger known as the blockchain. The public nature of the blockchain means that users only remain anonymous as long as their pseudonyms cannot be linked to their true identities.

This mandate has proved challenging to uphold in practice. Several vulnerabilities have enabled researchers, law enforcement, and possibly others to partially deanonymize users [12]. Publicized attacks have so far included: (1) linking different public keys that belong to the same user [31], (2) associating users’ public keys with their IP addresses [10], [28], and in some cases, (3) linking public keys to human identities [21]. Such deanonymization exploits tend to be cheap, easy, and scalable [31], [10], [28].

Although researchers have traditionally focused on the privacy implications of the blockchain [36], [41], [31], we are interested in lower-layer vulnerabilities that emerge from Bitcoin’s peer-to-peer (P2P) network. Recent work has demonstrated P2P-layer anonymity vulnerabilities that allow transactions to be linked to users’ IP addresses with accuracies up to 30% [10], [28]. Understanding how to patch these vulnerabilities without harming utility remains an open question. The goal of our work is to propose a practical, lightweight modification to Bitcoin’s networking stack that provides theoretical anonymity guarantees against the types of attacks demonstrated in [10], [28], and others. We begin

Fig. 1: Supernodes can observe flooding metadata to infer which node was the source of a transaction message (t×).

with an overview of Bitcoin’s P2P network, and explain why it enables deanonymization attacks.

A. Bitcoin’s P2P Network

Bitcoin nodes are connected over a P2P network of TCP links. This network is used to communicate transactions, the blockchain, and control packets, and it plays a crucial role in maintaining the network’s consistency. Each peer is identified by its (IP address, port) combination. Whenever a node generates a transaction, it broadcasts a record of the transaction over the P2P network; critically, transaction messages do not include the sender’s IP address—only their pseudonym. Since the network is not fully-connected, transactions are relayed according to epidemic flooding [37]. This ensures that all nodes receive the transaction and can add it to the blockchain. Hence, the broadcasting of transactions enables the network to learn about transactions quickly and reliably.

However, the broadcasting of transactions can also have negative anonymity repercussions. The statistical symmetry of Bitcoin’s current broadcasting mechanism spreads content isotropically over the graph; this allows adversarial peers who observe the spreading dynamics of a given transaction to infer the source IP of each transaction. For example, in recent attacks [10], [28], researchers launched a supernode that connected to all nodes in the P2P network (Figure 1), and logged all observed traffic. This allowed the supernode to observe the spread of each transaction over the network over time, and ultimately infer the source IP. Since transaction messages include the sender’s pseudonym, the supernodes were able to deanonymize users, or link their pseudonyms to an IP address [10], [28]. Such deanonymization attacks are problematic because of Bitcoin’s transparency: once a user is deanonymized, her other transactions can often be linked, even if she creates fresh pseudonyms for each transaction [31].

There have been recent proposals for mitigating these vulnerabilities, including broadcasting protocols that reduce the symmetry of epidemic flooding. Bitcoin Core [5], the most commonly-used Bitcoin implementation, adopted a protocol called diffusion, where each node spreads transactions with independent, exponential delays to its neighbors on the P2P graph. Diffusion is still in use today. However, proposed
solutions (including diffusion) tend to be heuristic, and recent work shows that they do not provide sufficient anonymity protection [19]. Other proposed solutions offer theoretical anonymity guarantees [13], but do so under idealistic assumptions that are unlikely to hold in practice. The aim of this work is to propose a broadcasting mechanism that (a) provides provable anonymity guarantees under realistic adversarial and network assumptions, and (b) does not harm the network’s broadcasting robustness or latency. We do this by revisiting the DANDELION system proposed in [13], and redesigning it to withstand a variety of practical threats.

B. Contributions

The main contributions of this paper are threefold:

(1) We identify several key idealistic assumptions made by DANDELION [13], and show how anonymity is degraded when those assumptions are violated. In particular, [13] assumes an honest-but-curious adversary that has limited knowledge of the P2P graph topology and only observes one transaction per node. If adversaries are instead malicious and collect more information over time, we show that they are able to arbitrarily weaken the anonymity guarantees of [13] through a combination of attacks, including side information, graph manipulation, black hole, and intersection attacks.

(2) We propose a new protocol called TWISTEDPAIR\(^1\) that subtly changes most of the implementation choices of DANDELION, from the graph topology to the randomization mechanisms for message forwarding. Mathematically, these (relatively small) algorithmic changes completely change the anonymity analysis by exponentially augmenting the problem state space. Using analytical tools from the theory of Galton-Watson trees and random processes over graphs, we show that TWISTEDPAIR offers significant anonymity gains over DANDELION, both theoretically and in simulation, when adversaries have stronger capabilities. TWISTEDPAIR is currently being considered for integration in Bitcoin Core.\(^2\)

(3) We demonstrate the practical feasibility of TWISTEDPAIR by evaluate an implementation on Bitcoin’s mainnet. We show that TWISTEDPAIR does not increase latency significantly compared to current methods for broadcasting transactions, and it is robust to node failures and misbehavior.

The paper is structured as follows: in [II] we discuss relevant work on anonymity in cryptocurrencies and P2P networks. In [III] we present our adversarial model, which is based on prior attacks in the literature. [IV] presents DANDELION in more detail; in [V] we analyze DANDELION’s weaknesses, and propose TWISTEDPAIR as an alternative. We present experimental evaluation results for TWISTEDPAIR in [VI] and discuss the implications of these results in [VII].

II. RELATED WORK

The anonymity properties of cryptocurrencies have been studied extensively. Several papers have exploited anonymity vulnerabilities in the blockchain [7], [39], [36], [41], [31].

\(^1\)The name TWISTEDPAIR is derived from a twisted pair cable, which resembles the proposed spreading pattern.

\(^2\)Bitcoin Core is a Bitcoin wallet used by over 70% of active nodes. The name TWISTEDPAIR has been altered for anonymous review.

suggesting that transactions by the same user can be linked, even if the user adopts different addresses [31]. In response, researchers proposed alternative cryptocurrencies and/or tumblers that provide anonymity at the blockchain level [29], [4], [43], [42], [25], [26]. It was not until 2014 that researchers turned to the P2P network, showing that regardless of blockchain implementation, users can be deanonymized by network attackers [29], [10], [11], [19]. In some cases researchers were able to link transactions to IP addresses with accuracies up to 30% [10]. These attacks proceeded by connecting a supernode to a majority of active Bitcoin server nodes. More recently, [8] demonstrated the serious anonymity and routing risks posed by an AS-level attacker. These papers suggest a need for networking protocols that defend against deanonymization attacks.

Anonymous communication for P2P/overlay networks has been an active research topic for decades. Most work relies on two ideas: randomized routing (e.g., onion routing, Chaumian mixes) and/or dining cryptographer (DC) networks. Systems that use DC nets [14] are typically designed for broadcast communication, which is our application of interest. However, DC nets are known to be inefficient and brittle [24]. Proposed systems [23], [49], [16], [48] improved these properties significantly, but DC networks never became scalable enough to enjoy widespread adoption in practice.

Systems based on randomized routing are generally more efficient, but focus on point-to-point communication (though these tools can be adapted for broadcast communication). Early works like Crowds [40], Tarzan [20], and P5 [45] paved the way for later practical systems, such as Tor [18] and I2P [50], as well as recent proposals like Drac [17], Pisces [34], and Vuvuzela [46]. Our work differs from this body of work along two principal axes: (1) usage goals, and (2) analysis metrics/results.

(1) Usage goals. Among tools with real-world adoption, Tor [18] is the most prominent; privacy-conscious Bitcoin users frequently use it to anonymize transmissions. However, Tor requires users to route their network traffic through a third-party service; this is neither a scalable nor adequate solution for users who are unaware of Bitcoin’s privacy vulnerabilities. Our goal is instead to propose internal solutions that can be implemented within the cryptocurrency itself. In this vein, I2P is an onion-routing overlay that is implemented within P2P networks [50]; the altcoin Monero currently is integrating I2P with its network. While this may be a workable long-term solution, cryptographic routing protocols are complex and difficult to implement correctly; Monero has been implementing I2P for at least four years [3] with no clear end in sight [6], and none of the other major cryptocurrency wallets (Bitcoin, Ethereum, Ripple) have announced plans to integrate anonymized routing. Hence, there is a need for simple, lightweight solutions.

(2) Analysis. The differences in analysis are more subtle. Many of the above systems include theoretical analysis, but none provide optimality guarantees. Moreover, they analyze per-user metrics, such as probability of linkage [40], [20], [46]. This metric overlooks the fact that adversaries often use data from many users to execute joint deanonymization. We are interested in the (mathematically more complex) problem of population-level deanonymization using all data available to the adversary. Finally, the protocols of prior work may...
appear similar, but the corresponding anonymity analyses and guarantees are significantly different.

The most relevant solution to our problem is a recent proposal called DANDELION \cite{13}, which uses statistical obfuscation to provide anonymity against distributed, resource-limited adversaries (pseudocode in Appendix \ref{app}). DANDELION propagates transactions in two phases: (i) an anonymity (or stem) phase, and (ii) a spreading (or fluff) phase. In the anonymity phase, each message is passed to a single, randomly-chosen neighbor in an anonymity graph $H$ (this graph can be an overlay of the P2P graph $G$). This propagation continues for a geometric number of hops with parameter $q$. However, unlike prior work, different users forward their transactions along the same anonymity graph $H$, which is chosen as a directed cycle in $G$. In the spreading phase, messages are rapidly flooded over the P2P network $G$ via diffusion, just as in today’s Bitcoin network. DANDELION periodically re-randomizes the line graph, so the adversaries’ knowledge of the graph is assumed to be limited to their immediate neighborhood.

Under these conditions, DANDELION exhibits near-optimal anonymity guarantees under a joint-deanonymization model \cite{13}. As mentioned earlier, DANDELION's guarantees are based on very restrictive adversarial assumptions. Our work illustrates DANDELION's fragility to basic Byzantine attacks and proposes a scheme that is robust to Byzantine intersection attacks. This relaxation requires completely new analysis, which is the theoretical contribution of this paper.

III. MODEL

A. Adversary

The adversaries studied in prior work exhibit two basic capabilities: creating nodes and creating outbound connections to other nodes. At one extreme, a single supernode can establish outbound connections to every node in the network; this resembles recent attacks on the Bitcoin P2P network \cite{10, 28} and related measurement tools \cite{27, 33}. At the other extreme is a botnet with many honest-but-curious nodes, each of which creates few outbound edges according to protocol. This captures the adversarial model in \cite{13} and botnets observed in Bitcoin’s P2P network \cite{30}. In this paper, we combine both models: a botnet adversary that can corrupt some fraction of Bitcoin nodes and establish arbitrarily many outbound connections.

We model the botnet adversary as a set of colluding hosts spread over the network. Out of $n$ total peers in the network, we assume a fraction $p$ (i.e., $np$ peers) are malicious. The botnet seeks to link transactions and their associated public keys with the IP addresses of the hosts generating these transactions. The adversarial hosts (or spies) need not follow protocol. Spies can generate as many outbound edges as they want, to whichever nodes they choose; however, they cannot force honest nodes to create outbound edges to spies. The spies perform IP address deanonymization by observing the transaction propagation patterns in the network. Adversaries log transaction information including, timestamps, sending hosts, and other control packets. This information, together with global knowledge (e.g., network structure) is used to deanonymize honest users.\footnote{Honest users are Bitcoin hosts that are not part of the adversarial botnet. We assume honest users follow the specified protocols without deviations.}

We assume adversaries are interested in simultaneously deanonymizing all the users in the network from observed information. This is distinct from a setting where adversaries seek to deanonymize specific users. While the latter is a more common, well-studied problem \cite{12, 11, 7}, typical solutions tend to require hosts to change their behavior, e.g., by adopting a new cryptocurrency. Our goal is to provide a (weaker) network-wide anonymity that does not require users to change behavior. While this approach will not stop targeted attacks, it does provide a first line of defense against broad deanonymization attacks that are currently eminently feasible.

Note that recent work on ISP- or AS-level adversaries \cite{8} can be modeled as a special case of this botnet adversary, except edges rather than nodes are corrupted. This adversary is outside the scope of this paper, but the topic is of great interest, and we expect that the intuitions developed in this paper may be useful for defending against such an adversary. In §VI we discuss the compatibility of our proposed methods with the countermeasures proposed in \cite{8} for large-scale adversaries.

B. Anonymity Metrics: Precision and Recall

The adversary’s goal is to associate transactions with users’ IP addresses through some association map. This association map can be interpreted as a classifier that classifies each transaction (and its corresponding metadata) to an IP address. Hence an adversary’s deanonymization capabilities can be measured by evaluating the adversary’s associated classifier. We adopt a common metric for classifiers: precision and recall.

Let $\mathcal{V}_H$ denote the set of all IP addresses of honest peers in the network. To begin, we will assume that each peer in $v \in \mathcal{V}_H$ performs exactly one transaction $X_v$. We relax this assumption in §V-B. Let $\mathcal{X} = \bigcup_{v \in \mathcal{V}_H} X_v$ denote the set of all transactions. Assuming $\mathcal{V}_H$ is known to the adversary, let $\mathfrak{M} : \mathcal{X} \rightarrow \mathcal{V}_H$ denote the adversary’s map from transaction $x \in \mathcal{X}$ to IP address $\mathfrak{M}(x) \in \mathcal{V}_H$. The precision and recall of $\mathfrak{M}$ at any honest peer $v \in \mathcal{V}_H$ are given, respectively, by

\[
D(v) = \frac{\mathbb{I}(\mathfrak{M}(X_v) = v)}{\sum_{u \in \mathcal{V}_H} \mathbb{I}(\mathfrak{M}(X_u) = v)} \quad (1)
\]

\[
R(v) = \frac{\mathbb{I}(\mathfrak{M}(X_v) = v)}{\mathbb{I}(\mathfrak{M}(X_v) = v)} \quad (2)
\]

where $\mathbb{I}(\cdot)$ denotes the indicator function. Precision (denoted $D(v)$) measures accuracy by normalizing against the number of transactions associated with $v$. A large number of transactions mapped to $v$ implies a greater plausible deniability for $v$. Recall (denoted $R(v)$) measures the accuracy or completeness of the mapping. We define the average precision and recall for the network as $D = \frac{\sum_{v \in \mathcal{V}_H} D(v)}{|\mathcal{V}_H|}$ and $R = \frac{\sum_{v \in \mathcal{V}_H} R(v)}{|\mathcal{V}_H|}$. In our theoretical analyses under a probabilistic model (see §III-C), we will be interested in the expected values of these quantities, denoted by $\mathbf{D}$ and $\mathbf{R}$, respectively.

C. Transaction and Network Model

We follow the probabilistic network model of \cite{13} §2. Let $\mathcal{V}_H$ denote the set of IP addresses of current honest
Algorithm 1: APPROXIMATE $2\eta$-REGULAR GRAPH

\textbf{Input:} Set $V = \{v_1, v_2, \ldots, v_n\}$ of nodes;
\textbf{Output:} A connected, directed graph $G(V,E)$ with average degree $2\eta$

\begin{algorithmic}
\State for $v \leftarrow V$ do
\Comment{pick $\eta$ random targets}
\State $N \leftarrow \emptyset$
\For{$i \leftarrow \{1, \ldots, \eta\}$}
\State $e \sim \text{Unif}(V \setminus \{v\} \setminus N)$
\State $N \leftarrow N \cup \{e\}$
\EndFor
\Comment{make connections}
\State $E = E \cup \{(v \rightarrow u), u \in N\}$
\EndFor
\State return $G(V,E)$
\end{algorithmic}

users in the network, $X_v$ the message from user $v \in V_H$ and $\mathcal{X}$ the set of all transaction messages. Let $\tilde{n} = |V_H|$ denote the number of honest peers; let the sets $V_H$ and $\mathcal{X}$ be known to the adversaries. We assume a uniform prior on $X_v$ over the set $\mathcal{X}$, i.e., the ordered tuple $(X_{v_1}, X_{v_2}, \ldots, X_{v_n})$ is a uniform random permutation of messages in $\mathcal{X}$ where $V_H = \{v_1, v_2, \ldots, v_n\}$. We also assume transaction times are unknown to the adversary.

We model the Bitcoin network as a directed graph $G(V,E)$ where the vertices $V = V_H \cup V_A$ comprise honest peers $V_H$ and adversarial peers $V_A$. The edges $E$ correspond to TCP links between peers in the network. Although these links are technically bidirectional, the Bitcoin network treats them as directed. An \textit{outbound} link from Alice to Bob is one that Alice initiated (and vice versa for inbound links); we also refer to the tail node of an edge as the one that originated the connection.

To construct the Bitcoin network, each node establishes up to eight outbound connections, and maintains up to 125 total connections \cite{4}. In practice, the eight outbound connections are chosen from each node’s locally-maintained address book; we assume each node chooses outbound connections randomly from the set of all nodes (Algorithm \ref{alg:random_graph}). This graph construction model results in a random graph where each node has expected degree $2\eta = 16$. Although this approximates the behavior of many nodes in Bitcoin’s P2P network, Byzantine nodes need not follow protocol. We will describe the behavior of Byzantine nodes as needed in the paper.

Upon creating a transaction message, peers propagate it according to a pre-specified spreading policy. The propagation dynamics are observed by the spies, whose goal is to estimate the IP addresses of transaction sources. For each transaction $x \in \mathcal{X}$ received by adversarial node $a \in V_A$, the tuple $(x,v,t)$ is logged where $v$ is the peer that sent the message to $a$, and $t$ is the timestamp when the message was received. The botnet adversary may also know partial information about the network structure. Clearly, peers neighboring adversarial nodes are known. However in some cases the adversary might also be able to learn the locations of honest peers not directly connected to botnet nodes. For simplicity, we use $O$ to denote all observed information—message timestamps, knowledge of the graph, and any other control packets—known to the adversary. Given these observations, one common source estimator is the simple-yet-robust \textit{first-spy estimator}, used in \cite{10, 28}. The first-spy estimator outputs the first honest node to deliver a given transaction to the adversary as the source.

\section{TWISTEDPAIR}

\textsc{Dandelion}’s theoretical anonymity guarantees only hold under three, idealized assumptions: (1) all nodes obey protocol, (2) each node generates exactly one transaction, (3) all Bitcoin nodes run \textsc{Dandelion}. None of these assumptions necessarily holds in practice. In this section, we show how \textsc{Dandelion}’s anonymity properties break when the assumptions are violated, and propose a new solution called \textsc{TwistedPair} that addresses these concerns. \textsc{TwistedPair} passes transactions over intertwined paths, or ‘cables’, before diffusing to the network (Fig. \ref{fig:twistedpair}).

This section is structured according to Table \ref{tab:threats}. The weak adversarial model in \cite{13} enables five distinct attacks: graph learning attacks, intersection attacks, graph construction attacks, black hole attacks, and deployment attacks. For each attack, we first demonstrate its impact on anonymity (and/or robustness); in many cases, these effects can lead to arbitrarily high deanonymization accuracies. Next, we propose lightweight implementation changes to mitigate this threat, and justify these choices with theoretical analysis and simulations. We adopt the same ‘stem phase’ and ‘fluff phase’ terminology as \textsc{Dandelion}, along with dandelion spreading (Algorithm \ref{alg:spread}). The main algorithmic changes in \textsc{TwistedPair} are:

\begin{itemize}
\item 1) use a random $4$-regular graph instead of a line graph for the anonymity phase (§\ref{sec:line_graph});
\item 2) use pseudorandom forwarding rather than independent, randomized forwarding (§\ref{sec:randomized});
\item 3) use open-loop graph construction- and maintenance mechanisms (§\ref{sec:openloop} and §\ref{sec:openloop}).
\end{itemize}

Perhaps surprisingly, these small algorithmic changes completely alter the anonymity analysis by introducing an exponentially-growing state space. For example, moving from a line graph to a $4$-regular graph (item (1)) invalidates the exact probability computation in \cite{13}, and requires a more sophisticated analysis over Galton-Watson branching processes to understand effects like intersection attacks. We also simulate the proposed mechanisms for all attacks and find improved anonymity compared to \textsc{Dandelion}.

\subsection{Graph-Learning Attacks}

Theoretical results in \cite{13} assume that the anonymity graph is unknown to the adversary; that is, the adversary knows the randomized graph construction protocol, but it does not

\footnote{Simulation code will be released; URL redacted for anonymous review.}
We propose an alternative solution that inherently protects against adversaries that are able to learn the anonymity graph—either due to the TWISTED PAIR protocol itself or other implementation issues. In particular, we suggest that TWISTED PAIR should use random, directed, d-regular graphs instead of line graphs as the anonymity graph topology. 4-regular graphs naturally extend line graphs, which are 2-regular. Although the forwarding mechanism will be revisited in § IV-B for now, users relay transactions randomly to one of their two outbound neighbors in the 4-regular graph until fluff phase.

Although theoretical analysis of expected precision on d-regular anonymity graphs is challenging for d > 2, we can simulate randomized spreading over different topologies, while measuring anonymity empirically using theoretically-optimal (or near-optimal) estimators. Figure 5 plots the average precision obtained on d-regular graphs, for d = 2 (corresponding to a line graph), 4 and 6, as a function of p, the fraction of adversaries, for a network of 50 nodes. The blue solid line at the bottom corresponds to the line graph when the graph is unknown to the adversary; this matches the theoretical precision of $O(p^2 \log(1/p))$ shown in [13]. The solid lines in Figure 5 (i.e., unknown graph) were generated by running the first-spy estimator, which maps each transaction to the first honest node that forwarded the transaction to an adversarial node. When the graph is unknown, we show in Theorem 1 that the first-spy estimator is close to optimal; hence, Figure 5 shows results from an approximately-precision-optimal estimator.

When the graph is known, we approximate the maximum-precision estimator differently. [13] showed that the precision-optimal estimator is a maximum-weight matching from transactions to nodes, where the weight of the edge between node $v$ and transaction $x$ is the posterior probability $\Pr(X_v = x | O)$ (Theorem 3, [13]). Since the anonymity graph contains cycles, this posterior is difficult to compute exactly, because it requires (NP-hard) enumeration of every path between a candidate source and the first spy to observe a given transaction [44]. We therefore approximate the posterior probabilities by assuming that each candidate source can only pass a given message along the shortest path to the spy that first observes the message (note that the shortest path is also the most likely one). This path likelihood can be computed exactly and used as a proxy for the desired posterior probability. Given these approximate likelihoods, we compute a maximum-weight matching, and calculate the precision of the resulting matching.

Assuming the adversary knows the graph and uses this quasi-precision-optimal estimator, the precision on a line graph increases to the blue dotted line at the top of the plot. For example, at $p = 0.15$, knowing the graph gives a precision boost of 0.12, or about 250%. On the other hand, if a 4-regular graph is unknown to the adversary, it has a precision very close to that of line graphs (orange solid line in Figure 3). But if the graph becomes known to the adversary (orange dotted line), the increase in precision is smaller. At $p = 0.15$, the gain is 0.06—half as large as the gain for line graphs. This suggests that 4-regular graphs are more robust than line graphs to adversaries learning the graph, while sacrificing minimal precision when the adversary does not know the graph.

Figure 3 also highlights a distinct trend in precision values.

### TABLE I: Summary of changes proposed in TWISTED PAIR, with references to relevant evidence and/or analysis.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph-learning attacks (§IV-A)</td>
<td>Order-level precision increase [13]</td>
<td>4-regular anonymity graph</td>
<td>Limits precision gain (Thm. 1, Fig. 4)</td>
</tr>
<tr>
<td>Intersection attacks (§IV-E)</td>
<td>Empirical precision increase (Fig. 5)</td>
<td>Pseudorandom forwarding</td>
<td>Robust to intersection attacks (Thm. 2)</td>
</tr>
<tr>
<td>Graph construction attacks (§IV-C)</td>
<td>Empirical precision increase (Fig. 5)</td>
<td>Non-interactive graph construction</td>
<td>Reduces precision gain (Fig. 6)</td>
</tr>
<tr>
<td>Black-hole attacks (§IV-D)</td>
<td>Transactions do not propagate</td>
<td>Random stem timers</td>
<td>Provides robustness (Prop. 4)</td>
</tr>
<tr>
<td>Partial deployment (§IV-E)</td>
<td>Arbitrary recall increase (Fig. 11)</td>
<td>Protocol-agnostic stem selection</td>
<td>Improves recall (Thm. 2, Fig. 4)</td>
</tr>
</tbody>
</table>
as the degree \( d \) varies. If the adversary has not learned the topology, then line graphs have the lowest expected precision and hence offer the best anonymity. As the degree \( d \) increases, the expected precision progressively worsens until \( d = n \), where \( n \) is the number of peers in the network. When \( d = n \) (i.e., the graph is a complete graph), peers forward their transactions to a random peer in each hop of the dandelion stem. On the other hand, if the topologies are known to the adversary, then the performance trend reverses. In this case, line graphs (for \( d = 2 \)) have the worst (highest) expected precision among random \( d \)-regular graphs; As the degree \( d \) varies. If the adversary has not learned the neighborhood upstream of \( v \), the best anonymity is offer by the maximum expected precision on a random \( d \)-regular graph of degree \( d = n \) ([32]). We use the following two lemmas, whose assumption follows from the locally-tree-like nature of sparse graphs. We use the following two lemmas, whose assumptions follow from the locally-tree-like nature of sparse graphs.

\[ \mathbb{E}[\max_{x \in \mathcal{X}} \mathbb{P}(X_v = x | S, \Gamma(V_A), F_u, E_v^{(h,h)}) | F_U, E_v^{(h,h)}] \leq \frac{1}{\mathbb{E}[1 + \max_{x \in \mathcal{X}} \mathbb{P}(X_v = x | S, \Gamma(V_A), F_u, E_v^{(h,h)})]} .\]

\[ \mathbb{E}[\max_{x \in \mathcal{X}} \mathbb{P}(X_v = x | S, \Gamma(V_A), F_u, E_v^{(h,h)}) | F_U, E_v^{(h,h)}] \leq \frac{1}{\mathbb{E}[1 + \max_{x \in \mathcal{X}} \mathbb{P}(X_v = x | S, \Gamma(V_A), F_u, E_v^{(h,h)})]} .\]

Finally, recall that our simulations used the ‘first-spy’ estimator as the deanonymization policy for \( d \)-regular graphs when the graph is unknown to the adversary. \( A \text{ priori} \), it is not clear whether this is an optimal estimator for \( d \)-regular graphs for \( d \geq 4 \) ([13]). Thus the precision gain from line graphs to \( d \)-regular graphs (for \( d = 4 \)) increases, precision decreases monotonically until \( d = n \).

**Theorem 1:** The maximum expected precision on a random \( 4 \)-regular graph with graph unknown to the adversary is bounded by \( D_{\text{OPT}} \leq 8D_{\text{PS}} + 6p^2 + O(p^3) \), where \( D_{\text{OPT}} \) and \( D_{\text{PS}} \) denote the expected precision under the optimal and first-spy estimators respectively.

**Proof:** For \( i, j, k, l \in \{a, h\} \), let \( E_v^{(i,j,k,l)} \) denote the event that \( v \)’s left successor, right successor, left predecessor and right predecessor are of types \( i, j, k \) and \( l \) respectively, where type \( a \) denotes an adversarial node and a type \( h \) denotes an honest node. Also, assume that for any honest node \( v \in V_H \) the number of messages forwarded by \( v \) is statistically independent of the local neighborhood upstream of \( v \). This assumption follows from the locally-tree-like nature of sparse random graphs ([32]). We use the following two lemmas, whose proofs are included in Appendix [B].

**Lemma 1:** Let \( J_v \) denote the number of transactions (from honest servers) that reach \( v \) before reaching an adversary. Then, for each event \( E \in \{E_v^{(a,a)}, E_v^{(h,h)}, E_v^{(a,h)}, E_v^{(h,a)}\} \)

\[ \mathbb{E}[\max_{x \in \mathcal{X}} \mathbb{P}(X_v = x | S, \Gamma(V_A), E_v^{(i,j,k,l)}, J_v) | E_v^{(i,j,k,l)}] \leq \frac{1}{\mathbb{E}[1 + \max_{x \in \mathcal{X}} \mathbb{P}(X_v = x | S, \Gamma(V_A), E_v^{(i,j,k,l)}, J_v) | E_v^{(i,j,k,l)}]} \]

**Lemma 2:** For any server \( v \in V_H \), let \( F_v \) denote the number of transactions that (i) reach \( v \) before reaching any adversary and (ii) are forwarded by \( v \) along its left outgoing edge. Then

\[ \mathbb{E}[\frac{1}{\mathbb{E}[1 + \max_{x \in \mathcal{X}} \mathbb{P}(X_v = x | S, \Gamma(V_A), E_v^{(i,j,k,l)}, J_v) | E_v^{(i,j,k,l)}]}] \leq 2D_{\text{PS}}(x) \]

Now, recall the events \( E_v^{(i,j,k,l)} \), where \( i, j, k, l \in \{a, h\} \). There are a total of \( 2^4 = 16 \) such events that are possible for the neighborhood around server \( v \). Out of these events, \( \binom{4}{2} = 6 \) of them occur with a probability of \( p^2(1-p)^2 \) (such as \( E_v^{(a,a)} \) for e.g.). Similarly \( \binom{4}{3} = 4 \) events occur with a probability of \( p^3(1-p) \) and one event occurs with a probability of \( p^4 \). Since the cost can be at most \( 1 \), the above events contribute to a cumulative cost of at most \( 6p^2 + 4p^3 + p^4 \).

The remaining cases are events where only one neighbor is adversarial—\( E_v^{(a,h)}, E_v^{(h,a)}, E_v^{(a,h)} \) and \( E_v^{(h,a)} \)—or when all of the neighbors are honest \( E_v^{(h,h)} \). Note that each of these events occur with a probability of at least \( p(1-p)^3 \) and hence the trivial bound used above cannot be used here. Let us first consider the event \( E_v^{(h,h)} \) where only the left predecessor node of \( v \) is an adversary. Let \( U \in V_H \) denote the right predecessor of \( v \) and \( F_U \) the number of fresh transactions that are forwarded by \( U \) to \( v \). Then from Lemma 1 we have

\[ \mathbb{E}[\max_{x \in \mathcal{X}} \mathbb{P}(X_v = x | S, \Gamma(V_A), F_v, E_v^{(h,h)} | F_U, E_v^{(h,h)})] \leq \frac{1}{\mathbb{E}[1 + \max_{x \in \mathcal{X}} \mathbb{P}(X_v = x | S, \Gamma(V_A), F_v, E_v^{(h,h)} | F_U, E_v^{(h,h)})]} \]

**Lemma 3:** For any server \( v \in V_H \), let \( F_v \) denote the number of transactions that (i) reach \( v \) before reaching any adversary and (ii) are forwarded by \( v \) along its left outgoing edge. Then

\[ \mathbb{E}[\frac{1}{\mathbb{E}[1 + \max_{x \in \mathcal{X}} \mathbb{P}(X_v = x | S, \Gamma(V_A), E_v^{(i,j,k,l)}, J_v) | E_v^{(i,j,k,l)}]}] \leq 2D_{\text{PS}}(x) \]

Finally consider \( E_v^{(h,h)} \), in which case \( v \)’s location is completely hidden from the adversaries. Let \( I \) be the set of such nodes. Since each adversary is a neighbor to at most 4 honest nodes, there are at least \( \tilde{n} - 4np = (1 - 5p)n \) nodes in \( I \). So \( \forall x \in \mathcal{X} \), we have

\[ \mathbb{E}[\max_{x \in \mathcal{X}} \mathbb{P}(X_v = x | S, \Gamma(V_A), G, I, E_v^{(h,h)}, J_v) | E_v^{(h,h)}] \leq \frac{1}{|I|} \leq \frac{1}{(1 - 5p)n} \]

and hence

\[ \mathbb{E}[\max_{x \in \mathcal{X}} \mathbb{P}(X_v = x | S, \Gamma(V_A), E_v^{(h,h)} | E_v^{(h,h)})] \leq \frac{1}{(1 - 5p)n} \]

Summing over all the cases considered gives the result.

This result says that precision under the first-spy estimator is within a constant factor of optimal \((p^2)\) is a fundamental lower bound on the maximum expected precision of any scheme ([13]). Thus the precision gain from line graphs to 4-regular graphs is indeed small, as Figure 5 suggests. These observations motivate the use of 4-regular graphs, specifically in lieu of higher-degree regular graphs. First, 4-regular graphs have similar precision to complete graphs when the graph is known (i.e., the red dotted line is close to the middle black solid line, which is a lower bound on precision for regular graphs), but they sacrifice minimal precision when the graph is unknown. Hence, they provide robustness to graph-learning.

**Lesson:** Use 4-regular graphs instead of line graphs for \( H \).

**B. Intersection Attacks**

The previous section showed that 4-regular graphs are robust to deanonymization attacks, even when the adversary knows the graph. However, those results assume that each user generates exactly one transaction per epoch. In this section, we relax the one-transaction-per-node assumption, and allow nodes to generate an arbitrary number of transactions. DANDELION specifies that each transaction should take an independent path over the anonymity graph. In our case, this implies that if a node generates multiple transactions, each one will traverse a random walk (of geometrically-distributed
length) over a 4-regular digraph. Under such a model, adversaries can aggregate metadata from multiple, linked transactions to launch intersection attacks. We first demonstrate Dandelion’s vulnerability to intersection attacks, and then provide an alternative propagation technique.

**Attack.** Suppose the adversary knows the graph. For each honest source \( v \in V_H \), each of its transactions will reach one of the \( np \) spy nodes first. In particular, each spy node has some fixed probability of being the first spy for transactions originating at \( v \) (given the graph). We let \( \Psi_v \) denote the pmf of the first spy for transactions starting at \( v \); the support of this distribution is the set of all spies. We hypothesize that in realistic graphs, for \( v \neq w \), \( \Psi_v \neq \Psi_w \).

This hypothesis suggests a natural attack, which consists of a training phase and a test phase. In the training phase, for each candidate source, the adversary simulates dandelion spreading \( N \) times. The resulting empirical distribution of first-spies for a given source determines the adversary’s estimate of \( \Psi_v \). The adversary computes such a signature for each candidate source.

At testing, the adversary gets to observe \( m \) transactions from a given node, \( m \ll N \). The adversary again computes the empirical distribution \( \hat{\Psi} \) of first-spies from those \( m \) observations. The adversary then classifies \( \hat{\Psi} \) to one of the \( |V_H| \) classes (i.e., source nodes) by matching \( \hat{\Psi} \) to the closest \( \Psi_i \) from training. For each trial, \( \Psi \) and \( \hat{\Psi} \) are matched by maximizing the likelihood of signatures (i.e. by minimizing the KL divergence).

Figure 5 shows the recall for such an attack on a 4-regular graph of size 1000 with fraction \( p = 0.3 \) of spies. The two trend lines correspond to signatures trained on \( N = 15000 \) and \( N = 35000 \) simulations. By observing 10 transactions per node, the recall exceeds 0.8. This suggests that independent random forwarding leads to serious intersection attacks. Hence, a naive implementation of Dandelion critically damages anonymity properties.

**Solution.** To address these attacks, we consider forwarding mechanisms with correlated randomness. The key insight is that messages from the same source should traverse the same path; this prevents adversaries from learning additional information from multiple transactions. However, a naive implementation (e.g., adding a tag that identifies transactions from the same source, and sending all such transactions over the same path) makes it trivial to link transactions, which is undesirable for other reasons. Hence, we consider three forwarding schemes that pseudorandomize the forwarding trajectory. In “one-to-one” forwarding, each node maps each of its inbound edges to a unique outbound edge; messages in stem mode only get relayed according to this mapping (Fig. 4). Each node also chooses exactly one outbound edge for all of its own transactions. The idea is that we are pseudorandomizing not by source, but by incoming edge (for relayed transactions). Similarly, “all-to-one” forwarding maps all inbound edges to the same outbound edge, and “per-incoming-edge” forwarding maps each inbound edge to a uniformly selected outbound edge (with replacement).

Perhaps counterintuitively, these spreading mechanisms alter the anonymity guarantees even when the graph is unknown. Our next result—the second main theoretical contribution of this paper—suggests that one-to-one forwarding has near-optimal precision when the adversary does not know the graph, even in the face of intersection attacks (recall the lower bound of \( p^2 \) \([13]\)). The other two mechanisms do not.

**Theorem 2:** Suppose the graph is unknown to the adversary, each node generates an arbitrary number of transactions, and the adversary can link transactions from the same user. The expected precision of the precision-optimal estimator for the one-to-one (\( D_{OPT-IO} \)), all-to-one (\( D_{OPT-AO} \)), and per-incoming-edge (\( D_{OPT-PIE} \)) message forwarding schemes are:

\[
D_{OPT-IO} = O(p^2 \log \left( \frac{1}{p} \right)) \tag{5}
\]

\[
D_{OPT-AO} = O(p) \tag{6}
\]

\[
D_{OPT-PIE} = \Omega(p). \tag{7}
\]

(Proof in Section \([IV-B1]\))

Figure 6 illustrates simulated first spy precision values for each of these techniques, as well as diffusion (the status quo). ‘Per-transaction’ forwarding denotes the baseline i.i.d. random forwarding. This figure is plotted for the special case of one transaction per node; if nodes were to generate arbitrarily many transactions, the pseudorandom lines would stay the same (all-to-one, one-to-one, and per-incoming-edge), whereas Figure 5 suggests that the per-transaction curve could increase arbitrarily close to 1, depending on traffic patterns. The same is true for diffusion, as illustrated in prior work \([47]\). In addition, when the graph is unknown and there is only one transaction per node, Figure 6 suggests that one-to-one forwarding achieves precision values that are close to the lower bound of per-transaction forwarding. An important question is what happens when the graph is known. The following corollary bounds the jump in precision under such a scenario.

**Proposition 1:** If the adversary knows the graph and internal routing decisions, then the precision-optimal estimator for one-to-one forwarding has an expected precision of \( O(p) \).

(Proof in Appendix \([D]\))

An adversary that does not know internal forwarding decisions for all nodes has lower precision, because it must disambiguate between exponentially many paths for each transaction. Despite requiring completely new analysis, the 4-regular graph results for Theorem 2 and Proposition 1 are order-equivalent to the line graph results in \([13]\). Nonetheless, we maintain that 4-regular graphs are preferable to line graphs because (1) the precision bound constants are smaller (as seen in \([IV-A]\), an (2) 4-regular graphs are more difficult to learn, since more information is needed to specify a 4-regular graph. Quantifying this fact is an interesting question for future work.

**Lesson.** Use pseudorandom, one-to-one forwarding.

1) **Proof of Theorem 2.** We start with two lemmas that reduce the problem to a first-spy precision calculation.

**Lemma 3 (Intersection):** Under each of the spreading mechanisms (all-to-one, one-to-one, and per-incoming-edge), the adversary’s maximum expected precision \( D_{OPT} \) is not a function of the number of transactions per node.

This proof follows directly from the pseudorandomness of the forwarding mechanisms, and is omitted for brevity.
Lemma 4 (First-Spy Optimality): For all spreading mechanisms (all-to-one, one-to-one, and per-incoming-edge), there exists a constant $C$ such that $D_{FS}^\Omega \leq C \cdot D_{FS} + O(p^2)$, where $D_{FS}$ denotes the expected precision of the first-spy estimator.

Lemma 4 is proved analogously to Theorem 1. The full proof is omitted for brevity. Together, lemmas 3 and 4 imply that to characterize the precision-optimal estimator, we can focus on the first-spy estimator. For brevity, we prove only all-to-one and one-to-one results in the following lemmas.

Lemma 5 (One-To-One First-Spy): The expected precision of the first-spy estimator for one-to-one forwarding satisfies $D_{FS-OLT} = O(p^2 \log \frac{1}{p})$.

Proof: Let $v \in V_H$, and denote the vertex to which $v$ forwards its message $X_v$ as $s \in V$. In the case that $s \in V_H$, the precision of the first-spy estimator for $v$ is 0, because $X_v$ is never matched to $v$. When $s \in V_A$, the precision of the first-spy estimator for $v$ is $\frac{1}{|V_s|}$, where $W_v$ denotes the set of nodes from which all fresh messages that $v$ transmits to $s$ originate. Note that $v \in W_v$. Now $E[D_{FS}(v)] = P(s \in V_A)E[\frac{1}{|V_s|}|s \in V_A]$, where $P(s \in V_A) = p$ and $E[\frac{1}{|V_s|}|s \in V_A] = \sum_{w=1}^{\infty} P(|W_s| = w|s \in V_A) \frac{1}{w}$.

Lemma 6: Under one-to-one forwarding, $P(|W_s| = w|s \in V_A) = \frac{2p}{1-p} \frac{1-p}{1+p}^w$.

Proof: Using Lemma 6 to expand the summation gives $E[\frac{1}{|V_s|}|s \in V_A] = \sum_{w=1}^{\infty} \frac{2p}{1-p} \frac{1-p}{1+p}^w = -\frac{2p}{1+p} \log \frac{2p}{1+p}$. Thus, $E[\frac{1}{|V_s|}|s \in V_A] = 2p \frac{1-p}{1+p} \log \frac{1+p}{2p}$.

Lemma 7 (All-To-One First-Spy): The expected precision of the first-spy estimator for all-to-one forwarding satisfies $D_{FS-OLT} = O(p)$.

Proof: We demonstrate upper and a lower bounds on $D_{FS-OLT}$. As before, these bounds are obtained by computing the expected precision of a given node $v \in V_H$, and we denote the node to which $v$ forwards its message $X_v$ as $s \in V$. Let $W_v$ be the set of nodes from which all fresh messages transmitted by $v$ to $s$ originate; our goal is to compute $E[\frac{1}{|V_s|}|s \in V_A]$.

Lower bound. We use the following lemma:
let $\nu$ denote the probability that a node is a spy or forwards its transactions on its grey edge, $\nu = \frac{1}{2}(1 + p)$. The mean of this offspring distribution $\mu = 2(1 - \nu) = 1 - p < 1$ for $p = 0$, so this GW process is subcritical and goes extinct with probability 1 $[9]$. Hence we can focus our analysis on trees of finite size. We use Lemma 8 to expand $E \left[ \frac{1}{|W|} \right.$ $\left. s \in V_A \right] =$

$$= \sum_{w=1}^{\infty} \frac{1}{w(w+1)!} \frac{(2w)!}{w!} \left( \frac{1 - p}{2} \right)^{w-1} \left( \frac{1 + p}{2} \right)^{w+1}$$

$$= \frac{1}{4} (1 + p)^2 \sum_{w=1}^{\infty} \frac{2c^2}{w} 2^w \left( \frac{1}{4} (1 - p^2) \right)^{w-1} \left( \frac{1}{4} (1 - p^2) \right)^{w+1}$$

(8)

where $\leq \left( \frac{1}{2}\right)^{w+1} \leq \frac{1}{w(w+1)}$. This makes the above expression $\leq 1$. Hence, $E \left[ \frac{1}{|W|} \right.$ $\left. s \in V_A \right] \leq \frac{c^2}{2\pi} (1 + p)^2$ and $D_{PS-ADO} \leq \frac{c^2}{2\pi} p + O(p^2)$. \hfill $\blacksquare$

C. Graph-Construction Attacks

We have considered graph-learning attacks and intersec­tion attacks. Another important aspect of TWISTEDPAIR is the anonymity graph construction, which should be fully distributed. DANDELION proposed an interactive, distributed algorithm (explained below) that constructs a randomized approximate line graph. In this section, we study how Byzantine nodes can change the graph to boost their accuracy. First, we show how to generate 4-regular graphs in the presence of Byzantine nodes. Next, we show how to choose an integral parameter for robustness against Byzantine nodes manipulating the graph. We start with the graph-construction protocol from DANDELION.

1) Graph construction in DANDELION $[13]$: For any integral parameter $\ell > 0$, DANDELION uses the protocol ApxLine($\ell$) to build a $\ell$-approximate line graph as follows: (1) Each node contacts $\ell$ random candidate neighbors, and asks for their current in-degrees. (2) The node makes an outbound connection to the candidate with the smallest in-degree. Ties are broken at random. This protocol is simple, distributed, and allows the graph to be periodically rebuilt. Though the resulting graph need not be an exact line, nodes have an expected degree of two, and experiments show low precision. Increasing $\ell$ also enhances the likeness of the graph to a line and reduces the expected precision in simulation.

2) Construction of 4-regular graphs.: A natural extension of DANDELION’s graph-construction protocol to 4-regular graphs involves repeating ApxLine($\ell$) twice for parameter $\ell > 0$. That is, each peer makes two outgoing edges, where the target of each edge is chosen according to ApxLine($\ell$). As in the approximate line algorithm, the resulting graph is not exactly regular. However, the expected node degree is 4, and because each node generates two outgoing edges, the resulting graph has no leaves. This improves anonymity because leaf nodes are known to degrade average precision $[13]$.

3) The impact of Byzantine nodes: Byzantine nodes can misbehave as recipients and/or creators of edges. As recipients, nodes can lie about their in-degrees during the degree-checking phase. As creators of edges, misbehaving nodes can generate many edges, even connecting to each honest node.

<table>
<thead>
<tr>
<th>Algorithm 2: APPROXIMATE 4-REGULAR GRAPH</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Set $V = {v_1, v_2, \ldots, v_n}$ of nodes;</td>
</tr>
<tr>
<td><strong>Output:</strong> A connected, directed anonymity graph $H(V,E)$ with average degree $^4$</td>
</tr>
<tr>
<td><strong>for</strong> $v$ $\leftarrow$ $V$ <strong>do</strong></td>
</tr>
<tr>
<td>$/<em>$ pick two random targets $</em>/$</td>
</tr>
<tr>
<td>$u_1 \sim$ Unif($V \backslash {v}$)</td>
</tr>
<tr>
<td>$u_2 \sim$ Unif($V \backslash {v, u_1}$)</td>
</tr>
<tr>
<td>$/<em>$ make connections $</em>/$</td>
</tr>
<tr>
<td>$E = E \cup (v \rightarrow u_1) \cup (v \rightarrow u_2)$</td>
</tr>
<tr>
<td><strong>end</strong></td>
</tr>
<tr>
<td><strong>return</strong> $H(V,E)$</td>
</tr>
</tbody>
</table>

Lying about in-degrees. In step (1) of the graph-construction protocol, when a user queries an adversarial neighbor for its current in-degree, the adversary might deliberately report a lower degree than its actual degree. This can cause the querying user to falsely underestimate the neighbor’s in-degree and make a connection. In the extreme case, the adversarial node can consistently report an in-degree of zero, thus attracting many incoming edges from honest nodes. This degrades anonymity by increasing the likelihood of honest nodes passing their transactions directly to the adversary in the first hop. We find experimentally that such attacks significantly increase precision as $\ell$ grows; plots are omitted due to space constraints.

To avoid nodes lying about their in-degrees, we abandon the interactive aspect of DANDELION graph construction. In ApxLine(1), users select a random peer and make an edge regardless of the recipient’s in-degree. We therefore run ApxLine(1) twice, as shown in Algorithm 2. Note that Algorithm 2 closely mirrors the graph construction protocol used in Bitcoin’s P2P network today, so the protocol itself is not novel. In the line graphs of DANDELION, a higher $\ell$ value was needed since ApxLine($\ell$) for $\ell = 1$ was shown to have significantly worse anonymity performance than $\ell \geq 2$. However such a loss is avoided in TWISTEDPAIR since the application of ApxLine(1) twice eliminates leaves.

What is not previously known is how the approximate-regular construction in Algorithm 2 affects anonymity compared to an exact 4-regular topology. First, note that the expected recall does not change because dandelion spreading (Algorithm 2) has an optimally low maximum recall of $p + O(\frac{1}{\lambda})$, regardless of the underlying graph (Thm. 4, $[13]$). Hence, we wish to understand the effect of approximate regularity on maximum expected precision. We simulated dandelion spreading on approximate 4-regular graphs and exact 4-regular graphs, using the same approximate precision-optimal estimator from Figure 3. For comparison, we have also included the first-spy estimator. Figure 8 shows that the difference in precision between 4-regular graphs and approximate 4-regular graphs (computed with Algorithm 2) is less than 0.02 across a wide range of spy fractions $p$. Compared to line graphs $[13]$ Figure 8], 4-regular graphs appear significantly more robust to irregularities in the graph construction.

Creating many edges. DANDELION is naturally robust to nodes that create a disproportionate number of edges, because
Fig. 8: Average precision for approximate 4-regular graphs compared to exact 4-regular graphs.

Fig. 9: Honest-but-curious spies obey graph construction protocol.

Fig. 10: Malicious spies make outbound edges to every honest node.

Proposition 2: Consider dandelion spreading (Algorithm 4) with \( q = 0 \) over a connected anonymity graph \( H \) constructed according to graph-construction policy \( P \) (Algorithm 1). Let \( D_{OPT}(P) \) and \( R_{OPT}(P) \) denote the maximum expected precision and recall over graphs constructed according to \( P \). Now consider an alternative policy \( Q \) that is identical to \( P \) except adversarial nodes are allowed to choose their outbound edges arbitrarily. Let \( D_{OPT}(Q) \) and \( R_{OPT}(Q) \) denote the maximum expected precision and recall over all graphs constructed according to \( Q \). Then

\[
R_{OPT}(Q) = R_{OPT}(P) \\
D_{OPT}(Q) = D_{OPT}(P). 
\]

(Proof in Appendix 2). This result bounds the deanonymization abilities of Byzantine nodes in general and supernodes in particular [10], [28], neither of which was covered by the analysis of [13]. It shows that for the special case where the transition probability from stem to fluff phase is zero (i.e., infinite stem phase), supernodes gain no deanonymization power by connecting to most or all of the honest nodes. In practice, we need \( q > 0 \) to reduce broadcast latency, but analyzing this requires an upper bound on the probability of detecting the source of a diffusion process under sampled timestamp observations—a known open problem [38], [51]. We therefore simulate precision for nonzero \( q \) as a function of spy fraction \( p \), when spies obey protocol (Fig. 9) and when they generate outbound edges to every honest node (Fig. 10).

We generate a P2P graph via Algorithm 1 with out-degree \( q = 0 \). Now consider an alternative policy \( Q \) that is identical to \( P \) except adversarial nodes are allowed to choose their outbound edges arbitrarily. Let \( D_{OPT}(Q) \) and \( R_{OPT}(Q) \) denote the maximum expected precision and recall over all graphs constructed according to \( Q \). Then

\[
R_{OPT}(Q) = R_{OPT}(P) \\
D_{OPT}(Q) = D_{OPT}(P). 
\]

(Proof in Appendix 2). This result bounds the deanonymization abilities of Byzantine nodes in general and supernodes in particular [10], [28], neither of which was covered by the analysis of [13]. It shows that for the special case where the transition probability from stem to fluff phase is zero (i.e., infinite stem phase), supernodes gain no deanonymization power by connecting to most or all of the honest nodes. In practice, we need \( q > 0 \) to reduce broadcast latency, but analyzing this requires an upper bound on the probability of detecting the source of a diffusion process under sampled timestamp observations—a known open problem [38], [51].

We therefore simulate precision for nonzero \( q \) as a function of spy fraction \( p \), when spies obey protocol (Fig. 9) and when they generate outbound edges to every honest node (Fig. 10). We generate a P2P graph via Algorithm 1 with out-degree \( q = 0 \), and an anonymity graph \( H \) via Algorithm 2 except spies generate outbound edges to every honest node.

Figures 9 and 10 highlights two points: First, even when spies follow protocol, increasing \( q \) increases precision. \( q \) has the largest effect when \( p \) is small; when \( p = 0.1 \), using \( q = 0.5 \) increases the expected precision by about 0.1 compared to \( q = 0 \). For \( q \leq 0.2 \), we can expect increases in precision on the order of 0.05. Second, spies can increase their precision by adding outbound edges; when \( p = 0.1 \) and \( q = 0.5 \), we observe a precision increase of about 0.2. Thus by choosing parameter \( q \leq 0.2 \), we can limit the increase in average precision to 0.1, even when spies connect to every honest node.

Lesson. To defend against graph-manipulation attacks, use noninteractive protocols and small \( q \).

D. Black-Hole Attacks

Since dandelion spreading forwards messages to exactly one neighbor in each hop, propagation can terminate entirely if an adversarial relay chooses not to forward a message; we refer to this as a black-hole attack. To prevent black-hole attacks, TWISTEDPAIR sets a random expiration timer at each stem relay immediately upon receiving transaction messages. If the relay does not receive an INV (i.e. an advertisement) for the transaction before his timer expires, then the relay diffuses the transaction. This policy provides a two-fold advantage over DANDELION: (1) messages are guaranteed to eventually propagate through the network, and (2) the random timers can help anonymize peers initiating the spreading phase in the event of a black-hole attack.

To implement this, TWISTEDPAIR nodes are initialized with a timeout parameter \( T_{base} \). In the stem phase, when a relay \( v \) receives a transaction, it sets an expiration time \( T_{out}(v) \):

\[
T_{out}(v) \sim current\_time + \exp(1/T_{base}),
\]

i.i.d. across relays. If the transaction is not received again by relay \( v \) before \( T_{out}(v) \), \( v \) broadcasts the message using diffusion. Pseudocode is shown in Algorithm 5 (Appendix A).

Algorithm 5 solves the problem of message stalling, as relays independently broadcast if they have not received the message within a certain time. However, the protocol also ensures that the first relay node to broadcast is approximately uniformly selected among all relays that have received the message. This is due to the memorylessness of the exponential clocks: conditioned on a given node blocking the message, each of the remaining clocks can be reset assuming propagation latency in the stem is negligible. Ideally, the exponential clocks should be slow enough that they only trigger (with high probability) during a black-hole attack. On the other hand, they must also be fast enough to keep propagation latency low. This trade-off is analyzed in the following proposition.

Proposition 3: For a timeout parameter \( T_{base} \geq \frac{k(1-\delta_{hop})}{2 \log(1-\epsilon)} \), where \( k, \epsilon \) are parameters and \( \delta_{hop} \) is the time between each hop (e.g., network and/or internal node

Input: Main (directed) P2P graph $G(V,E)$, desired degree $d$ of output anonymity graph

Output: Directed anonymity graph $H(V,E)$

for $v \in V$ do
  /* Find TwistedPair neighbors */
  $D_v = \{ w \in N_{\text{out}}(G,v) \mid w \text{ supports TwistedPair} \}$
  if $0 \leq |D_v| < \frac{d}{2}$ then
    $E \leftarrow E \cup D_v$
  end
/* Non-TwistedPair neighbors */
if $|D_v| = 0$ then
  $\mathcal{R} \leftarrow \frac{d}{4}$ nodes drawn uniformly from $N_{\text{out}}(G,v)$, without replacement
  $E \leftarrow E \cup \mathcal{R}$
end

The standard deviation of $\Delta_2$ is identical to the mean. Thus by choosing $T_{\text{base}}$ as in Proposition 3 in Algorithm 5 we incur an additional delay at most a constant factor $1/(2e)$ from our delay $\Delta_1$ otherwise.

Lesson. Use random timers selected according to Prop. 7.

E. Partial-Deployment Attacks

The original Dandelion analysis does not consider the fact that instantaneous, full-network deployment of TwistedPair is practically infeasible. In this section, we show that if implemented naively, Byzantine nodes can exploit partial Dandelion deployment to launch serious anonymity attacks. We also demonstrate a (counterintuitive) implementation mechanism that neutralizes this threat. Our discussions with Bitcoin Core developers suggest that they are more likely to accept solutions in which the anonymity graph is an overlay of the existing P2P network. Given this, we consider two natural approaches for constructing the anonymity graph.

Version-checking (Algorithm 3) generates an anonymity graph from edges between TwistedPair-compatible nodes. Each node $v$ first identifies its outbound peers on the main P2P network that support TwistedPair; we call this set $D_v$. $D_v$ can be learned from existing signaling in Bitcoin’s version handshake. Next, $v$ runs Algorithm 2, drawing candidate neighbors only from $D_v$. If $|D_v| = 1$, $v$ uses the single node in $D_v$ as its outbound anonymity graph edge. If $|D_v| = 0$, $v$ picks 2 outbound neighbors uniformly at random, and uses them as anonymity graph edges. Upon forwarding a transaction to a node that does not support TwistedPair, the receiving node will, by default, relay the message using diffusion, thereby ending the stem phase. While version checking is a natural strategy, adversarial nodes can lie about their version number and/or run nodes that support TwistedPair.

Under the second approach, no-version-checking, each node $v$ instead selects 2 outgoing edges uniformly from the set of all outgoing edges, without considering TwistedPair-compatibility. This noninteractive protocol shortens the expected length of the stem, thereby potentially weakening anonymity guarantees.

If all nodes supported TwistedPair, these two approaches would be identical. To model gradual deployment, we assume that all spy nodes run TwistedPair, and a fraction $\beta$ of the remaining honest nodes are using TwistedPair. Honest users are distributed uniformly over the network. Let $V_D$ denote the nodes that support TwistedPair: $|V_D| = \frac{pn}{\beta} + (1 - \beta)n$. We wish to characterize the maximum expected recall of TwistedPair as a function of $p$ and $\beta$, for version-checking and no-version-checking. The following theorem bounds this quantity under a recall-optimal estimator.

Theorem 3: Consider $n$ nodes in an approximately-$2\eta$-regular graph $G$ generated according to Algorithm 1. A fraction $p$ of nodes are spies running TwistedPair. Among the remaining honest nodes, a fraction $\beta$ support TwistedPair. Under version-checking, the expected recall across honest, TwistedPair-compatible nodes, under a recall-optimal mapping strategy satisfies

$$\frac{p}{f} (1 - (1 - f)^n) \leq R_{\text{OPT}} \lesssim \frac{p}{f} (1 - (1 - f)^n) + \left(1 - (1 - f)^n + \frac{C(1 - \beta)}{n} \right)$$

(12)

where $f$ is the fraction of all nodes that support TwistedPair and $C = 1 - \frac{p}{f} (1 - (1 - f)^n) \frac{1 - (1 - \phi)}{n\phi}$, where $\phi = 1 - (1 - \frac{1}{n\phi})^n$, and $n = (1 - p)n$ is the number of honest nodes in the system. Under no-version-checking,

$$p \leq R_{\text{OPT}} \lesssim p + (1 - \beta)(1 - q)\left(1 - \frac{p}{f} \phi \frac{n}{n\phi} \right).$$

(13)

(Proof in Appendix G) Here $\lesssim$ denotes approximate inequality; i.e., $A(n) \lesssim B(n)$ implies that there exist constants $n_0 > 0$ and $C' > 0$ such that for all $n > n_0$, $A(n) \leq C'B(n)$. In this case, such a condition holds for any $C' > 1$.

Figure 11 plots these results for $p = 0.2$ fraction of spies and $q = 0.2$ probability of ending the TwistedPair stem, on an approximately-16-regular P2P graph. Our theoretical bounds delimit the shaded regions; for comparison, we include a lower bound on the recall of diffusion, computed by simulating diffusion with a first-spy estimator. The green solid line is a lower bound on the maximum expected recall of any spreading protocol (Thm. 2, [13]). When $\beta$ is small, version-checking recall close to 1; in the same regime, no-version-checking exhibits lower recall than both diffusion and
version-checking. Intuitively, when adoption is low (low $\beta$), any TWISTEDPAIR peers are likely to be spies. Therefore, version-checking actually increases the likelihood of getting deanonymized. In the same low-$\beta$ regime, no-version-checking is more likely to choose a TWISTEDPAIR-incompatible node $w$ as the next stem node. While this prematurely ends the stem, it still introduces more uncertainty than vanilla diffusion.

**Lesson.** Use no-version-checking to construct the graph.

V. Evaluation

A. Implementation in Bitcoin

We have developed a prototype implementation of TWISTEDPAIR. Our prototype is a modification of Bitcoin Core (referred to as Core from now on), the most commonly-used client implementation in the Bitcoin network. In total, our implementation required a patch modifying approximately 500 lines of code. A vital part of our implementation is allowing nodes to recognize other TWISTEDPAIR nodes in a straightforward way. In Core, the way to signal supported features is by modifying the nServices field in the handshake. For example, segwit support is signalled by setting bit 4 in nServices. In our case, TWISTEDPAIR support is signalled by setting the 25th bit.

In order to minimize our footprint on the Core codebase, we insert TWISTEDPAIR functionality into the preexisting main threads/signals that handle the processing and transmission of messages. However, creating the 4-regular anonymity graph and processing TWISTEDPAIR transactions requires careful consideration of the many concurrency and DoS protection mechanisms already at play in Core. For instance, Core’s data structures for transactions and inventory messages are designed to facilitate responses to GetData requests, while broadcasting transactions to all nodes with exponential delays. These data structures are not sufficient for TWISTEDPAIR because they facilitate broadcasting knowledge of transaction and block hashes. In particular, TWISTEDPAIR nodes need to hide knowledge of transactions that are still in the stem phase, but at the same time ensure that they are relayed properly and not stalled by adversaries. Our approach is to store stem mode transactions in an additional data structure, the “embargo map,” such that embargoed transactions are omitted from GetData responses. The embargo map serves two purposes: 1. it tracks transactions that are currently in the stem phase and 2. it ensures that malicious adversaries can not stop propagation of transactions. When a TWISTEDPAIR transaction arrives, the node marks the transaction as “under embargo” and assigns a time, $t = 10s + \text{Poisson}(30)$, that the node waits to see the transaction again before forcing the fluff phase itself. If the next hop in the stem phase is a malicious node that stops stem propagation, the embargo time ensures the transaction will still be propagated.

B. Experimental Setup

We used our prototype implementation to conduct integration experiments, by launching our own nodes running the TWISTEDPAIR software, and connecting to the actual Bitcoin network. The primary goal of our experiments is to characterize how TWISTEDPAIR affects transaction propagation latency. The experiments also serve to validate our implementation and its compatibility with the existing network.

For our experiments we launched a total of 30 Dandelion instances of m3.medium Amazon EC2 nodes (t2.medium used in Seoul and Mumbai where m3.medium is not available). The nodes are spread geographically across 10 different AWS regions (California, Sydney, Tokyo, Frankfurt, etc.) – 3 nodes per region. To control the topology between the Dandelion nodes, we use Core’s -connect or -addnode command line flags. Our measurements use the Coinscope tool to connect to each node in the Bitcoin network and record a timestamped log of transaction propagation messages.

C. Evaluation Results

1) Propagation latency at fixed stem lengths: We conducted a preliminary experiment to inform our choice of the coin flip stem length parameter, $q$. In this experiment, we arranged our TWISTEDPAIR nodes in a chain topology (each with one outgoing connection to the next, with the last
node connected to the Bitcoin network) so that we could deterministically control the stem length. Based on 20 trials for stems up to length 12, we estimated that each additional hop adds an expected 300 milliseconds to the propagation latency. There is also a constant 2.5 second delay added to each transactions due to the exponential process of propagation when it first enters the fluff phase. Taking a propagation delay of 4.5 seconds as our goal, we chose \( q = 0.1 \) or \( q = 0.2 \) as our parameter (recall the expected stem length is \( 1/q \)).

The observed delays originate from two primary sources. First, each hop incurs network latency due to transit time between nodes. A recent measurement study of the Bitcoin network estimated a median latency of 110ms between nodes [22], and Bitcoin transaction propagation requires three messages (INV, GETDATA, TX). The latency between our EC2 nodes is somewhat faster, with a median of only 86ms across all pairs. Although our EC2 nodes are geographically distributed, they are closely connected to internet backbone endpoints. Second, Bitcoin Core buffers each INV messages for an average of 2.5 seconds; however, our implementation pushes TWISTEDPAIR transactions to be relayed immediately, so internal node delays should be negligible. We therefore heuristically estimate 300 milliseconds of delay per stem node, which is consistent with our preliminary experiments.

2) Topology: In order to create a connected graph of TWISTEDPAIR nodes, we use all of the eight outgoing connections from each of our nodes to connect to other TWISTEDPAIR nodes. This ensures that we can measure many different stem lengths and how they affect the propagation to the rest of the network. We must also account for an artifact of our experiment setup, namely that short cycles in the stems are more likely among our 30 well connected TWISTEDPAIR nodes. We therefore parse debug logs from our nodes for each trial in order to determine the effective stem length, which we then use as the basis of our evaluation.

In our experiment, we re-randomize the topology frequently so as to not bias our results to a specific setting. For each randomization of the connections, we generate 5 transactions, 10 seconds apart. We repeat this “burst” 4 times a day over three separate days. The transactions are injected to the rest of the network. A recent measurement study of the Bitcoin network estimated a median latency of 110ms between nodes [22], and Bitcoin Core buffers each INV messages for an average of 2.5 seconds; however, our implementation pushes TWISTEDPAIR transactions to be relayed immediately, so internal node delays should be negligible. We therefore heuristically estimate 300 milliseconds of delay per stem node, which is consistent with our preliminary experiments.

In our experiment, we re-randomize the topology frequently so as to not bias our results to a specific setting. For each randomization of the connections, we generate 5 transactions, 10 seconds apart. We repeat this “burst” 4 times a day over three separate days. The transactions are injected into the network via a randomly chosen one of our nodes. We show the results of the experiment in Figures [12] and [13].

Figure [12] plots the time it takes TWISTEDPAIR transactions to reach 10% of the network. As mentioned in Section V-C1 for every additional hop in the stem phase there is a minimum delay added by three messages and a expected delay of 2.5 seconds. The solid green line represents the minimum expected delay and the dotted blue line represent the best linear fit over the transaction data. As we expect, the propagation delay to reach 10% coverage increases with the path length due to the minimum added delay. We also computed the two-sided Pearson Correlation Coefficient over the two variables; path length and time to reach 10%. The coefficient \( r = 0.292 \) implies a small positive correlation between the two variables.

Figure [13] plots the time it takes a transaction to go from 10% to 50% of the network with respect to the path length.

Unlike the first scatter plot, visual inspection doesn’t reveal any relationship between the two variables in this case. This is also what we expect because TWISTEDPAIR should not have any impact on transaction propagation after it has entered fluff phase. We also perform a Mann-Whitney U Test to test the null hypothesis: time from 10-50% coverage does not depend on path length. Using the path length as the independent variable, we split it into two categories: high and low. The blue dotted line in Figure [13] is the boundary of the two categories where there are 26 samples in “low” and 29 in “high”. The Mann Whitney U test gives a U statistic of 443 and a p-value of 0.269. This implies that there is weak evidence against the null hypothesis, therefore we fail to reject it.

As expected, the minimum delay brought on by TWISTEDPAIR has a positive correlation with the time it takes to reach 10% of the Bitcoin network. Similarly, once a transaction has left the stem phase, it begins normal propagation through the network and is therefore no longer be affected by TWISTEDPAIR. This prediction is confirmed by our test of independence of hop length (high vs. low) and time from 10-50% coverage.

VI. CONCLUSION

A gap exists between the theory and practice of protecting user anonymity in cryptocurrency P2P networks. In particular, there are no safeguards against population-level deanonymization, which is the focus of this paper. We aim to narrow that gap by identifying strong or unrealistic assumptions in a state-of-the-art proposal [13], demonstrating the anonymity effects of violating those assumptions, and proposing lightweight fixes in the form of TWISTEDPAIR. This methodology complements the usual development pattern in cryptocurrencies, which has mainly evolved by applying ad hoc patches against specific attacks. We instead take a first-principles, theoretically-justified approach to design.

Finally, TWISTEDPAIR does not explicitly protect against ISP- or AS-level adversaries, which can deanonymize users through routing attacks [8]. Understanding how to analyze and protect against such attacks is of fundamental interest. However, note that TWISTEDPAIR is already compatible with a number of the countermeasures proposed in [8]. For instance, [8] proposes to enhance network diversity through multi-homing of nodes and routing-aware network connectivity. Such countermeasures directly support TWISTEDPAIR by ensuring that nodes are less likely to establish outbound anonymity edges exclusively to spies. We expect that understanding how to enforce such network diversity in the presence of Byzantine nodes will be an important question moving forward, both for anonymity and general network robustness.

REFERENCES


APPENDIX

A. Algorithms

DANDELION pseudocode is presented in Algorithm 4

Pseudocode for handling black-hole attacks is included in Algorithm 5

B. Proofs of Lemmas Used in Theorem 7

Lemma 1: Let $J_v$ denote the number of transactions (from honest servers) that reach $v$ before reaching an adversary. Then, for each event $E \in \{ E_{V_{(h, h) \rightarrow h, h)}, E_{V_{(h, h) \rightarrow h, h)}, E_{V_{(h, h) \rightarrow h, h)} \}$ $E[\max_{x \in X} P(\{X_v = x\mid S, \Gamma(V_A), E, J_v\}) | E, J_v] \leq \frac{1}{x+1}$

Proof: W.l.o.g., let $U_1, U_2, \ldots, U_{n_v}$ be the servers whose transactions are received by $v$ and let $W_1 =$
Algorithm 4: Dandelion Spreading [13]. $\mathcal{N}_{\text{out}}(G,v)$ denotes the out-neighbors of node $v$ on directed graph $G$.

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

Algorithm 5: TwistedPair Spreading at node $v$. The protocol guarantees eventual network-wide propagation of transactions.

**Input**: Message and timeout parameter $(X,T_{\text{base}})$ received by $v$ in the anonymity phase, out-neighbors $\mathcal{N}_{\text{out}}(G,v)$ on anonymity graph $G$, spreading graph $H$, parameter $q \in (0,1)$
$T_{\text{out}}(v) \sim \exp(1/T_{\text{base}})$ /* set timer
forward $(X,T_{\text{base}})$ according to dandelion
/* wait until message re-received */
while current_time $\leq T_{\text{out}}$ do
    if $X$ received then
        timer $\leftarrow$ inactive
    end
    continue
end
/* start diffusion */
if timer is active then
    DIFFUSION($X,v,H$)
end

\[
\{v,U_1,U_2,\ldots,U_{J_v}\}. \text{ Consider any matching } x \text{ where } x_u \text{ is the message assigned to server } u. \text{ Then }
\]

\[
\mathbb{P}(S|G,W_v,X = x) = \mathbb{P}(S|G,W_v,X = x')
\]  
(14)
where $x'$ is a new assignment of messages such that $x'_u = x_u$ for all $u \in V_H$, $u \notin \mathcal{N}_v$ and $x'_w = x_w$. This implies, for any fixed $x \in \mathcal{X}$,

\[
\mathbb{P}(X_v = x, S, X_{V_H \setminus \{v\}}|\Gamma(V_A), E, J_v, G, W_v) = \mathbb{P}(X_v = x, S, X_{V_H \setminus \{v\}}|\Gamma(V_A), E, J_v, G, W_v)
\]

\[
= \mathbb{P}(S|\Gamma(V_A), E, J_v, G, W_v, X_v = x, X_{V_H \setminus \{v\}}) \\
\leq \frac{1}{J_v + 1}
\]  
(15)
\[
\Rightarrow \mathbb{P}(X_v = x|S, \Gamma(V_A), E, J_v) \leq \frac{1}{J_v + 1}
\]

\[
\Rightarrow \max_{x \in A} \mathbb{P}(X_v = x|S, \Gamma(V_A), E, J_v) \leq \frac{1}{J_v + 1}
\]

\[
\mathbb{E}[\max_{x \in A} \mathbb{P}(X_v = x|S, \Gamma(V_A), E, J_v)]|E,J_v| \leq \frac{1}{J_v + 1}
\]

The claim follows. \]

**Lemma 2**: For any server $v \in V_H$, let $F_v$ denote the number of transactions that (i) reach $v$ before reaching any adversary and (ii) are forwarded by $v$ along its left outgoing edge. Then $\mathbb{E} \left[ \frac{1}{1+ \frac{w}{p}} \right] \leq 2D_{PS}(v)$.

**Proof**: For server $v$, consider event $E_v$ in which the node incident on $v$’s left outgoing edge is an adversary. Also, let $\mathcal{L}_v$ denote the event that $X_v$ is forwarded along $v$’s left outgoing edge. Then clearly,

\[
D_{PS}(v) \geq \mathbb{P}(E_v, \mathcal{L}_v)E[D_{PS}(v)|E_v, \mathcal{L}_v] = \frac{p}{2} \mathbb{E}[D_{PS}(v)|E_v, \mathcal{L}_v].
\]  
(17)

Now, from our assumption $F_v$ is independent of the events $E_v$ and $\mathcal{L}_v$. In this case, the expected precision becomes $\mathbb{E}[D_{PS}(v)|E_v, \mathcal{L}_v] = \mathbb{E} \left[ \frac{1}{1+ \frac{w}{p}} \right]$ , which combined with Equation (17) gives the lemma. \]

C. Proofs of Lemmas Used in Theorem 2

**Lemma 6**: Under one-to-one forwarding, $\mathbb{P}(|W_v| = w | s \in V_A) = \frac{2p}{1+p} \frac{1-p^w}{1-p}$.

**Proof**: Under one-to-one forwarding, all messages other than $X_v$ transmitted by $v$ to $s$ are received by $v$ from the same predecessor $v'$. Likewise, all messages transmitted by $v'$ to $v$ are received by $v'$ from the same predecessor $v''$ (other than $X_v'$ in the case that $X_v'$ is transmitted to $v$). Continuing this reasoning results in a line graph of predecessor nodes. Note that the first adversary predecessor node in this line graph prevents any subsequent predecessors from contributing to $W_v$. We condition on $|T|$, the number of vertices in the line graph: $\mathbb{P}(|W_v| = w | s \in V_A) = \infty_{t=0}^{\infty} \mathbb{P}(|T| = t | s \in V_A) = \frac{p}{2} \mathbb{E} \left[ \frac{1}{1+ \frac{w}{p}} \right]$.

\[
\frac{p}{2} \frac{1-p^{w-1}}{1-p} \left( \frac{1}{2} \right)^{w-1} \frac{1}{2} \frac{1}{2} \frac{1}{2} = 2p \frac{1-p^{w-1}}{(1+p)^w}
\]

**Lemma 8**: Under all-to-one forwarding, $\mathbb{P}(|W_v| = w | s \in V_A) = \frac{2(pw)^w}{(w+1)!} \frac{1-p}{1+p} \frac{1-p}{1+p} \frac{1-p}{1+p} = \frac{2(pw)^w}{(w+1)!} \frac{1-p}{1+p} \frac{1-p}{1+p} \frac{1-p}{1+p}$.

**Proof**: In a 4-regular digraph, every $v \in V$ has two predecessors. Thus, the upstream graph of $v$ may be modeled as a binary tree $B_v$ rooted at $v$. For any member vertex $m$ of $B_v$, if $m \in V_A$ then neither $m$ nor any members of the sub-tree rooted at $m$ transmit fresh messages to $s$ via $v$. Additionally, every vertex transmits all received and generated messages
across one outbound edge (either left or right with equal probability) for the entire epoch. No messages are transmitted across the other outbound edge for the entire epoch. For any member vertex \( m \) of \( B_v \), if \( m \) transmits messages across the outbound edge that is not part of \( B_v \) then neither \( m \) nor any members of the sub-tree rooted at \( m \) may contribute messages transmitted by \( v \) to \( s \). These cases occur with probability \( \frac{1}{2} \).

Accounting for these cases yields a new tree \( T_v \) rooted at \( v \). This tree \( T_v \) consists of the remaining members of \( B_v \) after pruning sub-trees rooted at adversary nodes and sub-trees rooted at nodes that transmit all messages across the outbound edge that is not part of \( B_v \). A node is the root of a pruned sub-tree with probability \( \frac{1}{2}(1+p) \).

All messages generated by members of \( T_v \) are transmitted by \( v \) to \( s \). The number of nodes in \( T_v \) is equal to \( |W_v| \). Since a node is the root of a pruned sub-tree with probability \( \frac{1}{2}(1+p) \), then a node is a member of \( T_v \) with probability \( \frac{1}{2}(1-p) \). Note that \( v \) is a member of \( T_v \) with probability 1. When \( T_v \) consists of \( |W_v| = w \) nodes, the leaves of \( T_v \) each have two pruned children \((w+1) \text{ in total}\). Therefore, \( \mathbb{P}(|W_v| = w \in V_A) = \frac{2^{m+1}}{(m+1)!} \frac{1-p}{2} w-1 \frac{1+p}{2} w+1 \).

\[D. \text{ Proof of Proposition 7}\]

Given a graph \( G \) and observations after one epoch \( S \), an adversary can construct sets \( S_v \) at every adversary node. Each set \( S_v \) consists of the fresh messages forwarded by \( v \). As a worst-case assumption, suppose the adversary learns the one-to-one forwarding mappings, but not the edges over which honest nodes send their own messages. As a result, an adversary must first match each honest node \( u \in V_H \) with one of two possible sets \( S_u \) and \( S_u' \). Then, the adversary must match these honest nodes with messages in these sets. Let \( A_S \) denote the event in which \( u \) is matched to the correct set \( S_u \), and let \( A_S' \) denote the event in which \( u \) is matched to the incorrect set \( S_u' \).

The expected precision for \( u \in V_H \) is \( \mathbb{E}[D_{\text{mat}}(u)|G,S] = \mathbb{P}(A_S) \mathbb{E}[D_{\text{mat}}(u)|G,S,A_S] + \mathbb{P}(A_S') \mathbb{E}[D_{\text{mat}}(u)|G,S,A_S'] = \mathbb{P}(A_S) \mathbb{E}[D_{\text{mat}}(u)|G,S,A_S] = \mathbb{P}(A_S) \frac{1}{|S_u|}. \) The overall expected precision may be written as \( \frac{1}{|S_u|} \mathbb{P}(A_S) \mathbb{E}[D_{\text{mat}}(u)|G,S,A_S] \). Each term \( \frac{1}{|S_u|} \) occurs in the summation \( |S_u| \) times, which means that the expected precision over all graphs \( G \) may be written as \( \frac{2^n}{(n+p)\mathbb{P}(A_S)} \mathbb{P}(A_S) \mathbb{E}[D_{\text{mat}}(u)|G,S,A_S] \).

Note that \( \mathbb{P}(|S_u| = 0) \) is given by \( \mathbb{P}(|S_u| = 0) = \lim_{i \to 0} \mathbb{P}(T = i) \mathbb{P}(|S_u| = 0 | T = i) = \lim_{i \to 0} \left(1-p^i\right)^{\frac{1}{2}} = p^2 \frac{2^n}{1+p}. \) Thus, \( \mathbb{P}(|S_u| > 0) = 1 - p^2 \frac{2^n}{1+p} \), where \( \frac{1}{2} \leq \mathbb{P}(A_S) \leq 1 \), then

\[ \frac{2^n}{(n+p)(1+p)} \leq D_{\text{mat}} \leq \frac{2^n}{(1-p)n} \frac{1-p}{2} \frac{2^n}{1+p} \quad \frac{1+p}{2} \frac{2^n}{1-p^2} \leq D_{\text{mat}} \leq \frac{2^n+2^n}{(1-p)^2} - 4p^3. \]

\[E. \text{ Proof of Proposition 2}\]

The proof follows by identifying that the first spy node to receive a message along dandelion’s stem is a sufficient statistic for detection. Since the stem-phase occurs via peers’ outbound edges, \( \mathcal{P} \) and \( \mathcal{Q} \) have similar stem-phase propagation and differ only in the diffusion phase. As such precision and recall are not affected by how the message spreads in the diffusion phase.

For simplicity, let us assume a small \( q \), i.e., messages are always received by a spy node in the stem-phase before diffusion begins. For any message \( x \in \mathcal{X} \) let \( \mathcal{O}_x^1 \) be the random variable comprising a three-tuple \((U_h, U_a, T)\) where, in the stem-phase propagation of \( x \) (i) \( U_a \in \mathcal{V}_q \) is the first spy to receive \( x \), (ii) \( U_h \in \mathcal{V}_H \) is the honest peer that forwarded \( x \) to \( U_a \), and (iii) \( T \) is the time when \( x \) was received by \( U_a \). Next, let \( \mathcal{O}_x^2 \) denote the random variable that comprises of observations made by the adversary after it has been forwarded by \( U_a \) in the stem-phase. This includes all tuples \((u,v,t)\) such that honest peer \( u \in V_H \) forwarded \( x \) to \( v \in V_A \) at time \( t > T \). Lastly let \( \mathcal{O}_x \) denote the honest peer \( v \in V_H \) that is the source of transaction \( x \). To get a worst-case guarantee we also assume that the adversary has complete knowledge of the topology \( H \) of the anonymity graph.

Since the stem-phase propagation is over a line, we observe that once a message \( x \) reaches a spy node \( U_a \) for the first-time, the subsequent spreading dynamics depends entirely on the action taken by \( U_a \) (who it forwards \( x \) to, when it forwards etc.) and is conditionally independent of the past. This is true for every message \( x \in \mathcal{X} \). As such we have,

\[ \mathbb{P}(O_1^2, \ldots, O_n^2 | O_1^1, \ldots, O_n^1, Y_1, \ldots, Y_n, H) = \mathbb{P}(O_1^2, \ldots, O_n^2) \] \[ \mathbb{P}(O_1^1, \ldots, O_n^1, Y_1, \ldots, Y_n, H) \] \[ \Rightarrow \mathbb{P}(Y_1, \ldots, Y_n | O_1^1, \ldots, O_n^1, Y_1, \ldots, Y_n, H) = \mathbb{P}(Y_1, \ldots, Y_n | O_1^1, \ldots, O_n^1) \] \[ \Rightarrow \mathbb{P}(Y_1, \ldots, Y_n | O_1^1, \ldots, O_n^1, Y_1, \ldots, Y_n, H) = \mathbb{P}(Y_1, \ldots, Y_n | O_1^1, \ldots, O_n^1, H) \] \[ \forall i \in [n]. \]

Thus the posterior is conditionally independent of later observations, given stem-phase observations \( O_1^1, \ldots, O_n^1 \). Now, consider a network \( H' \) that is derived from \( H \) by removing all outgoing edges from adversarial peers. Since the observations \( O_n^1 \log \) transactions that have been received for the first time by a spy, it implies the routes taken by the transactions do not include any spy node. Hence the statistics of the stem-phase spreading are identical in \( H' \) and \( H \). Mathematically this implies,

\[ \mathbb{P}(Y_1, O_1^1, \ldots, O_n^1, H) = \mathbb{P}(Y_1, O_1^1, \ldots, O_n^1, H') \] \[ = \mathbb{P}(Y_1, O_1^1, \ldots, O_n^1 | H') = \mathbb{P}(Y_1, O_1^1, \ldots, O_n^1, H'). \]

From Theorems 3 and 4 in \( [13] \) we know that the optimal value of the expected precision and recall is a function of the posterior probabilities \( \mathbb{P}(Y_1, O_1^1, \ldots, O_n^1, H') \), which by combining Equations (20) and (21) in turn equals \( \mathbb{P}(Y_1, O_1^1, \ldots, O_n^1, H') \).

We finish the proof by applying the above results on networks \( \mathcal{P} \) and \( \mathcal{Q} \). Let \( H_P \) and \( H_Q \) denote the topologies of \( \mathcal{P} \) and \( \mathcal{Q} \) respectively; let \( H_P' \) and \( H_Q' \) denote the networks obtained by removing outgoing spy edges from \( \mathcal{P} \) and \( \mathcal{Q} \).
respectively. By construction we have $H_p = H_Q$. As such the two probability spaces, each comprising of the random variables $Y_{x_1}, \ldots, Y_{x_n}$, $O_1, \ldots, O_l$, $H'$, pertaining to the networks $P$ and $Q$ are identical. Hence we conclude the optimal values of precision and recall in the two networks must also be the same.

F. Proof of Proposition [2]

Proof: Let $v_i$ be the source of a message that propagates along a path $v_1, v_2, \ldots, v_k$ of length $k$. Let $\delta_{\text{hop}}$ be the delay incurred between each hop, and let $T_{\text{out}}(v_i)$ be the random timeout at node $v_i$ for $i = 1, \ldots, k$. Note that the message takes $\delta_{\text{hop}} \cdot k$ time to traverse $k$ hops and reach $v_k$. We desire that none of the $v_i$, $i = 1, \ldots, k$, initiate diffusion during this time with high probability. Since the random variables $T_{\text{out}}(v_i)$ are exponential, this probability can be bounded as

$$e^{-(k-1)\delta_{\text{hop}}/T_{\text{base}}} \cdot e^{-(k-2)\delta_{\text{hop}}/T_{\text{base}}} \cdots e^{-(k-k)\delta_{\text{hop}}/T_{\text{base}}} \geq 1 - \epsilon \Rightarrow T_{\text{base}} \geq \frac{-k(k-1)\delta_{\text{hop}}}{2\log(1-\epsilon)},$$

Thus $\hat{v}$ must also be the same.

G. Proof of Theorem [3]

Under dandelion spreading, for any message $x$, the recall-optimal estimator chooses the node $v$ that maximizes $P(X_v = x|O)$ (Theorem 4 [13]). Since we assume a uniform prior on sources, this is equivalent to a maximum-likelihood estimator that returns $\hat{v} = \arg\max_x P(O|X_v = x)$. From [13], we know that for a given adversarial mapping strategy, the expected recall is equivalent to the probability event.

Consider a TwistedPair source node $v$ that transmits a transaction $x$. In order to characterize the expected recall over all honest nodes in $V_D$, by symmetry, it is sufficient to compute the probability of detecting $v$ as the source of $x$, where the probability is taken over the spreading realization, randomness in the graph, and any randomness in the adversary’s estimator. This proof bounds the probability of detection for $v$ under version-checking and no-version-checking. Let $D$ denote the event where $v$ has at least one outbound neighbor that supports TwistedPair (i.e., $|D_v| > 0$), and let $\bar{D}$ denote the complement of that event.

Version-checking: For the lower bound, we have $P(\hat{v} = v) = P(\hat{v} = v|D)P(D) + P(\hat{v} = v|\bar{D})P(\bar{D}) \geq P(\hat{v} = v|D)P(D)$.

We can separately bound each of these terms:

$$P(D) = 1 - \frac{n-1-|V_D|}{n-1} \cdot \frac{n-2-|V_D|}{n-2} \cdots \frac{n-\eta-|V_D|}{n-\eta} \geq 1 - \left(\frac{n-1-|V_D|}{n-1}\right)^\eta = 1 - (1 - f)^\eta,$$

where $f = p + (1-p)\beta$ is the total fraction of nodes running TwistedPair, and $P(\hat{v} = v|D) \geq \frac{p}{f}$, since the first-spy estimator detects the true source if the first node in the stem is a spy node. Thus $P(\hat{v} = v) \geq \frac{p}{f} (1 - (1 - f)^\eta)$.

To compute the upper bound, we have

$$P(D) \leq 1 - \left(\frac{n-1-|V_D|}{n-\eta}\right)^\eta = 1 - (1 - \frac{f n}{n-\eta})^\eta,$$

$$P(\bar{D}) \leq \left(\frac{n-1-|V_D|}{n-1}\right)^\eta = (1 - f)^\eta.$$

Trivially, $P(\hat{v} = v|D) \leq 1$. To bound $P(\hat{v} = v|D)$, we condition on the event $S$, where $v$’s first node in its TwistedPair stem (call it $w$) is a spy node. The first-spy estimator is recall-optimal if there is a spy node in the stem (Theorem 4 in [13]). We have $P(\hat{v} = v|D) = P(\hat{v} = v|D, S)P(S|D) + P(\hat{v} = v|D, \bar{S})P(\bar{S}|D)$, and

$$P(\hat{v} = v|D, S) \leq 1, P(S|D) = \frac{p n}{f n - 1}, P(\bar{S}|D) \leq 1 - \frac{p}{f}.$$

To bound $P(\hat{v} = v|D, \bar{S})$, we condition on $F$, the event where $w$ chooses to extend the stem. $P(\hat{v} = v|D, S) = P(\hat{v} = v|F, D, S)P(F|D, S) + P(\hat{v} = v|F, D, \bar{S})$. For this upper bound, we also assume that an oracle gives the adversary the source of the diffusion process in the spreading phase. That is, let $\ell_1(x), \ldots, \ell_M(x)$ denote the stem nodes associated with transaction $x$; in this case, $\ell_1(x) = v$, and $M_z$ denotes the length of the stem. We assume an oracle gives the adversary $\ell_M(x)$. If $w$ chooses not to extend the stem (event $\bar{F}$), then the $\ell_M(x) = w$. We also assume that the adversary knows $V_D$, the set of nodes running TwistedPair. Hence, we have $P(F|D, S) = 1 - q$ and $P(F|D, \bar{S}) = q$. We now wish to bound $P(\hat{v} = v|F, D, S)$ and $P(\hat{v} = v|F, D, \bar{S})$—the probabilities of detection given that $w$ passed the message to honest TwistedPair neighbor $w$, conditioned on $w$’s decision to either extend or terminate the stem, respectively.

Recall that if there is a spy in the stem (e.g., $\ell_i(x) \in V_A$ for some $i \in \{M_z\}$), then the first-spy estimator is recall-optimal. If there are no spies in the stem (i.e., when $V_A \cap \{\ell_1(x), \ldots, \ell_M(x)\} = \emptyset$, $\ell_i(x) \rightarrow \ell_{M_z}(x) \rightarrow O$ form a Markov chain. Since none of the stem nodes are spies, the spy observations are conditionally independent of the source node given $\ell_{M_z}(x)$. Since the adversary learns $\ell_{M_z}(x) = \ell$ exactly from the oracle, the recall-optimizing strategy becomes $\hat{v} = \arg\max_x P(\ell_{M_z}(x) = \ell | X_v = x)$.

Part 1: To bound $P(\hat{v} = v|F, D, S)$, note that $\ell_{M_z}(x) = w$. Suppose that in addition to revealing $w$, the oracle also tells the adversary that $w$ is not the true source. Consider the set $R = \{u \in V_D | w \in D_u\}$, which contains all nodes that could have feasibly relayed a TwistedPair transaction to $w$. For nodes $u \in R$, the likelihood of each node being the source is $P(\ell_{M_z}(x) = w|X_u = x) = \frac{1}{\min_{u \in R}|D_u|}(q + (1-q)\delta)$ where $\delta$ is the probability that the stem, having reached $w$ without terminating, loops back to $w$ and terminates at some later hop. Conditioned on the stem passing through $w$, the rest of the stem is independent of the source, so $\delta$ does not depend on $u$. Let $\tilde{v}$ denote the most likely source among the nodes in $R$. $\tilde{v} = \arg\max_{u \in R} P(\ell_{M_z}(x) = w|X_u = x) = \arg\min_{u \in R}|D_u|$. It is straightforward to show that for any alternative source $z = \tilde{v}$, $z \in V_H$ has a lower likelihood. This follows trivially if $z \in R$. If $z \notin R$, the stem from $z$ to $w$ would require at least two hops, which reduces the likelihood of candidate source $v^*$ by a factor of at least $(1-q)$. Hence $\tilde{v}$ is the ML source estimate, and we want to know $P(\hat{v} = v)$. 17
Since each node $u$ in $R$ is equally likely to have the smallest $D_u$ set, we have $\mathbb{P}(\tilde{v} = v | F, D, S) \leq \mathbb{E}[\frac{1}{Z}]$. We know that $|R| \geq 1$, because $v$ is connected to $w$ by construction. Each node chooses its $\eta$ connections independently, this can be computed as $\mathbb{E}[1/(Z + 1)]$ where $Z \sim \text{Binomial}(n - 1, \phi)$, and $p$ is the probability of any given node choosing $w$ as one of its outbound edges. We can compute $\phi$ as $\phi = 1 - \frac{n^2}{n^2 - 2n^2 - 3 \cdots - n^{2-\eta}}$. Henceforth, we will abuse notation and take $\phi = 1 - 1 - \frac{1}{n-\eta}$. By using this lower bound on $\phi$, we are reducing the probability of any given node choosing $w$ as an outbound edge, and thereby increasing the overall probability of detection. Given that, it is straightforward to show that

$$\mathbb{P}(\tilde{v} = v | F, D, S) \leq \mathbb{E}\left[1 \over Z + 1\right] = 1 - (1 - \phi)^{\hat{n}} \triangleq \zeta. \quad (22)$$

**Part 2:** We want to show that $\lim_{n \to \infty} \mathbb{P}(\tilde{v} = v | F, D, S) = 0$; if this is the case, then the asymptotic inequality in Theorem 3 holds for any $C' > 1$. There are three ways the adversary can identify the correct source $u$ under conditions $F$, $D$, and $S$: 1) the stem eventually loops back to $v$, and then transmits to a spy node (we call this event $A$), 2) the stem loops back to $v$, which terminates the stem (event $B$), or 3) the stem terminates before reaching a spy node, and the set of TwistedPAIR nodes with outbound edges to the stem’s terminus $\ell$ includes $v$ (event $C$). Note that we are still assuming the adversary learns the last stem node $\ell$. Also, $B$ and $C$ are not sufficient conditions for detection, but they do guarantee a nonzero probability of detection under a recall-optimal estimator. Therefore, since these events are disjoint, $\mathbb{P}(\tilde{v} = v | F, D, S) \leq \mathbb{P}(A) + \mathbb{P}(B) + \mathbb{P}(C)$; we can individually bound each of these events.

To bound $\mathbb{P}(A)$, we first compute the probability of the stem loop back to $v$ without reaching any spy nodes or terminating (we call this event $A'$); since this event is necessary but not sufficient for detection under event $A$, we have $\mathbb{P}(A) \leq \mathbb{P}(A')$. $\mathbb{P}(A')$ can be upper bounded by relaxing the restriction of not hitting any spy nodes before reaching $v$. So we just want the probability of the stem looping around and reaching $v$ before terminating. We let $M_x$ denote the stem length (ignoring the first two hops of $v$ and $w$); it is a geometric random variable with parameter $q$. We let $T_v$ denote the random number of hops before hitting node $v$, given starting point $w$ (the number of hops excludes $w$). Thus $\mathbb{P}(A') \leq \mathbb{P}(T_v \leq M_x)$. As an upper bound, we assume all nodes run TwistedPAIR. Hence, each node has an equal probability $\frac{1}{n-1}$ of forwarding the message to $v$, so $T_v \sim \text{Geometric}(\frac{1}{n-1})$. Then $\mathbb{P}(T_v \leq M_x) = \sum_{i=1}^{\infty} \mathbb{P}(M_x = i) \mathbb{P}(T_v \leq i) = \sum_{i=1}^{\infty} q(1 - q)^{i-1}(1 - (1 - \frac{1}{n} - q)^i) = 1 - q(1 - \frac{1}{n} - q)^i(1 - 1 - q (1 - 1 - (1 - \frac{1}{n} - q)^{i-1} - q)^{i-1} i^2 = 1 - q \eta = 0$, so $\lim_{n \to \infty} \mathbb{P}(A) = 0$. The same argument applies for event $B$, so $\lim_{n \to \infty} \mathbb{P}(B) = 0$.

For event $C$ we have $\mathbb{P}(C) = \sum_{u \in V_n \setminus \{v\}} \mathbb{P}(\hat{\ell}_M(x) = u | D_v) = \mathbb{P}(\hat{\ell}_M(x) = w) + \mathbb{P}(\hat{\ell}_M(x) = w) | u \in V_n \setminus \{v, w\} \leq \mathbb{P}(\hat{\ell}_M(x) = w) + \frac{\gamma}{\gamma} \quad (24)$

where (24) holds because we already know that $w \in D_v$ by definition, and since we are conditioning on event $D$, $v$ will never relay the message to a non-TwistedPAIR node. (24) holds because each TwistedPAIR node other than $w$ is equally likely to be in $D_v$, so for $u \in V_n \setminus \{v, w\}$, $\gamma \triangleq \mathbb{P}(u \in D_v)$ does not depend on $u$. By the same logic as before, $\lim_{n \to \infty} \mathbb{P}(\hat{\ell}_M(x) = w) = 0$. Hence we only need to show that $\lim_{n \to \infty} \gamma = 0$. We can write out $\gamma = \frac{(n-2)}{n-\eta} = \frac{n-1}{n-1+2}$, so $\lim_{n \to \infty} \gamma = 0$. Given this, we have that $\lim_{n \to \infty} \mathbb{P}(\tilde{v} = v | F, D, S) = 0$.

Combining the two parts gives $\mathbb{P}(\tilde{v} = v | D, S) \leq \zeta q$. \quad (25)

Overall, we have $\mathbb{P}(\tilde{v} = v) \leq 1 - 1 - \frac{1}{n-\eta} \leq \frac{(n-1)}{n-\eta} \frac{pn}{n-\eta} + (1 - \frac{q}{q}) q \zeta + (1 - f)^{q}$. Taking the limit as $n \to \infty$ gives $1 - 1 - \frac{1}{n-\eta}$ $\frac{p}{p} + (1 - \frac{q}{q}) q \zeta + (1 - f)^{q}$. The claim follows.

**No-version-checking:** The lower bound comes directly from Theorem 2 in [13].

To prove the upper bound, we first condition on whether $v$’s selected stem relay $w$ is a spy node; as before, $S$ denotes this event. In this section, we will repurpose our previous notation and use $D$ to denote the event where the selected relay supports TwistedPAIR. Again, we will assume that the adversary knows the underlying graph $H$, $V_D$, and the final stem node $\ell_M(x)$. We have

$$\mathbb{P}(\tilde{v} = v) = \mathbb{P}(\tilde{v} = v | S) \mathbb{P}(S) + \mathbb{P}(\tilde{v} = v | S) \mathbb{P}(S) \quad (26)$$

To bound $\mathbb{P}(\tilde{v} = v | S)$, we condition on $D$: $\mathbb{P}(\tilde{v} = v | S, D) = \mathbb{P}(\tilde{v} = v | S, D) \mathbb{P}(D) \mathbb{P}(S)$. 

$$\mathbb{P}(\tilde{v} = v | S, D) \leq \frac{\beta(1 - \beta)n}{(1 - \beta)(1 - p)n - 1} = \frac{(1 - \beta)(1 - p)n}{(1 - \beta)(1 - p)n - 1} \quad (27)$$

$$\mathbb{P}(\tilde{v} = v | S, D) \leq \zeta q \quad (28)$$

where (27) follows from (25), and (28) follows from (22) by assuming the adversary is told that $w$ is not the true source. Combining gives $\mathbb{P}(\tilde{v} = v | S) \leq \zeta \frac{q \beta n}{n - 1} + \frac{(1 - \beta)(1 - p)n}{n - 1} = \frac{\zeta (1 - \beta)(1 - q)n}{n - 1}$. Plugging into (26) gives $\mathbb{P}(\tilde{v} = v) \leq \frac{pn}{n - 1} + \frac{(1 - p)n}{n - 1} \zeta + (1 - f)^{q}$. Taking the limit as $n \to \infty$ gives $1 - 1 - \frac{1}{n-\eta}$ $\frac{p}{p} + (1 - \frac{q}{q}) q \zeta + (1 - f)^{q}$. The claim follows.