ethp2psim: Evaluating and deploying privacy-enhanced peer-to-peer routing protocols for the Ethereum network

Ferenc Béres, István András Seres, Domokos M. Kelen, András A. Benczúr

March 3, 2023

Abstract

Network-level privacy is the Achilles heel of financial privacy in cryptocurrencies. Financial privacy amounts to achieving and maintaining blockchain- and network-level privacy. Blockchain-level privacy recently received substantial attention. Specifically, several privacy-enhancing technologies were proposed and deployed to enhance blockchain-level privacy. On the other hand, network-level privacy, i.e., privacy on the peer-to-peer layer, has seen far less attention and development.

In this work, we aim to provide a peer-to-peer network simulator, **ethp2psim**, that allows researchers to evaluate the privacy guarantees of privacy-enhanced broadcast and message routing algorithms. Our goal is two-fold. First, we want to enable researchers to implement their proposed protocols in our modular simulator framework. Second, our simulator allows researchers to evaluate the privacy guarantees of privacy-enhanced routing algorithms. Finally, **ethp2psim** can help choose the right protocol parameters for efficient, robust, and private deployment.

1 Introduction

Ethereum is the most popular public blockchain according to the number of issued transactions. Ethereum's public ledger is completely transparent: every accounts' balance and transaction history is visible to everyone. In numerous applications, this level of transparency is undesirable. There are numerous privacy-enhancing technologies described in the literature: mixers [13, 15, 18], stealth addresses [20], and confidential transactions [5, 22]. Some of these protocols had been already deployed and are continuously serving the Ethereum community by enhancing their privacy on the blockchain [2, 21].

However, blockchain privacy on its own is not enough to achieve financial privacy. Users should be able to broadcast transactions in a privacy-preserving manner. Specifically, no adversary should be able to link cryptocurrency addresses to their actual users' unique identifiers, e.g., IP addresses, by logging network-level communication between Ethereum full nodes. This problem is even more pronounced in proof-of-stake Ethereum where consensus participants need to regularly broadcast privacy-critical messages such as blocks and attestations. Previous work has already shown various attacks on privacy that exploited peer-to-peer (P2P) information [3, 8, 12]. To enhance the privacy of cryptocurrency users, several privacy-enhanced broadcast and routing algorithms were proposed [4, 7, 14].

In this work, we provide the following contributions.

- We develop an open-source simulator, **ethp2psim**, that allows anyone to implement and evaluate various privacy-enhanced broadcast and message routing algorithms.
- Using our simulator, we evaluate the privacy guarantees of existing privacy-enhanced message routing algorithms on the Ethereum P2P layer.
- We identify several trade-offs between privacy, robustness, and efficiency. The quantification of these trade-offs can inform the deployment of these protocols.

The rest of this paper is organized as follows. In Section 2, we present the related work on privacy on the P2P layer of cryptocurrencies. In Section 3, we introduce the pertinent background knowledge on privacy-enhanced message routing algorithms. In Section 4, we describe our system and threat model of Ethereum's P2P layer. We introduce **ethp2psim** in Section 6 and evaluate state-of-the-art routing protocols in Section 7. Finally, we conclude our work in Section 8.

2 Related work

2.1 Privacy-enhanced transaction broadcasting

Dandelion [4] and Dandelion++ [7] are the two most important related works. In Dandelion(++), users (nodes in the P2P graph) do not directly broadcast their transactions. Rather, each user flips a biased coin and with probability p they broadcast their transaction while with probability $1-p$ they forward their message to a single randomly selected neighbor. Dandelion(++) offers a privacy-enhanced routing algorithm with minimal implementation complexity. Specifically, Dandelion is implemented and deployed on the Monero network. In our work, we also adapted ideas from the mixnet literature, specifically, onion routing [16] and applied it in the context of PoS Ethereum.

2.2 Evaluating privacy-enhanced routing protocols

Serena et al. observed that privacy-preserving routing algorithms incur larger delays and less robustness on message delivery [17]. Sharma et al. evaluated the anonymity guarantees of several anonymity schemes used by cryptocurrencies (e.g., Dandelion and Dandelion++), and found they provide little to no anonymity guarantees [19]. In their work, they also build a simulator along the way to evaluate previous routing protocols. However, their simulator is not modular and does not take into account numerous Ethereum-specific subtleties that severely change the anonymity analysis.

3 Preliminaries

In this section, we present the pertinent background knowledge on Ethereum's P2P protocol.

3.1 Participants

We assume that the following participants run full nodes in the Ethereum P2P network. These participants can be cast into two main categories: validators (i.e., attestors, aggregators, block proposers) and regular users. The main difference is that regular users do not participate in the consensus algorithm, hence, they only broadcast regular transactions and not consensus-critical messages. For each type of participants, our goal is to devise a protocol that allows no adversary to link the message (e.g., attestation, block or transaction) to its originator.

Attestors. Every validator needs to broadcast an attestation in each epoch once, in a specific slot.

Aggregators. Designated attestors collect several attestations, i.e., BLS signatures, and batch them. Afterwards they forward the batched signatures to their neighbors

Block proposers. Validators are selected to propose new blocks according to their amount of stake. Since the stake distribution is public information, we cannot hope to achieve larger degree of anonymity than the Shannon-entropy of the stake distribution.

Users. Users occasionally broadcast transactions. They are not necessarily validators.

3.2 The topology of the Ethereum P2P network

Ethereum's P2P graph is a dynamically changing graph. It is a permissionless network, i.e., nodes can join or leave the network whenever they want. Therefore, it is nearly impossible to obtain a precise description of the Ethereum P2P graph at any given time. Our simulations rely on publicly available static measurements (i.e., snapshots) of this dynamically evolving graph.

In its simplest form, one might assume that the Ethereum P2P graph is a random regular graph, since the default Ethereum client randomly selects a fixed number of peers to connect with. However, the real graph might look different as users can modify their client software's P2P settings. We model Ethereum's P2P graph as a weighted, directed graph, where weights on the edges denote the pairwise latencies on that particular connection. Gencer et al. [9] measured the peer-to-peer latency of Ethereum peers. They found that Ethereum's P2P latencies follow a distribution with $171ms$ average and a standard deviation of $76ms$. The Ethereum P2P graph is measured and described by Cortes and Bautista [6]. They found that the Ethereum2 P2P graph is a heavily centralized network that exhibits a spoke-hub distribution. We were also able to obtain a snapshot of the Goerli testnet, which we believe is sufficiently similar to the mainnet P2P topology. In our simulations, we used the aforementioned results of these measurement studies to approximately model Ethereum's P2P network.

4 Modelling Ethereum's P2P layer

4.1 Threat Model

We assume a local active adversary, i.e., an adversary can only observe communication that passes through them. It is well-known that one can only have anonymous communication against global active adversary if mixnets are employed. In the case of Ethereum, the applicability of mixnets is questionable as they incur considerable latencies that is detrimental to achieving consensus. Therefore, we do not consider the global active adversary model.

4.2 Anonymity Goals

We want to achieve the following anonymity goals by applying privacy-enhanced message routing protocols.

Unlinkability Our main goal is that the adversary should not be able to link on-chain (e.g., Ethereum addresses) and off-chain (e.g., IP addresses) identities. A stronger anonymity notion would be sender unlinkability, i.e., the adversary should not be able to tell whether two messages were sent by the same user or not. Currently, there is no message routing protocol that would achieve this stronger notion of privacy.

Robustness A privacy-enhanced message routing protocol must be robust. In particular, messages must reach the vast majority of the network even if a considerable fraction of the network is adversarial, and timeliness should not be impacted by possible network issues in a small number of nodes.

Latency Ethereum's PoS consensus protocol demands low latency of every participant. Participants with high network latency might miss rewards or worse may be punished by not broadcasting certain messages, i.e. inactivity leak.

Long-term feasibility Any privacy-enhancing technique should be studied by considering its properties on long time scales, as participants might use the same off- and on-chain identities for years.

Spam prevention When messages are forwarded in plain text, it is relatively easy to filter out spam messages. However, if we want to move towards routing protocols that forward encrypted messages, then it is more challenging to fend off spam messages. Even in the latter case, we want to prevent spam messages to be forward on the P2P layer.

Ease of implementation Ideally, we want to apply a message routing protocol that is easy to implement and deploy, as implementation complexity inherently implies an increased attack surface.

5 P2P privacy solutions

5.1 Dandelion and Dandelion++

Dandelion [4] and Dandelion++ [7] are the primary hop-by-hop routing algorithm to enhance privacy on the P2P level. In a nutshell, in both of these protocols messages can be in two phases: stemming (or sometimes anonymity) and spreading (sometimes broadcast) phase. In the stemming phase, nodes flip a biased coin and decide whether forward to message to a random neighbour, i.e., continue with the stem phase, or broadcast the message. Dandelion is implemented and deployed by the Monero cryptocurrency. Recently it was shown by Sharma et al. [19] that these protocols provide little to no anonymity guarantees in most realistic scenarios.

Even worse, for both of these protocols, there exists also an Ethereum-specific timing side-channels attack. In this attack, an adversary might be able to infer its position in the anonymity phase of a given message in knowledge of slot starting times. This valuable information can lead to loss of anonymity in these protocols.

5.2 Onion routing

Our solution to Ethereum P2P network-level privacy is an onion routing based message protocol that we detail in a separate document [11]. Here we just remark, that a crucial difference between hop-by-hop routing and onion routing is that in onion routing all messages are encrypted during the anonymity phase, while they are plain-text during the broadcast phase. The advantage of this approach is that the adversary cannot know whether two cyphertexts belong to the same message. However, major challenges are robustness, latency, and spam protection.

An onion-routing inspired P2P privacy protocol for Ethereum can take two different paths. First, it would either use the public Tor network [10] or implement and deploy an integrated Tor-like overlay network on top of the currently existing Ethereum P2P graph. There are pros and cons. We summarize theoretical anonymity guarantees and implementation challenges in more depth in a separate document [11].

6 ethp2psim: a peer-to-peer network simulator for Ethereum

In this section, we describe in detail our **ethp2psim** simulator. The simulator is open-source and available at https://github.com/ferencberes/ethp2psim. This simulator aims to enable researchers to implement privacy-enhanced message routing protocols and evaluate their anonymity guarantees in the context of PoS Ethereum.

Our simulator is written in Python and follows object-oriented design patterns. The simulator consists of the following main classes that are also desribed in the related documentation[1].

6.1 Network

This class defines the P2P network in which users can evaluate their protocols. The default P2P graph is regular (see Listing 2). However, the simulator can work with user-defined P2P graphs as well. Currently, we use a P2P graph obtained from the Goerli testnet as shown in Listing 1.

```
from ethp2psim.network import *
nw_gen = NodeWeightGenerator('stake')
ew_gen = EdgeWeightGenerator('normal')
goerli = GoerliTestnet()
net = Network(nw_gen, ew_gen, graph=goerli.graph)
```

Listing 1: Initialize a P2P network based on Goerli testnet data. Channel latencies (edge weights) are sampled from a normal distribution while node weights are proportional to the distribution of staked Ether values.

6.2 Message

This class describes the messages that are forwarded and broadcast between P2P participants. We think of messages as attestations, blocks, or transactions. We abstract away the metadata of messages, e.g., message type or size.

[1] https://ethp2psim.readthedocs.io/en/latest/?badge=latest

6.3 Protocol

This class defines and implements the various message-spreading protocols we want to evaluate. Currently, we implemented and considered a simple broadcast to all algorithms, Dandelion [4], Dandelion++ [7], and a variant of our Onion routing based algorithm [11].

Since our simulator is highly modular (see Listing 2), we hope to receive contributions from the community to evaluate novel message-spreading algorithms.

```python
from ethp2psim.network import *
from ethp2psim.protocols import BroadcastProtocol
from ethp2psim.adversary import Adversary
nw_gen = NodeWeightGenerator('stake')
ew_gen = EdgeWeightGenerator('normal')
# random 3 regular graph with 10 nodes
net = Network(nw_gen, ew_gen, 10, 3)
protocol = BroadcastProtocol(net, broadcast_mode='all')
# adversary controls a random 10% of P2P network nodes
adversary = Adversary(protocol, 0.1)
# message originating from node 0
msg = Message(0)
msg.process(adversary)
# message was sent to 3 neighbors of node 0
print(len(msg.queue))
# output: 3
```

Listing 2: First, we initialize a random regular P2P network along with a broadcast to all protocol and the related adversary. Next, we create a message originating from node 0 and we start to propagate it on the P2P network by calling the *msg.process* function.

6.4 Adversary

This class contains the implementation of various adversarial strategies aiming to reduce anonymity guarantees provided by the implemented routing protocols. At the time of writing, we have implemented several general- and protocol-specific adversarial strategies.

Regarding the positioning of the corrupted nodes, the adversary might corrupt nodes randomly in the P2P graph or selectively targeting central nodes (e.g., nodes with high degree or Betweenness) as seen in Listing 3. Therefore, we can evaluate which faults cause more significant anonymity losses.

```python
from ethp2psim.network import *
from ethp2psim.protocols import BroadcastProtocol
from ethp2psim.adversary import Adversary
seed = 42
```

```python
# Generate random Barabási-Albert graph with 20 nodes
G = nx.barabasi_albert_graph(20, 3, seed=seed)
nw_gen = NodeWeightGenerator('stake')
ew_gen = EdgeWeightGenerator('normal')
# use the Barabási-Albert graph as P2P network
net = Network(nw_gen, ew_gen, graph=G, seed=seed)
protocol = BroadcastProtocol(net, broadcast_mode='all', seed=seed)
# select 4 highest degree nodes to be adversaries
adv_nodes = net.get_central_nodes(4, 'degree')
# initialize adversary
adversary = Adversary(protocol, adversaries=adv_nodes, seed=seed)
print(adversary.nodes)
# output: [5, 0, 4, 6]
```

Listing 3: Four nodes with highest degree are set to be adversarial nodes from a random Barabási-Albert graph[2]. Note that by setting the random seed you can get reproducible results.

6.5 Simulator & Evaluator

The simulator class defines the experiments we wish to run on a specific P2P graph topology with a particular routing algorithm against a specific adversarial strategy. After simulating multiple messages, we can use the Evaluator to calculate the deanonymization power of the adversary with respect to various performance metrics:

- **hit ratio:** The fraction of messages where the adversary correctly identified the originator.
- **inverse rank:** We take the average of $\frac{1}{r_m}$ values over the messages where r_m is the rank of the originator for a given message m in the adversary's ordered list of possible candidates for the originator of m.
- **entropy:** the average of Shannon entropies over all messages.
- **NDCG**: the average normalized discounted cumulative gain [1] over all messages. Basically, we take the average of $\frac{1}{log_2(1+r_m)}$ as only the originator is considered to be the single relevant hit among the candidates.

The generated report includes the average ratio (*message_spread_ratio*) of nodes reached by messages in the P2P network.

```python
from ethp2psim.network import *
from ethp2psim.protocols import DandelionProtocol
from ethp2psim.adversary import DandelionAdversary
```

[2] https://networkx.org/documentation/stable/reference/generated/networkx.generators.random_graphs.barabasi_albert_graph.html

```python
from ethp2psim.simulator import Simulator, Evaluator
seed = 42
# Sample staked ether amounts as node weights
nw_gen = NodeWeightGenerator('stake')
# Sample channel latencies from a normal distribution
ew_gen = EdgeWeightGenerator('normal')
net = Network(nw_gen, ew_gen, num_nodes=100, k=20, seed=seed)
# A message is broadcasted with 40% probability in the stem (anonymity) phase of Dandelion
protocol = DandelionProtocol(net, 0.4, broadcast_mode="sqrt", seed=seed)
# Adversary controls 10% of all nodes and it is in knowledge of the line graph
adversary = DandelionAdversary(protocol, 0.1, active=False, seed=seed)
# 20 random messages are simulated where originators are selected with recpest to their stakes
simulator = Simulator(adversary, num_msg=20, use_node_weights=True, verbose=False, seed=seed)
simulator.run()
# The first sent heuristic is used to determine first broadcaster for each message
evaluator = Evaluator(simulator, estimator='first_sent')
print(evaluator.get_report())
# output: {'estimator': 'first_sent', 'hit_ratio': 0.2, 'inverse_rank': 0.35,
# 'entropy': 2.05, 'ndcg': 0.48, 'message_spread_ratio': 1.0}
```

Listing 4: Evaluating the efficacy of an adversary controling 10% of P2P network nodes against Dandelion. The main goal of the adversary is to determine the originator node for each of the 20 simulated message.

7 Evaluation

In this section, we evaluate the privacy guarantees of various network anonymity protocols. Furthermore, we analyze the effect of different network topologies and active adversaries on privacy and robustness. We detail the following four experimental measurements.

7.1 Deanonymization performance

First, let's start with a simple experiment where we compare the deanonymization power of the adversary when it uses the first-reach or the first-sent heuristics to determine the originator for each message. These estimator strategies are used to guess the first node that broadcasted a given message based on the observations of all adversarial nodes. In short, an adversary using the first-reach heuristic predicts a node to be the first broadcaster if it is the first node that it heard the message from. On the other hand, using the channel latency information of adjacent channels, a first-sent estimator tries to identify the neighbor that first sent the message to any of the adversarial nodes. Naturally, the two predictions might not coincide as the triangle inequality does not necessarily hold for P2P network latency.

In this experiment, we use a random regular graph with 1000 nodes and 50 degree to compare the two heuristics against multiple protocols. Not surprisingly,

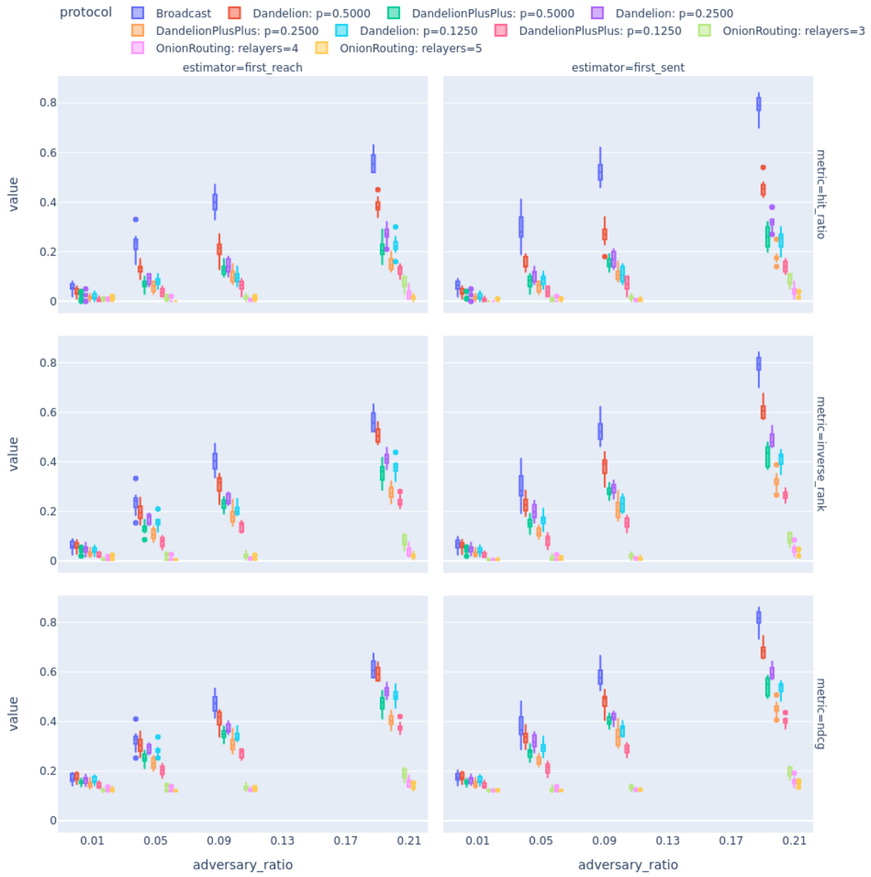

Figure 1: Evaluation of the first sent and first reach estimators for three performance metrics (i.e., hit ratio, inverse rank, NDCG) with respect to multiple protocols (colors) and adversary ratios (x-axis).

our results in Figure 1 show that the adversary with the first-sent estimator performs significantly better. However, we highlight that only the hit ratio, inverse rank, and normalized discounted cumulative gain [1] (NDCG) can reflect this behavior where ground truth information about message sources is compiled into the evaluation.

Unfortunately, entropy does not depend on the ground-truth. It only measures the uncertainty of the predicted distribution, but not its closeness to the ground-truth. Nevertheless, the entropy for Dandelion++ is higher than for Dandelion in Figure 2. The prediction entropy for broadcast to all and our Onion Routing based protocol is zero as the predicted distribution only contains the most likely candidate. A possible future work could include additional less-likely candidates

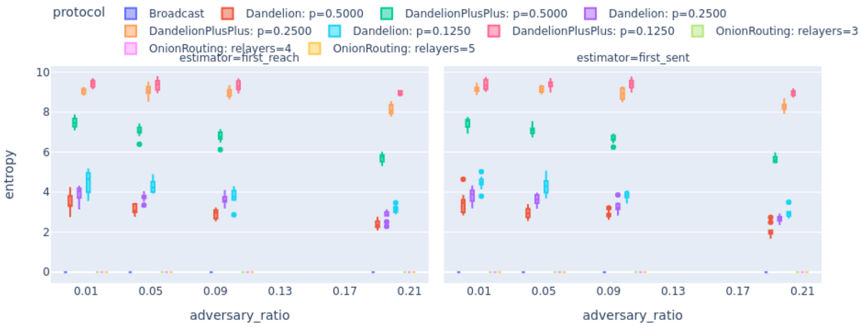

Figure 2: The Shannon entropy of the first sent and first reach estimators for various message spreading algorithms.

as well in the prediction distribution, this way better reflecting the knowledge of the adversary.

In Figure 1, it is interesting to see how Dandelion can confuse the adversary compared to simple broadcasting in terms of hit ratio (e.g., first-sent performance drops from 0.5 to 0.3 in case of 10% adversarial nodes) which might indicate that it is overly restrictive, as it doesn't contain information about much of the predicted distribution. Instead, *our recommendation is to use inverse rank or NDCG for evaluation.* These metrics can better reflect that despite the higher uncertainty introduced by Dandelion(++) the adversary can still make a good educated guess in knowledge of the current anonymity graph (i.e., line-graph for Dandelion). For example, in Figure 1, it is quite shocking to see the change in inverse rank from 0.5 to 0.4, which means that on average Dandelion improves only half a rank for the predicted message source, in the case of 10% adversarial nodes.

7.2 Different network topologies

In Figure 3, we observe how different graph topologies, such as a random regular graph and a scale-free graph (Goerli testnet's topology), affect the adversary's deanonymization power measured by various different metrics (e.g., hit ratio, inverse rank, NDCG). The deanonymization performance is displayed with respect to the ratio of adversarial nodes (see the x-axis) in the P2P network.

Here, we make four main observations related to privacy:

- The achieved privacy is quite brittle in the case of 0.2 adversary ratio: 0.5 *inverse_rank* for Dandelion with 0.5 broadcast probability means that the adversary outputs a vector of candidates and on average the true originator is put to the 2nd place.
- Dandelion(++) with the least broadcasting probability ($p = 0.125$) provides the highest privacy among the considered Dandelion(++)-style pro-

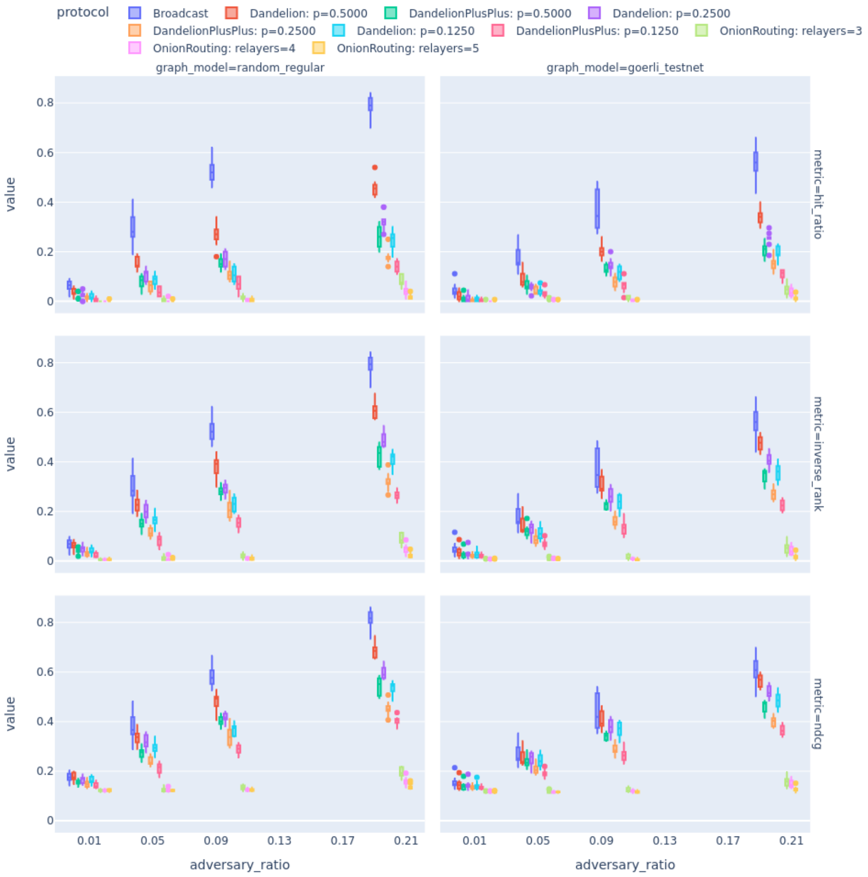

Figure 3: The effect of different network topologies on the adversaries' deanonymization power.

tocols.

- The results are promising for our Onion Routing based protocol where the efficiency of the adversary is less affected by the ratio of adversarial nodes in the P2P network.
- In general the Goerli testnet exhibits more privacy across all metrics.

7.3 Broadcast settings

Next, in Figure 4, observe the significant change in the results when a message is propagated to all neighbors, instead of a random square root of them (as we did it in former experiments), during the broadcast phase. It is quite shocking

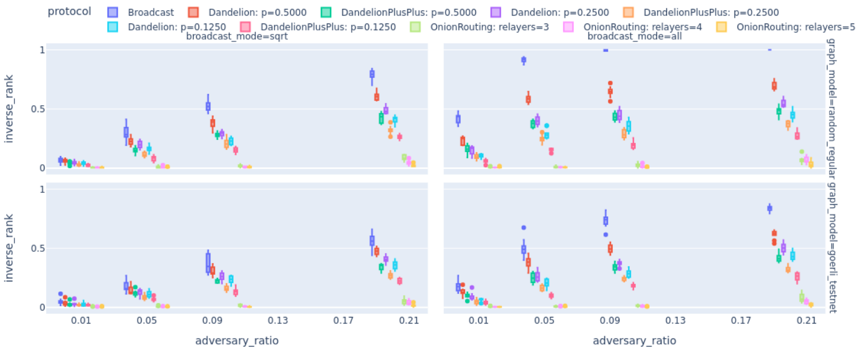

Figure 4: The effect of different broadcast settings on the adversaries' deanonymization power measured in inverse rank.

that an adversary controlling 10% of all nodes can be almost sure about the identity of the message source in the case of a simple broadcast protocol. Clearly, Dandelion(++) can significantly decrease the deanonymization performance of the adversary but it has a high price in terms of robustness detailed in the next section.

7.4 Robustness against active and passive adversaries

In our next experiment, we consider two types of adversaries. A passive adversary that follows the protocol and only logs the timestamp information when its nodes encounter messages. We also implemented an active adversary that does not forward messages at all. In Figure 5, we show that this is especially problematic for Dandelion(++). Imagine that an active adversary sits in the stem (anonymity) phase of Dandelion(++). Basically, if a message encounters an adversarial node on the line graph then it will be never broadcasted. The more and more adversaries censor messages the larger the portion of messages that are not heard by nodes in the P2P network. This is even more concerning, when the high-degree nodes are compromised (e.g., *adversary_centrality*='degree'). Note that the random regular graph is more robust against (active) adversaries than the Goerli testnet.

In Figure 6, once again, we see the low levels of privacy (measured in inverse rank in this figure) provided by various privacy-enhanced routing algorithms. It is easy to see that in our setting active and passive adversaries have the same power to deanonymize messsages. Deanonymization results are slightly better for the Goerli testnet's topology, i.e., the adversary is less powerful on a scale-free graph. In our experiments, the random regular graph has a higher edge density, hence, the adversary can make a more informed guess about the originator of the messages.

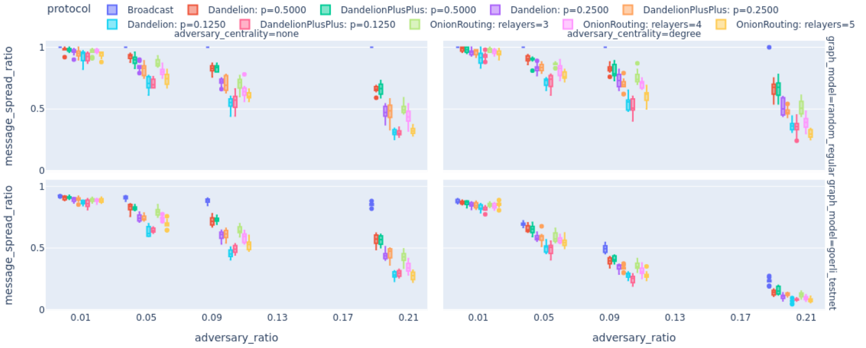

Figure 5: Messages reach fewer nodes if central nodes actively block them.

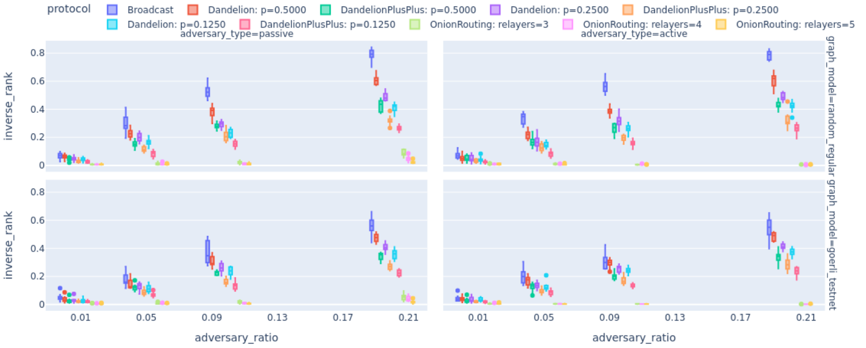

Figure 6: Active and passive adversaries have the same power to deanonymize messsages. Deanonymization results are slightly better for the Goerli testnet's topology.

8 Conclusion

In this work, we developed **ethp2psim** a network privacy simulator of Ethereum's P2P layer that allows the development, testing, and evaluation of privacy-enhanced message routing algorithms such as Dandelion(++), onion routing, or any custom-specified algorithm. We considered several existing privacy-preserving routing algorithms and evaluated their privacy guarantees using this simulator.
Our simulator is open source and open to contributions. It can be enhanced in several ways.

- Due to the modular nature of **ethp2psim**, one might add new privacy-preserving algorithms and novel adversarial strategies to our package.
- We considered several notions of privacy such as Shannon entropy, the

ratio of deanonymized messages (*hit_ratio*), inverse rank or NDCG. One might also add other measures to evaluate the privacy and anonymity guarantees of P2P message routing algorithms.

- Several parameters and protocol internals of Ethereum's proof-of-stake consensus algorithm is simplified. One can refine the simulation by incorporating a more fine-grained implementation of Proof-of-Stake Ethereum's protocol internals, such as network latencies, and graph topologies.

We invite the Ethereum community to propose, implement, and evaluate the robustness and privacy guarantees of novel privacy-preserving routing algorithms using **ethp2psim**.

Acknowledgements The development of this simulator and our research was funded by the Ethereum Foundation's Academic Grant Rounds 2022.

References

[1] Azzah Al-Maskari, Mark Sanderson, and Paul Clough. The relationship between ir effectiveness measures and user satisfaction. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 773–774. ACM, 2007.

[2] Ferenc Béres, István A Seres, András A Benczúr, and Mikerah Quintyne-Collins. Blockchain is watching you: Profiling and deanonymizing ethereum users. In *2021 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, pages 69–78. IEEE, 2021.

[3] Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. Deanonymisation of clients in bitcoin p2p network. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 15–29, 2014.

[4] Shaileshh Bojja Venkatakrishnan, Giulia Fanti, and Pramod Viswanath. Dandelion: Redesigning the bitcoin network for anonymity. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(1):1–34, 2017.

[5] Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards privacy in a smart contract world. In *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers*, pages 423–443. Springer, 2020.

[6] Mikel Cortes-Goicoechea and Leonardo Bautista-Gomez. Discovering the ethereum2 p2p network. In *2021 Third International Conference on Blockchain Computing and Applications (BCCA)*, pages 81–88. IEEE, 2021.

[7] Giulia Fanti, Shaileshh Bojja Venkatakrishnan, Surya Bakshi, Bradley Denby, Shruti Bhargava, Andrew Miller, and Pramod Viswanath. Dandelion++ lightweight cryptocurrency networking with formal anonymity guarantees. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2(2):1–35, 2018.

[8] Giulia Fanti and Pramod Viswanath. Anonymity properties of the bitcoin p2p network. *arXiv preprint arXiv:1703.08761*, 2017.

[9] Adem Efe Gencer, Soumya Basu, Ittay Eyal, Robbert van Renesse, and Emin Gün Sirer. Decentralization in bitcoin and ethereum networks. In *International Conference on Financial Cryptography and Data Security*, pages 439–457. Springer, 2018.

[10] kaiserd. A tor-based validator anonymity approach (incl. comparison to dandelion), Nov 2022.

[11] Domokos M Kelen, Ferenc Béres, István A Seres, and András A Benczúr. Integrated onion routing for peer-to-peer validator privacy in the ethereum network. Available at https://info.ilab.sztaki.hu/ kdomokos/OnionRoutingP2PEthereumPrivacy.pdf.

[12] Philip Koshy, Diana Koshy, and Patrick McDaniel. An analysis of anonymity in bitcoin using p2p network traffic. In *Financial Cryptography and Data Security: 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers 18*, pages 469–485. Springer, 2014.

[13] Sarah Meiklejohn and Rebekah Mercer. Möbius: Trustless tumbling for transaction privacy. *Proceedings on Privacy Enhancing Technologies*, 2018(2):105–121, 2018.

[14] David Mödinger, Henning Kopp, Frank Kargl, and Franz J Hauck. Towards enhanced network privacy for blockchains. In *Proc. of the 38th IEEE Int. Conf. on Distributed Computing Systems (ICDCS)*, 2018.

[15] Alexey Pertsev, Roman Semenov, and Roman Storm. Tornado cash privacy solution version 1.4. 2019.

[16] Michael G Reed, Paul F Syverson, and David M Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected areas in Communications*, 16(4):482–494, 1998.

[17] Luca Serena, Mirko Zichichi, Gabriele D'Angelo, and Stefano Ferretti. Simulation of dissemination strategies on temporal networks. In *2021 Annual Modeling and Simulation Conference (ANNSIM)*, pages 1–12. IEEE, 2021.

[18] István András Seres, Dániel A Nagy, Chris Buckland, and Péter Burcsi. Mixeth: efficient, trustless coin mixing service for ethereum. *Cryptology ePrint Archive*, 2019.

[19] Piyush Kumar Sharma, Devashish Gosain, and Claudia Diaz. On the anonymity of peer-to-peer network anonymity schemes used by cryptocurrencies. *arXiv preprint arXiv:2201.11860*, 2022.

[20] Peter Todd. Stealth addresses. *Post on Bitcoin development mailing list*, 2014.

[21] Zhipeng Wang, Stefanos Chaliasos, Kaihua Qin, Liyi Zhou, Lifeng Gao, Pascal Berrang, Ben Livshits, and Arthur Gervais. On how zero-knowledge proof blockchain mixers improve, and worsen user privacy. *arXiv preprint arXiv:2201.09035*, 2022.

[22] Zachary J Williamson. The aztec protocol. *URL: https://github.com/AztecProtocol/AZTEC*, 2018.