Integrated Onion Routing for Peer-to-Peer Validator Privacy in the Ethereum Network

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

March 3, 2023

1 Introduction

In this document, we summarize our findings regarding the applicability of onion routing for validator privacy in the Ethereum network. The document is written in the context of [1], where we summarize the motivation, background, and related literature related to validator privacy research. The ideas presented in this document are our own, however, some were invented through discussions with members of the Validator Privacy Research discord group (see Acknowledgements in Section 4).

Onion routing is a privacy technique for anonymous communication, also used in the Tor network [2]. It allows an originator and another party to communicate through a series of relayers in a way that no single party can confidently link the identity of the originator to the recipient or the content of the message. Onion routing works by encapsulating the messages in multiple layers of encryption, where each relayer is able to reverse a single layer of encryption.

We investigate whether a scheme based on onion routing can be used to make validator on-chain identities (i.e., public key) unlinkable to validator off-chain identities (i.e., IP addresses) in the Ethereum proof-of-stake protocol (PoS). This work focuses on a solution self-contained in the Ethereum network, unlike [3], which investigates the possibility of using the Tor network for the same purpose.

It is generally considered very hard to implement Tor-like solutions in a way that there is no privacy leakage through various wire-protocol level side-channels such as timing side-channels or other metadata leakages (e.g., the size of the messages and other fingerprints, quasi-identifiers). These details are out of scope for this work, and in what follows, we primarily focus on Ethereum-specific considerations and omit general onion routing-specific details and adversaries.

Ultimately, we conclude that onion routing indeed can be used to enhance validator privacy. However, the design and implementation have to pay attention to several Ethereum-specific details, such as side-channels or spam protection.

2 Onion routing for validator privacy

The basic goal we set out to accomplish is to make it possible for PoS participants to broadcast attestations and propose blocks to the network without revealing their off-chain identity. If a participant broadcasts self-originated messages themselves, then adversaries can link the participant's off-chain and on-chain identities with high probability through various strategies [4]. Dandelion [5] and Dandelion++ [6] propose schemes where the role of the originator is separated from the role of the broadcaster, and the message arrives from the originator to the broadcaster through a series of hops in the peer-to-peer network, where each hop is a relayer who transmits the message to the next one in plain text. However, it turns out that employing relayers is insufficient to prevent similar attacks [7]. Further, a fundamental problem with such approaches is the first hop, who is aware of the content of the message and also communicates with the originator directly, revealing both the public key and IP address at the same time, see Section 3.2.

The general idea of an onion routing scheme is similar to Dandelion-style schemes in that the message is relayed from the originator to the broadcaster through a number of hops. The main difference, however, is that the relayers forward the message encrypted, and only the broadcaster sees the message in plain-text. The originator encrypts the message using the public key of each relayer, and each relayer forwards the message by first unencrypting it using its private key. Successfully executed, onion routing has the potential to prevent any single party from linking the IP address and public key of the originator.

2.1 Proposed protocol

In this section, we summarize a protocol that uses onion routing in the way we described in the previous section.

Relayer selection. The originator needs to select at least 3 nodes from a uniform distribution over the participants of the peer-to-peer network, without revealing information about the selected nodes to any other participant. Ideally, these need to be *independent* of existing peer-to-peer neighborhoods, otherwise, the selection is not uniform. We describe the reason for the lower limit of 3 in Section 2.2, and further discuss the challenges related to random selection in Section 3.1.

Establishing the communication channel. The originator o first needs to establish an encrypted communication channel to the first relayer r_1, including exchanging public encryption keys, shared secrets, etc. When the channel is established, the originator needs to repeat the process with the second relayer r_2, *however*, this is done through the already established encrypted channel, with r_1 relaying messages between o and r_2. This way r_2 has no knowledge of the off-chain identity of o. When an $o \leftrightarrow r_2$ channel is established, the process

is repeated for r_3, r_4, etc., each time using the channel from the previous step. The final channel between o and the broadcaster $r_k = b$ uses all r_i relayers.

Channel maintenance. Since each new channel carries the additional risk of being compromised and establishing a channel takes time, the established channel needs to be maintained as long as possible. Maintenance consists of exchanging keepalive-messages, spam-protection proofs described in Section 3.4, and generating dummy traffic described in Section 3.3.

Originating messages. When the originator needs to broadcast attestations or propose blocks, they use the established channels to do so.

2.2 Privacy analysis

In this section, we study the privacy guarantees of the protocol described in Section 2.1. Let us assume that an adversary controls the set of nodes A with $|A| = n$ of the N participants of the networks. Let us also assume that the communication channel has $k \geq 3$ participants apart from the originator o, denoted by $r_1, \ldots, r_k = b$, where b is the broadcaster. Then for each participant

$$P(r_i \in A) = \frac{n}{N}. \tag{1}$$

If all participants r_i are under the control of the adversary, then the privacy of the originator is obviously compromised on the first attestation or block proposal that goes through the channel. Thus, the best we could hope to show is that *as long as one participant r_i is honest*, the originator keeps their privacy. Unfortunately, this is almost, but not quite true, as we will see in a moment.

To make a more formal analysis, we first need some assumptions.

Assumption 2.1 (Local adversary)**.** The adversary is only able to observe the communication of the nodes they control.

Assumption 2.2 (Relayer uniformity)**.** The relayer selection is uniformly random and the selection process does not reveal any information about the selection to others.

Assumption 2.3 (No global correlation)**.** The first relayer r_1 cannot correlate the forwarded messages to public information in other ways, where public information means information available to nodes outside of o, r_i.

Proposition 2.4. *Given Assumptions 2.1 to 2.3 and $k \geq 3$, the privacy of the originator is protected as long as either r_1 or both r_{k-1} and $r_k = b$ are honest. As a consequence,*

$$P(\textit{compromised}) \leq \alpha(1 - (1 - \alpha)^2) \leq 2\alpha^2, \tag{2}$$

where α is the probability of any single relayer being controlled by the adversary.

Proof. Since the only entity aware of the off-chain identity of the originator is r_1 (Assumption 2.1), if r_1 is honest, the privacy of o is preserved: for any other node, the anonymity set for the originator is the set of all participants (Assumption 2.2). If r_1 is compromised, but both r_{k-1} and r_k are honest, then r_1 and any other adversary nodes in the channel can only rely on public information (i.e., information from after the message is broadcasted) regarding the on-chain identity of o, which is of no use (Assumption 2.3). □

We believe that any stricter theoretical bounds are unreasonable. There are two reasons for this. First, if both the off-chain identity of o and the identity of b are both known to the adversary, then the anonymity set of o is greatly reduced: the adversary can attempt to identify the broadcasters for all messages in the network through strategies similar to those described in [7]. This leads to the requirement that *both* r_{k-1} and $r_k = b$ need to be honest in order to hide the identity of the originator.

Second, if both r_1 and either r_{k-1} or r_k are controlled by the adversary, then additional honest intermediate r_i relayers with $1 < i < k-1$ can not guarantee the anonymity of o unfortunately. The reason for this is that if two adversary nodes are in the same channel, then they can use end-to-end correlation attacks [8, 9] to find out about the fact that they are in the same channel, effectively making any intermediate relayers negligible from a privacy standpoint. Such attacks can work either passively, by correlating packet timings and sizes, or in this case even actively, deterministically introducing small but detectable latency patterns into the channel traffic. Note, however, that if such attacks can be prevented, then the privacy guarantee slightly improves: if we can prevent end-to-end correlation attacks, then either r_1 or *any* two consecutive relayers $r_i, r_{i+1}, 1 < i$ being honest is sufficient to guarantee the anonymity of o.

Proposition 2.4 further relies on Assumptions 2.1 to 2.3 described above. Preparing for adversaries that can observe communication between 3rd parties is out of the scope of this work, so we do not further discuss Assumption 2.1. The remaining two are fairly strong assumptions. Assumption 2.2 is a hard but well-contained problem, which we discuss in Section 3.1. Assumption 2.3 is more insidious, as there are many possible ways of attempting to correlate information from before the honest node in the channel to public information.

Also note that while r_1 is aware of the off-chain identity of o, r_1 is not provided with the information that they are the first relayer, so even if all participants of a channel are controlled by the adversary, they can not be certain of this fact. However, from the perspective of any r_i, the previous relayer in the sequence always has a higher than uniform chance of being o, which leaks information. Further, a relayer in the channel can employ timing-based attacks to try to reveal its index i, as described in Section 3.2.

3 Challenges and considerations

In this section, we discuss the challenges relating to Assumptions 2.2 and 2.3, as well as other requirements such as spam-resilience, latency, and robustness.

3.1 Relayer selection

Selecting participants for the communication channel such that it includes at least one honest node is a nontrivial task. Further, the design space of relayer selection approaches is vast: we describe methods we are aware of, and list arguments for and against them. The most feasible method with our current understanding seems to be a combination of rate limiting nullifier (RLN) [10] proofs and the usage of Ethereum's DevP2P protocol [11], described in Sections 3.1.3 and 3.1.5

3.1.1 Sampling existing neighbors

The originator can randomly sample k nodes from its neighbors from the existing peer-to-peer network. This approach is beneficial mainly because it is very easy to implement. Disadvantages include the fact that each relayer knows that the originator is one of their neighbors, potentially opening the possibility of intersection-attacks [12]; and also the fact that eclipse-attacks [13] can amplify the probability of de-anonymizing a targeted originator.

3.1.2 Asking the neighbors for their neighbors

The originator could choose r_1 from its existing neighbors, ask r_1 to choose r_2, etc. This approach avoids intersection attacks, however, we can assume that the first adversary node encountered only offers further adversary nodes as potential relayers. As a result, the probability of being compromised is *at least* α, with eclipse-attacks again amplifying the probability of being de-anonymized.

3.1.3 Using the existing DevP2P protocol

One could repeatedly use the already established node-discovery protocol of Ethereum [11] to sample nodes from the network. However, while the protocol does have some sybil-resistant properties, it is unclear whether DevP2P is sufficiently sybil-resistant to be employed for relay selection.

3.1.4 Distributed relayer-sharing protocols

It is possible to design a protocol for sharing possible relayer nodes between clients, similar to how new transactions spread around the network using the mempool. A key difference, however, is that messages containing node information can be treated as ephemeral, meaning that a node discards the information after relaying it, only keeping a secret pool of randomly selected nodes for future

use as relayers. However, any such scheme needs to be carefully studied to assess multiple possible failure points, including the potential impact on network load, spam resistance, and resistance against biased adversary nodes who also participate in the protocol.

3.1.5 Proof of being a validator in zero-knowledge

A general approach to providing Sybil resistance is to only select relayers from nodes that have proven that they are themselves unique validators in PoS. This can be done using rate limiting nullifiers (RLN), a zero-knowledge proof system [14, 10] that can be used by the validators to prove that they are indeed validators, along with the fact that the proof is unique to them, without revealing anything further. Then, the originator can collect a certain m number of possible relayers and corresponding RLN proofs, and randomly sample k of them as relayers. Assuming at most n validators controlled by the adversary, this means a guaranteed at most $\frac{n}{m}$ chance of a single chosen relayer being controlled by the adversary. The sampling of m possible relayers can be done in the fashion described in Sections 3.1.3 or 3.1.4.

3.2 First-hop detection

In this section, we describe an attack intended to find the relayer index i of an r_i, or more specifically whether a given relayer is the first in a channel. The same attack can be used against Dandelion-style solutions, in which case it serves as an immediate de-anonymization vector for r_1 against the originator.

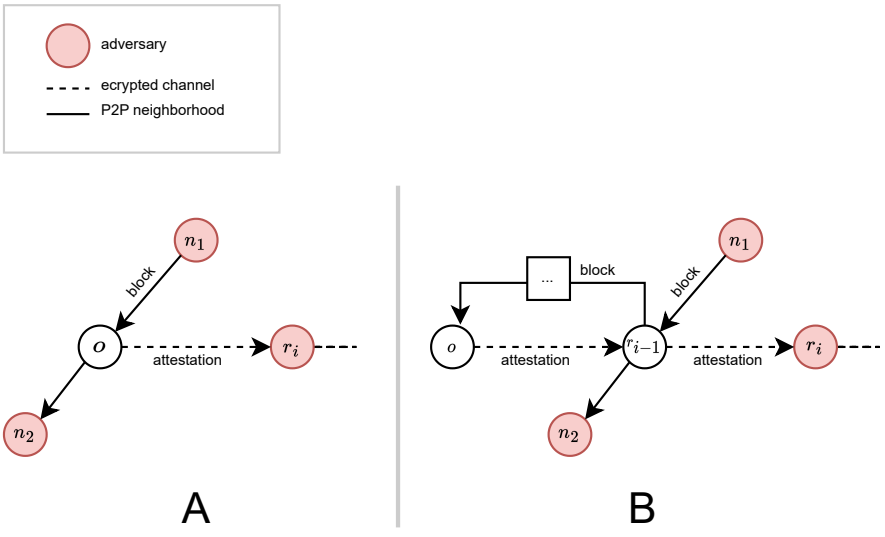

Figure 1: Possible scenarios for first-hop detection.

The main assumption behind the attack is the relayers can detect the timing

of the relayed attestation, even in its encrypted form, while in Dandelion-style solutions it requires no additional assumptions. The goal of the relayer r_i is to determine whether the previous relayer in the sequence r_{i-1} is the originator. We describe a way to attempt this by manipulating the timing of the attestation.

The earliest time an originator can make an attestation is when they become aware of a new head block [15], at which point they are incentivized to broadcast the attestation as fast as possible. The adversary can use this to their advantage by placing a neighbor n_1 in connection with r_{i-1}, acquiring new blocks as fast as possible (for example through a supernode or many nodes placed geographically evenly throughout the network), and propagating the blocks to r_{i-1} through n_1.

The two possible scenarios that need to be differentiated between are displayed in Figure 1. In scenario A, $o = r_{i-1}$, while in scenario B $o \neq r_{i-1}$ and r_{i-1} is connected to o through some other path in the P2P graph. As we can expect the propagation path $n_1 \to r_{i-1} \to [\ldots] \to o \to r_{i-1} \to r_i$ to take longer than $n_1 \to r_{i-1} \to r_i$, we can expect this method to be able to differentiate between A and B given realistic estimates for propagation and hop latencies. An additional n_2 neighbor placed in connection with r_{i-1} can serve as a benchmark for such latencies.

Protecting against this kind of attack by introducing random or minimum delays is not an effective strategy, as random delays can be averaged out over longer time periods, and minimum delays only work if part of the network is willing to not use them, otherwise, the attacker can just subtract the general minimum delay from the measured timings. That is to say, minimum delays can not protect everyone at the same time.

3.3 De-anonymization through public information

In this section, we discuss a strategy that adversaries could use to correlate events of the encrypted channel to global events, i.e., Assumption 2.3.

The largest concern regarding this is if r_1 is able to detect when they are relaying encrypted blocks or attestations, as these become public shortly thereafter. Each validator is required to send an attestation in each epoch, however, the slot when they are required to do this is selected randomly. Since the timing of slots is relatively tight [16], this makes it possible for r_1 to intersect the attestor-sets of each slot that o is active in. Further, it is safe to assume that attestations and blocks can be distinguished in encrypted form, as the latter is much larger in size. This makes such attacks even more problematic, as only a single validator publishes a block proposal in a particular slot.

A solution for these attacks is to generate *dummy traffic* in the channel, hiding the timing of real attestations and proposed blocks. Each originator sending a dummy attestation in each slot is plausible, as these are trivial in size [15] and can be discarded by the broadcaster. Blocks, on the other hand, can be over 100kb in size. A possible scheme to reduce the load generated by dummy blocks

is to partition validators into multiple classes, and have every validator in the same class send dummy blocks whenever the block-proposer for the given slot is from the same class as themselves. This results in a trade-off: larger classes mean larger anonymity sets for block proposers but also mean more overhead traffic in the network.

We note that the planned Single Secret Leader Election (SSLE) [17] scheme could, unfortunately, make the partitioning strategy for dummy block traffic impossible to implement, as the participants can no longer know whether they are in the same class as the secret leader. It is unclear whether an SSLE algorithm itself can include such a partitioning in a way that participants *can* compute the class of the leader without revealing the identity of the leader.

3.4 Spam protection

Spam protection is a crucial point of onion routing. When originators communicate in plaintext, then each relayer of a message can assert its validity. This prevents anonymously flooding the network with messages, as a spammer cannot freely generate an unlimited number of valid messages.

On the other hand, relayers in encrypted channels cannot assess the validity of messages. We propose the usage of an RLN scheme [14], similar to Section 3.1.5. However, instead of proving the validity of messages, we propose proving that the given relayer r_i is one of s allowed relayers for the originator o.

Proving the validity of messages using ZK technology is infeasible because attestation and block proposal have very tight latency requirements [16], and generating ZK-proofs can take a considerable amount of time (where even half a second is *considerable* for this use-case). However, if we use RLN constructs to limit the number of channels that any single originator is allowed to use at the same time, then the proofs can be computed before-the-fact.

Our goal is then to achieve for each originator to be able to uniquely prove that

- They are part of the validator set,
- The relayer r is index $1 \leq t \leq s$ of their allowed max s relayer capacity,
- The proof is for the given epoch with index e.

This is indeed possible to achieve by using RLN. Below we give an example scheme that achieves all three points. Using the notations and terminology of [10], the originator needs to prove

$$f(M, N, pk_r, y, e, t), \text{ with} \tag{3}$$

$$y = \text{hash}(k_o, e, t) \cdot \text{hash}(\text{pk}_r, e) + k_o, \text{ and} \tag{4}$$

$$N = hash(k_o, e, t, 0), \tag{5}$$

where M is the Merkle root containing all validators, pk_r is the public key of the relayer, e is the epoch number, t is the selected index of the allowed relayer

capacity of o, and k_o is the private key of o. The relayer keeps the resulting nullifier N and line evaluation y.

This scheme works by having the nullifier N be deterministic and unique for each (k_o, e, t) tuple, while offering no information about the identity of o. The originator o cannot prove two separate relayers for the same epoch e with the same index t without the nullifier being identical for both.

Given a public store for nullifiers, this solves the problem. However, such a store is not a given, and creating one is not easy, as it would recursively run into its own spam problem. For lack of a better solution, the described scheme also relies on the "two points make a line" technique of [10]: if the originator proves two separate pk_r relayer keys for the same (e, t) values, then the proofs together reveal the private key k_o of the originator through the corresponding line evaluations, y_1 and y_2. Since the originator is a staker in PoS, this, together with nullifiers to match proofs to each other, could serve as a sufficient deterrent from spamming the network.

3.5 Latency and robustness

Latency and robustness are crucial properties of any method that validators use to publish attestations or propose blocks. Relayers could go offline at any minute, and network-stability issues could cause attestations to be delayed. To improve the chances of successfully participating, the originator can use multiple channels, improving both expected latency and robustness. The dummy traffic proposed in Section 3.3 can also serve as a continuous benchmark for the properties of the channels, with sub-par channels being dropped in favor of better ones. This is also compatible with the spam-prevention scheme described in Section 3.4, where for example, $s = 12$ allows for either 4 parallel channels with 3 relayers each or 3 parallel channels with 4 relayers each.

The chance of being compromised is slightly worse when multiple channels are used. If a single channel results in o being compromised with probability β, then using t channels yields $P(\text{compromised}) = 1 - (1 - \beta)^t \leq t\beta$.

4 Conclusion

We conclude that while onion routing can work for validator privacy, it has a large number of possible failure modes, each of which needs to be addressed carefully. However, if all required assumptions can be met, then it has theoretical guarantees for keeping validators private.

Acknowledgements This research was funded by the Ethereum Foundation's Academic Grant Rounds 2022. We would like to thank the members of the Validator Privacy Research discord group, especially and in no particular order AtHeartEngineer, George Kadianakis, and Ksr (kaiserd), for discussion and feedback on many of the ideas described in this document.

References

[1] Ferenc Béres, István A Seres, Domokos M Kelen, and András A Benczúr. *ethp2psim: Evaluating and deploying privacy-enhanced peer-to-peer routing protocols for the Ethereum network.*

[2] Roger Dingledine, Nick Mathewson, and Paul Syverson. *Tor: The second-generation onion router*. Tech. rep. Naval Research Lab Washington DC, 2004.

[3] kaiserd. *A Tor-based Validator Anonymity Approach (incl. comparison to Dandelion)*. Nov. 2022. URL: https://ethresear.ch/t/a-tor-based-validator-anonymity-approach-incl-comparison-to-dandelion/14134.

[4] Sebastian Brügel. *Proof-of-Stake Validator Sniping Research*. Apr. 2022. URL: https://medium.com/hoprnet/proof-of-stake-validator-sniping-research-8670c4a88a1c.

[5] Shaileshh Bojja Venkatakrishnan, Giulia Fanti, and Pramod Viswanath. "Dandelion: Redesigning the bitcoin network for anonymity". In: *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 1.1 (2017), pp. 1–34.

[6] Giulia Fanti, Shaileshh Bojja Venkatakrishnan, Surya Bakshi, Bradley Denby, Shruti Bhargava, Andrew Miller, and Pramod Viswanath. "Dandelion++ lightweight cryptocurrency networking with formal anonymity guarantees". In: *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 2.2 (2018), pp. 1–35.

[7] Piyush Kumar Sharma, Devashish Gosain, and Claudia Diaz. "On the Anonymity of Peer-To-Peer Network Anonymity Schemes Used by Cryptocurrencies". In: *arXiv preprint arXiv:2201.11860* (2022).

[8] arma. *One cell is enough to break Tor's anonymity*. Feb. 2009. URL: https://blog.torproject.org/one-cell-enough-break-tors-anonymity/.

[9] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul Syverson. "Users get routed: Traffic correlation on Tor by realistic adversaries". In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 2013, pp. 337–348.

[10] Vitalik Buterin. *Some ways to use ZK-SNARKs for privacy*. June 2022. URL: https://vitalik.ca/general/2022/06/15/using_snarks.html.

[11] Ethereum. *Node Discovery Protocol v5*. Oct. 2020. URL: https://github.com/ethereum/devp2p/blob/master/discv5/discv5.md.

[12] Oliver Berthold and Heinrich Langos. "Dummy traffic against long term intersection attacks". In: *Lecture notes in computer science* (2003), pp. 110–128.

[13] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. "Eclipse attacks on bitcoin's peer-to-peer network". In: *24th {USENIX} Security Symposium ({USENIX} Security 15)*. 2015, pp. 129–144.

[14] barryWhiteHat. *Semaphore RLN, rate limiting nullifier for spam prevention in anonymous p2p setting*. Feb. 2019. URL: https://ethresear.ch/t/semaphore-rln-rate-limiting-nullifier-for-spam-prevention-in-anonymous-p2p-setting/5009.

[15] Ben Edgington. Jan. 2023. URL: https://eth2book.info/.

[16] blagoj. *Ethereum consensus layer validator anonymity using Dandelion++ and RLN conclusion*. Apr. 2022. URL: https://ethresear.ch/t/ethereum-consensus-layer-validator-anonymity-using-dandelion-and-rln-conclusion/12698.

[17] Vitalik Buterin. *Simplified SSLE*. Apr. 2022. URL: https://ethresear.ch/t/simplified-ssle/12315.