# Improving Anonymity of the Lightning Network using Multiple Path Segment Routing

**Joran Heemskerk**[1] , **Stefanie Roos**[1] , **Satwik Prabhu Kumble**[1]
[1]TU Delft

## Abstract

The Lightning Network (LN) is a second-layer solution built on top of the Bitcoin protocol, allowing faster and cheaper transactions without compromising on decentralization. LN is also designed to be more anonymous, since less information has to be shared with the entire network. This should, in theory, improve privacy as well. However, recent works have shown that this is not quite true: due to the deterministic nature of contemporary routing protocols, an adversarial node on a payment path is able to uniquely identify at least one sender or recipient for about 70% of observed transactions.
To combat this breach of anonymity, we propose a new routing algorithm that makes use of multiple path segments. By splitting the routing problem into multiple routing sub-problems and forming the final route by joining these sub-routes together, we introduce a degree of randomness which nullifies this kind of adversarial attack. Even when designing a counterattack, we still get a substantial improvement in anonymity, roughly tripling the number of source/destination pairs per attack. However, the protocol is also very costly, doubling the average fee and increasing the average hop count by more than 60%. This shows that the proposed protocol is not strictly superior to the current implementation, meaning that other (less drastic) protocol proposals are likely to give better cost/anonymity trade-offs.

## 1 Introduction

Proof-of-work blockchains, such as Bitcoin, are becoming increasingly popular as a novel way to exchange value over the internet. By creating and utilizing a set of cleverly chosen rules and incentives, participants in the Bitcoin network are able to independently form a consensus on the state of the network, without relying on one central source of truth [11].
In Bitcoin, the 'state of the network' is essentially a list of all transactions that have ever happened on Bitcoin. These transactions are stored on the Blockchain. Every participant on the network independently verifies these transactions, in order to make sure no one has cheated [3].

Although this method of validating transactions gives participants the ability to form a consensus without requiring a trusted third party, it does come with a number of downsides. The most notable of which is the 'scalibility problem': the ability to be able to handle more transactions as the usage of Bitcoin goes up.
To remedy this issue, numerous 'second-layer' solutions have been proposed [6], the most promising of which being the 'Lightning Network (LN)' [14]. On LN, users can send payments to each other, which are in effect 'claims' of collateral that are present on the Bitcoin blockchain. By doing this, it becomes possible to take a large number of transactions 'off-chain', thereby providing a solution to the 'scalability problem'.

However, LN comes with its own set of problems, particularly in the category of privacy.
Transactions on LN are onion-routed. In onion routing, the source decides on the routing path beforehand, and the message is encrypted at every hop along the transaction path. When an intermediary node receives a (encrypted) transaction, it only learns about the transaction itself and to what node it should forward the transaction. As such, only a small part of the complete payment path is revealed to the intermediary node: the previous node (from whom the node got the payment) and the next node (to whom it should forward the payment). The complete payment path, including real source and destination of the payment, is never revealed to the intermediaries. In theory, this means that the payment is completely anonymous.
But in practice, this does not provide nearly enough of a guarantee to anonymity. A recent work has shown that if an adversary is right after the sender on a payment path, or right before the recipient, then the sender or recipient is at risk of deanonymization [17].
Moreover, another recent study shows an even more concerning finding: the Lightning Network's routing algorithm is vulnerable to spying adversaries. With the use of public information, an adversarial node on a payment path is able to uniquely identify either the sender or the receiver of a payment in about 70% of cases, regardless of its position on the payment path [8].

The main attack vector that such an adversary uses is the *hash time-locked constract (HTLC)* protocol [4]. This protocol is required in order to send payments between nodes that are not directly connected. However, by analysing the timelock, adversarial nodes are able to estimate their position in relation to the recipient, which can reveal who the recipient node is.
Once the identity of the recipient node is revealed, it then becomes possible for the adversary to predict who the sender is, by analysing which nodes on the network would route their transactions through the adversary. Since LN's current routing protocols are largely deterministic, this type of analysis is feasible.

In order to combat this, investigation into a new routing protocol is in order.
This paper proposes a design for a routing protocol using multiple path segments. In this protocol, the complete route is formed by constructing two separate routes, which overlap at a central node, called the 'Dovetail' node. This routing protocol is loosely based on the 'Dovetail protocol' [16], which is a source-based Next Generation Internet (NGI) protocol, designed as a more privacy-minded alternative to the Internet Protocol (IP). However, since the requirements for Lightning network routing differ significantly from internet routing, the similarities remain in name only.
We also show the effects that that the proposed protocol has on anonymity and routing efficiency by simulating a network and sending transactions. We show that, for a simulation with simple graph types, the average number of pairs per attack more than triples when using our proposed routing algorithm. However, it comes at a cost for performance, doubling the average fee and increasing the average hop count by more than $60\%$.

The paper will be structured as follows: in Section 2, we give an in-depth look at how the Lightning Network operates, and define the types of adversarial attacks that the Lightning network is vulnerable to. We then describe our proposed routing algorithm in Section 3, designed to mitigate these weaknesses. This section explains the design considerations of our proposed protocol, but also discusses the design for a counterattack against our protocol. In Section 4, we describe our simulation framework and evaluate the performance of our routing protocol with LN's current routing protocols and the counterattack. In Section 5, we relate the acquired results back to the main research question. Finally, we conclude the paper with a discussion and proposal of further research in Section 6.

# 2 Attacking the Lightning Network

In this section, we will look at the lightning's routing process in more detail. We also analyse how the current routing protocols are vulnerable to adversarial attacks, which we aim to mitigate in our design.
However, before doing so, we will first look at HTLC's, a key concept for conducting LN transactions.

## 2.1 HTLC's and timelocks

Understanding how timelocks work in LN is the key to understanding its privacy vulnerabilities. 'Trustlessness', within the context of LN, refers to two parties not having to trust that the other party does not cheat, as cheating is disincentivized by the protocol. Parties only have to verify that the protocol rules are followed correctly. On LN, if two parties share a payment channel, it is possible to send a transaction in a trust-less manner over this channel. Sending a transaction to a recipient that the sender is not directly connected to, however, requires a more sophisticated protocol to guarantee trustlessness: *Hash Time Locked Contracts (HTLC's)* [1].
HTLC's work as follows: the recipient of a transaction chooses a secret number $s$. It then generates $h = \mathbf{SHA}256(s)$, and gives this to the sender. In order for the sender to now make an indirect payment to the recipient, the sender constructs a route via one or more intermediaries. For each intermediary, the sender then makes the following arrangement with its neighbouring intermediary: if you can give me the secret number $s$ that results in $h$ *within a certain time limit*, then I will give you the payment. This arrangement is known as the HTLC.
The intermediary then makes the same arrangement with the next node, which is either another intermediary node or the final recipient.
Since the final recipient knows $s$, it can claim the payment by revealing $s$ to the preceding node. Since this node now knows $s$ as well, it can claim the payment from its own preceding node, which can be another intermediary node or the sender. As such, the payment essentially ripples backwards, from the recipient to the sender.
Of course, the intermediary nodes don't want to risk losing their money, so they want to make sure that they can claim the payment from the previous node once they receive $s$ from the next node. To guarantee this, they make sure that the value of the preceding HTLC has a larger time limit than the ensuing HTLC, the difference being a sort of 'padding'. This padding is referred to as the *timelock*. The HTLC between two nodes on a payment path is therefore the sum of all timelocks that get added from the ensuing HTLC's.
A final note about timelocks: all the timelocks used on the network are public information. This is so that the sender can construct the total timelock required to send the payment beforehand.

## 2.2 Lightning routing protocols

There are currently three dominant routing protocols on the Lightning Network: LND [2], Eclair [3] and C-Lightning [4]. Each of these routing protocols use Dijkstra's algorithm [5], together with a certain implementation-specific cost function. These cost functions are based on a number of factors, such as fees, the locktime, or previous experiences with this channel. For

[1] https://en.bitcoin.it/wiki/Hash_Time_Locked_Contracts
[2] https://github.com/LightningNetwork/lnd
[3] https://github.com/ACINQ/eclair
[4] https://github.com/ElementsProject/lightning
[5] This is not entirely correct, as Eclair uses a modified version of Dijkstra known as *Yen's k shortest path algorithm* [18].

example, for LND (the most popular routing algorithm), the cost function is as follows:

$$cost = amount \cdot timelock \cdot riskfactor + fee + bias$$

Here, $amount$ is the amount to be sent in the transaction, $timelock$ indicates how much extra delay this channel adds to the total timelock, $riskfactor$ is a constant set to $15 * 10^{-9}$, $fee$ is the total fee for using this channel and $bias$ is a value based on previous experiences with this channel.
The cost functions for C-lightning and Eclair work similarly, although the parameters are different.

When sending a transaction on the Lightning Network, the sender ensures that every channel along the transaction path has a high enough capacity to handle the transaction. Additionally, since the sender knows the local balance distribution between itself and its neighbours, it makes sure that this balance is sufficient. Since the sender does not know of the state of channel balances on all channels that are not directly linked to itself, the payment may fail when one of these balances is insufficient. Since all previous nodes before a failing channel have already committed some collateral, they either have to wait until the total timelock expires or negotiate on revoking the HTLC[6], leading to a lot of overhead before they can use the locked collateral again for future payments.

### 2.3 Adversarial attacks on LN

Based on our knowledge on HTLC's and the existing routing protocols, we are able to conceive of an anonymity-compromising attack. This attack is identical to the attack described in [8]. For a more in-depth description of the attack, we refer to this paper.

#### Assumptions

We assume that the Lightning Network has a number of nodes that can be considered 'passive attackers'. When these nodes witness a transaction, they will try to determine the source and the destination of the transaction.
The only information that the adversarial nodes are aware of is public information: fees, timelocks, age of channels and nodes, and which node uses which routing protocol. Of course, they also know of the data that is revealed in the transaction itself.
The adversaries do not remember past events. All their choices are based on current information, and they do not keep track of past transactions in their future analysis. They are also bounded by polynomial computation time, meaning that they are unable to brute-force the decryption of encrypted messages.
Lastly, the adversaries work alone. In the past they have been able to collaborate by sharing transaction data (if they were on the same transaction path), but an upcoming protocol change makes it infeasible to trivially link transactions in this way [10].

[6]https://lists.linuxfoundation.org/pipermail/lightning-dev/2019-April/001986.html

#### Phase I

In Phase I, the adversary tries to determine the destination of a witnessed transaction. The transaction witnessed has a certain timelock. The value of this timelock is the total timelock, which is the sum of all the timelocks that come after passing through the adversary. Since the timelocks of all the nodes on the network are public, the adversarial node can perform a search, starting from the next node. From this node, all possible paths are followed. A path can be followed if the capacity is below the transaction amount and if the total summed timelocks does not exceed the timelock witnessed in the transaction. The search continues until either no more nodes are available, or the transaction path exceeds four hops (since path length complexity increases exponentially).
A node can be a destination candidate if the summed up timelocks from the adversary to this node equals the witnessed total timelock. However, it is possible for the transaction to add extra padding to the total timelock. This is called 'shadow routing'[1]. If shadow routing is applied, the total timelock required to reach a node may be lower than the sum of the timelocks along the path. This implies that, when shadow routing is applied, we drop the equality constraint and add every node along the path as a destination candidate.

#### Phase II

In phase II, we take all the possible recipients from Phase I and search for all the possible senders. This is done by going through all the nodes on the path backwards, starting at the preceding node. Using the constructed route $P$ from Phase I and the known cost functions for every node, we try to determine if a given preceding node might construct a route similar to optimal route that we constructed ourselves in Phase I. If the route is optimal, we consider this node to be a possible sender, or a intermediary node. If the route we find is not optimal, the node cannot be an intermediary node. However, it can still be a sender node, since the sender has more local knowledge (such as the state of the local channels). This extra knowledge may allow it to deviate from the optimal route, since it knows that the optimal route is infeasible due to the state of local channels.
Again, for a more complete description of this attack, we refer to Section 4 of [8].

## 3 Routing algorithm

Now that we know the attack we have to defend against, it is time to develop the protocol itself.
Our proposal is that of path segment routing. In path segment routing, the initial routing request is split up into parts. Instead of routing towards a destination directly, two separate routes are constructed, which overlap on one node on the network. The overlapping node is referred to as the 'Dovetail' node. The full route is constructed by joining these separate routes together, to form a complete route from the source to the destination. Figure 1 shows this protocol in action.

Building this routing algorithm requires investigation into three questions: how do we route the first path segment, how do we route the second path segment, and how do we decide on the Dovetail node. Below, each of these questions is dis-

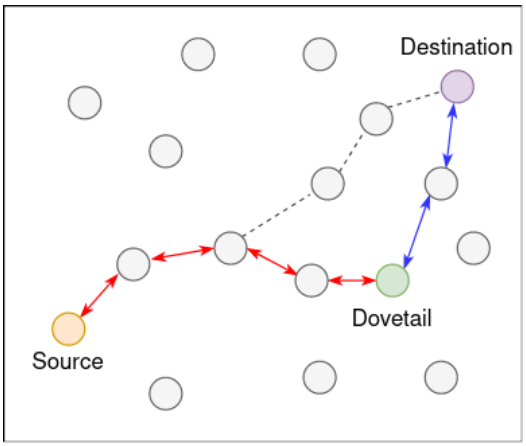



Figure 1: Visualising the segmented path routing protocol. The route from the 'Source' node to the 'Dovetail' node (red) is considered the first path segment, the route from the 'Dovetail' node to the 'Destination' node (blue) is considered the second path segment. The dotted line is the optimal path, as chosen by contemporary routing algorithms.

cussed separately. Afterwards, the final routing protocol is shown.

### 3.1 Routing the first path segment

Routing the first path segment (Figure 1, red path) is a fairly simple question to answer. This problem can be reduced to a straightforward routing problem, where the Dovetail node is now considered to be the destination node.

To answer this routing question, we can simply use one of the existing routing algorithms. For simplicity, we use LND's routing algorithm [9].

In the future, the protocol can be made more sophisticated by changing this choice for routing algorithm. One option is that the protocol may recursively call itself to do the routing. By doing so, we are able to construct a route that, in effect, has multiple Dovetail nodes.
Another option is to implement one of the anonymity-improving routing algorithms built by other team members. These are: adding random hops [7], hop change with partial route computation [5], sub-optimal routes [13], and finally, length-bound random walk insertion [12].

### 3.2 Routing the second path segment

Routing the second path segment (Figure 1, blue path) is a much harder question to answer compared to the first path segment. This problem cannot be reduced to a simple routing algorithm, since the route does not originate from the source node. Because of this, there might be a benefit to delegating the routing computation to other nodes. Doing so will mean the protocol moves away from being a fully source-routed protocol, to one that is only partially source-routed.
A big advantage of delegating routing to another node is that this node can make use of extra local information, such as the state of the channel balances. The node might also make use of 'private channels': channels that are never published via the LN gossip protocol [1].
A disadvantage of delegating routing to another node is that this opens up the protocol for many many new attack angles. For example, if the routing is delegated to an adversarial node and the adversary can see who is requesting the routing, then this gives the adversary a lot of extra information that may be used to compromise anonymity. Routing delegation is therefore not always better.
There are two candidates for routing delegation: the destination node and the Dovetail node.

**Routing done by destination node**
The first candidate for routing delegation is the destination node. Apart from being able to use local information, the advantage of using the destination node in particular is that this does not reveal any extra information to anyone. The destination node is already aware of all transaction data: the source node (which it is communicating with), the destination (itself), and the transaction amount.
However, the fatal flaw with this idea is that the destination node does not care about the efficiency of the payment. The transaction cost is paid by the sender, therefore there is no incentive to make the path any more optimal than is minimally required for the payment to succeed. This in turn means that the destination node will never make use of local knowledge such as private channels, since it can only harm them.

**Routing done by Dovetail node**
The second candidate is the Dovetail node. Choosing the Dovetail node for route delegation gives the same benefits as picking the destination node with regards to local information. Unlike the destination node however, the Dovetail node does have an incentive to route optimally. This is because the source node might have multiple candidates, that are competing for routing the payment.
However, even this might not be enough incentive for the Dovetail node to route optimally. This is because doing so opens the Dovetail node up to a new type of active attacker. This attacker might probe random nodes on the network and ask them to be a Dovetail in routing a transaction. The attacker then compares the timelock given by the Dovetail with the public information the attacker has about the network. If the timelock is lower than expected, then the Dovetail is likely using local information such as private channels.

**Final decision**
Despite the fact that moving away from source routing can give some interesting benefits, these benefits are by no means decisive. Therefore, in our current proposal we opt to keep the protocol simple by sticking to full-source routing. Giving more in-depth analysis of the benefits and costs of moving away from full-source routing is outside the scope of this research paper. We consider this a good subject for future research. In particular, analyzing the effects delegating route computation to the Dovetail node is a topic that invites further investigation.

### 3.3 Choosing the Dovetail node

Choosing what node should be the Dovetail node can have radically different results for efficiency and anonymity. Below is a list of points that might have an effect on the

anonymity and efficiency performance of picking the Dovetail node.

- When the chosen node is on the optimal path, the path segment route is identical to the optimal route and does not provide any anonymity benefits.
- The first (Figure 1 red) path segment results in different timelock values for intermediary nodes, whereas the second (Figure 1 blue) path segment does not have different timelock values for the intermediary nodes.
- The connectivity of the chosen node may influence the performance.

In our implementation, the only requirement we enforce is that the candidate node should not be along the optimal path (including the source and destination node), to make sure that our routing algorithm gives a different path. We do not attempt to make further guesses about which nodes are better candidates for being the Dovetail node, opting instead for simplicity. This means our Dovetail node is picked at random, from the set of all nodes that are not on the optimal path. By doing so, we limit the analysis an adversarial node can do when determining the Dovetail node. In the Results section, we investigate the effects on anonymity for different Dovetail node choices.
One slight optimization we do in routing is to not pick one candidate node, but multiple. We then check which of these candidate nodes gives the shortest route. Doing this drastically reduces that chance of picking a very bad candidate node. It slightly violates the principle that the Dovetail node is picked at random, since nodes closer to the source or destination are more likely to be picked as a Dovetail. However, we consider this slight bias to be negligible and do not account for it in our counterattack. Further research may be needed to investigate the effects of this optimization technique.

## 3.4 Full routing algorithm

Here is the pseudo-code for the complete path segment routing algorithm:

**Algorithm 1** Path segment routing

```
1:  function DOVETAIL(src, dove, dest, amount)
2:      delay ← 0                          ▷ Final timelock value is zero
3:      path2, amt, delay ← Dijk(dove, dest, amt, delay)
4:      if len(path2) = 0 then             ▷ No route found
5:          return [ ], −1, −1
6:      end if
7:      path1, amt, delay ← Dijk(src, dove, amt, delay)
8:      if len(path1) = 0 then             ▷ No route found
9:          return [ ], −1, −1
10:     end if
11:     fullpath ← p1 + p2[1:]             ▷ Append paths
12:     return fullpath, amt, delay
13: end function
```

In this function, the second path segment (line 3) is calculated before the first path segment (line 7). This is because lightning routing needs to be done backwards to deal with network fees.
The *Dijk* function used to calculate these routes is the Dijkstra algorithm used in current Lightning implementations. When routing the first path segment (line 7), the final delay (time lock value) and amount should be the delay that we ended up with in the second path segment (line 3).
In our implementation, we call the above pseudo-code with 5 different candidate Dovetail nodes, and pick the shortest route.

## 3.5 Adversarial counterattack

Since we have modified our routing protocol to be less deterministic, the adversarial attack model described in Section 2.3 should no longer be effective. We will therefore have to devise a new type of attack, which can deal with the introduced randomness.
Since the adversary can no longer assume optimality for the complete route, the sender and recipient pairs are very difficult to find directly. However, the adversary might still be able to determine the two end-points of a transaction, based on its position on the payment path. Although these endpoints might not explicitly reveal the sender and recipient of the transaction, they are still a good first step towards full deanonymization.
There are three possibilities: the attacker is on the first path segment, the attacker is on the second path segment, and the attacker is the Dovetail node itself.

**Attacker on first path segment**

If the attacker is on the first path segment, the attacker will try to determine the sender and the Dovetail node of the transaction. Since the Dovetail node is not the final recipient of the transaction, any node that has a delay higher than zero can be a Dovetail candidate node. Therefore, finding the sender-dovetail pairs requires dropping the constraint that the final timelock should be zero, identical to that of 'shadow routing'. It is expected that the anonymity sets are greatly improved if the attacker is on the first path segment, similar to the improvement found when applying shadow routing.

**Attacker on second path segment**

If the attacker is on the second path segment, the attacker will try to determine the Dovetail and recipient pair. For this, the attack described in Section 2.3 does not need to be modified. As such, there is no expected anonymity improvement in this case.

**Attacker is the Dovetail node**

The last possibility is that the sender is the Dovetail node itself. In this case, it will try to find the sender and recipient pairs. Phase I of the attack is not modified, since finding the recipient node should work identically. Phase II of the attack is modified, where the attacker will look for all sender nodes that pick the attacker as the recipient. Much like when the attacker is on the first path segment, this is identical to assuming shadow routing is applied. The recipient node anonymity set is not expected to improve much. However, because the constraints for routing through the attacker node are (by definition) not optimal, the number of possible sender nodes should improve quite drastically per recipient.

**Full counterattack design**

In our modified counterattack, the adversary first tries to guess on which part of the route the adversary is located. Since the Dovetail node should be roughly in the middle of the payment path, the adversary considers itself to be on the first path segment if the timelock it received is significantly higher than the average expected timelock. If it is significantly lower, it expects to be on the second path segment, and if it is roughly average, it guesses that it is the Dovetail node itself.

For clear results, we assume that the attacker is always able to correctly guess on which part of the payment path it is located. After it has guessed the location, it will determine the source and destination anonymity set of the payment.

Note that the source may not necessarily be the original sender and can also be the Dovetail node. The same applies for the destination and the recipient. We assume that finding the correct Dovetail node is considered to be a privacy compromise, since the real sender or recipient may be found by further investigating this node. However, we do not investigate how this can be done. Instead, we assume that doing so takes extra computation time for every candidate Dovetail node. As such, finding e.g. the sender/dovetail pair is a lower-bound for finding the real sender/recipient pair.

Given a Dovetail node and a path 1 or path 2 attack, finding the correct sender/recipient is an interesting topic for further research.

# 4 Evaluation and results

In this section, we compare the performance of our protocol with the current dominant routing protocol: LND. We also investigate the effects of different graph types, as well as the impact of the choice for the Dovetail node. But first, we describe the simulation framework itself, and the performance and anonymity metrics we use for evaluation.

## 4.1 Simulation framework

The simulation framework is written in Python 3. The simulation first constructs a network using the *networkx* package as a directed graph. The following two graphs have been chosen for experimentation: the Barabasi-Albert graph [7] (a graph with bias towards centralization) and the Erdos-Renyi graph [8] (a random uniform graph). As a basis, the parameters $n = 100$ nodes, $m = 2$ edges are chosen. However, we will also analyse the effects when changing these parameters. The Erdos-Renyi graph uses probability instead of $m$, so here we use $p = 0.02$. After forming the graph, the edges (channels) of the graph are filled with uniformally distributed random values for *Delay* (between 10 and 100, steps of 10), *BaseFee* (between 0.1 and 1, steps of 0.1), *FeeRate* (between 0.0001 and 0.001, steps of 0.0001) and *Balance* (between 100 and 10000). The adversarial nodes are chosen to be the top 10 most well connected nodes, since these nodes are the most dangerous as adversaries.

With the network set up, we now simulate the transactions. Two random nodes are chosen as the sender and the recipient, and a transaction amount is chosen with exponential distribution between 1 and 1000. The transaction is routed using the routing algorithm from Section 3.4. If the transaction requires a multi-hop path, we use the HTLC protocol as explained in Section 2. One small difference between our simulation and the real LN is that the timeout of the HTLC's is instant, so the simulated transactions either fail or succeed instantaneously. When an intermediary forwards a multi-hop payment and the intermediary is present in the list of adversaries, the intermediary will try to deanonymize the transaction (using either the old or the modified attack from sections 2.3 and 3.5 respectively).

More information about the simulation and how to reproduce the results can be found on the github page[9].

## 4.2 Evaluation metrics

We measure the performance of both the old and the new routing algorithms using two types of metrics: cost-efficiency and anonymity. When evaluating the new routing algorithm, we compare the results of these metrics with a similar attack on LND's routing algorithm, since nearly all nodes on LN use LND[2]. For performance evaluation, we look at the following metrics: the success rate of the payments $Success$, the average fee $AVG_{fee}$ and the average hop count $AVG_{hops}$.

For anonymity evaluation, we look at the percentage of transactions attacked $TX_{att}$, the average set sizes for the source $AVG_{source}$ and destination $AVG_{destination}$, the average number of pairs $AVG_{pair}$, the percentage of attacks for whom the correct pair is present in the anonymity sets $Present$ (should be close to 100%), and finally, the number of times a singular source $Sing_{source}$ or destination $Sing_{destination}$ was found (filtering out false positives).

As mentioned before, the source/destination pairs may not necessarily be the same as the sender/recipient pairs. Instead, they provide a lower bound for finding these pairs, since finding the real source/destination takes additional effort.

## 4.3 Results

| | Barabasi-Albert | | Erdos-Renyi | |
|---|---|---|---|---|
| | Old | New | Old | New |
| $Success$ | 92.51% | 83.12% | 82.43% | 70.14% |
| $AVG_{fee}$ | 1.08 | 2.33 | 2.62 | 4.63 |
| $AVG_{hops}$ | 3.40 | 5.74 | 5.62 | 9.01 |

Table 1: Cost-efficiency metrics for both the Barabasi-Albert ($n = 100, m = 2$) graph and the Erdos-Renyi ($n = 100, p = 0.02$) graph, simulating 1000 transactions.

In Table 1, we see the effects the proposed routing protocol on cost efficiency. We can see that the average fee more than doubles for the Barabasi-Albert graph, and increases by 76% for the Erdos-Renyi graph.

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

[8] https://networkx.org/documentation/stable/auto_examples/graph/plot_erdos_renyi.html

[9] https://github.com/jsheemskerk/Attacking-Lightning-s-anonymity

| | Barabasi-Albert | | | | | Erdos-Renyi | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Old | Path 1 | Center | Path 2 | New | Old | Path 1 | Center | Path 2 | New |
| $TX_{att}$ | 85.01% | - | - | - | 92.0% | 77.03% | - | - | - | 82.98% |
| $N_{att}$ | 1313 | 1000 | 177 | 851 | 2028 | 1967 | 1263 | 200 | 1159 | 2622 |
| $Present$ | 98.02% | 99.2% | 95.48% | 99.18% | 98.87% | 73.51% | 82.66% | 64.5% | 81.88% | 80.93% |
| $AVG_{pair}$ | 24.04 | 73.25 | 368.32 | 24.59 | 78.58 | 23.23 | 135.05 | 46.67 | 25.74 | 79.99 |
| $AVG_{source}$ | 20.63 | 17.84 | 44.01 | 21.51 | 21.66 | 18.63 | 14.83 | 21.95 | 21.23 | 18.20 |
| $AVG_{destination}$ | 1.81 | 17.94 | 8.19 | 1.63 | 10.24 | 1.18 | 15.03 | 1.725 | 1.24 | 7.92 |
| $Sing_{source}$ | 13.02% | 10.8% | 0.0% | 0.0% | 5.32% | 1.98% | 3.17% | 0.0% | 0.17% | 1.60% |
| $Sing_{destination}$ | 63.21% | 4.3% | 15.25% | 67.92% | 31.95% | 49.26% | 0.95% | 21.0% | 57.03% | 27.26% |
| $Sing_{either}$ | 70.45% | 14.9% | 15.25% | 67.92% | 37.17% | 50.33% | 4.12% | 21.0% | 57.2% | 28.87% |

Table 2: Anonymity metrics for both the Barabasi-Albert ($n = 100, m = 2$) graph and the Erdos-Renyi ($n = 100, p = 0.02$) graph when simulating 1000 transactions. To attain the values in the 'New' column, we multiply each values of the path segment columns with the number of attacks for that path segment, and divide by the total number of attacks. For instance, for the average pairs: $(73.25 \cdot 1000 + 368.32 \cdot 177 + 24.59 \cdot 851)/2028 = 78.58$

Moreover, the average hop count increases by 70% and 60% respectively. The increase in fee's and hop count cause a lower total success rate per transaction, dropping by about 10% for both graph types. It seems that the cost efficiency is taking a hit; however, the anonymity metrics from Table 2 luckily show a strong improvement. Although the transactions tend to be attacked much more frequently, the effects of the anonymity attacks are drastically reduced. The reason for this can be explained when looking at the attacks for different path segments individually.

Looking at the second path segment, we see that these results are roughly in line with that of the old deanonymization attack. The destination set size is still very low, leading to a low overall number of pairs.

For the first path segment, we see that the the number of destinations is much higher. This anonymity improvement mirrors that of shadow routing. This increase in the destination set size leads to a sharp increase in the average number of pairs.

Interestingly, the Barabasi-Albert shows the largest number of pairs when under a center attack. Although not shown, this holds true even when changing the parameters $n$ and $m$. This is because both the source and destination sets are very large, generating a large number of possible pairs. Although this was unexpected, The reason there are so many destinations might have to do with the fact that the constraints for possible sources per destination are loosened. Whereas the normal attack might have filtered out many destinations as candidates because many nodes will not route through the adversarial node, in this routing protocol, this efficiency constraint does not exist. As such, any candidate node from Phase II is likely going to show up in the final destination set size. This phenomenon is not nearly as influential on the Erdos-Renyi graph.

To summarize the anonymity increase, we look at $AVG_{pair}$ and $Sing_{either}$. For both types of graphs, $AVG_{pair}$ more than triples, and the number of times a singular source or destination has been found, $Sing_{either}$, roughly halves.

Next, we look at the effects of varying the parameters $n$ and $m$ for the Barabasi-Albert graph in relation to the metrics $AVG_{hops}$ and $AVG_{pair}$. Varying parameter $n$ does not give any interesting insights and is not shown. Both of these metrics increase as the number of nodes on the graph increase. This is simply due to there being more nodes on the graph, so paths generally take longer and the number of candidate nodes increase.

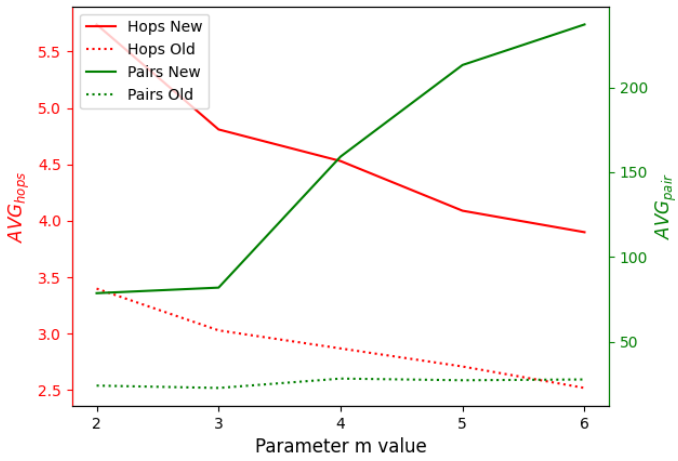
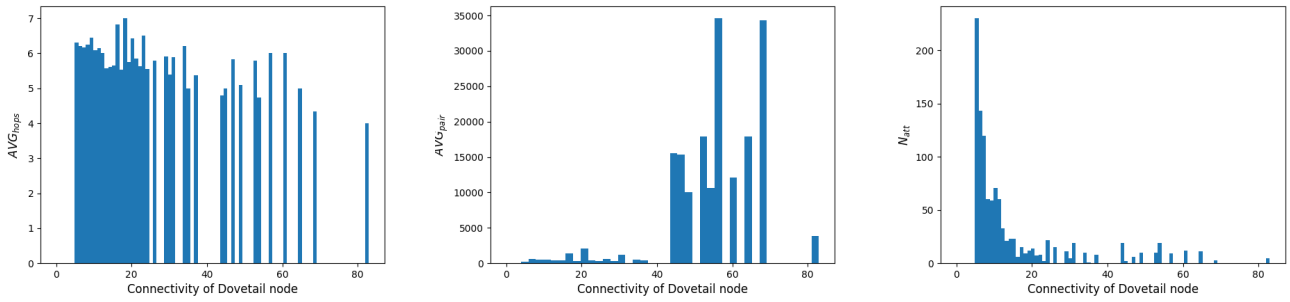



Figure 2: Varying $m$ for the Barabasi-Albert graph where $n = 100$.

More interestingly is the effect of the parameter $m$ on these metrics, shown in Figure 2. Here we see that increasing $m$ leads to a decline in $AVG_{hops}$ and an increase in $AVG_{Pair}$ for both the old and the new routing protocol. However, as the green line shows, the number of pairs increases substantially faster in the new protocol compared to the old protocol. This gives the impression that the proposed protocol is more effective for graphs with a higher density.

Finally, we also look at the effect of the connectivity of the Dovetail node in relation to $AVG_{hops}$, $AVG_{pair}$ and $N_{att}$. In Figure 3, we see the effect of the Dovetail node connectivity on each of these metrics. The main observation is that $N_{att}$ (c) decreases sharply as the connectivity of the Dovetail node increases. This gives a strong impression that a more well-connected Dovetail node is a better option than one that is less well connected. $AVG_{pair}$ (b) gives a similar impres-




(a) (b) (c)

Figure 3: Showing the relation between Dovetail node connectivity and (a) $AVG_{hops}$, (b) $AVG_{pair}$ and (c) $N_{att}$, for a large Barabasi-Albert ($n = 500, m = 5$) graph with 1000 transactions.

sion: the number of pairs increases as the Dovetail node connectivity increases. However, this behavior seems to fluctuate somewhat when changing the graph parameters (not shown), so this is less reliable. But in all cases tested, a very low connectivity resulted in a low value for $AVG_{pairs}$. Lastly, even $AVG_{hops}$ (a) seems to decrease slightly as the Dovetail connectivity increases.
All in all, the results seem to imply that having a Dovetail node with a high connectivity is more beneficial than having a Dovetail node with a very low connectivity.

## 5 Conclusion

The Lightning Network's current routing protocols suffer from a lack of randomness when constructing routes. Because of this, adversaries that are on the payment path are able to perform anonymity-compromising attacks to determine the sender and recipient. By splitting up the routing protocol into multiple path segments, we are able to nullify the anonymity-compromising attack conducted by an adversary on the payment path. Even when designing a counterattack for deanonymizing a specific segment of the modified routing protocol, we still get substantially larger anonymity sets compared to contemporary routing algorithms, more than tripling the number of pairs in total. The only exception to this is the destination anonymity set when on the second path segment, but this is a much smaller percentage of all transactions than with the old routing algorithms. We have also shown how using a more well-connected Dovetail node, or when using a more highly connected graph, the anonymity improvements are even stronger.
However, we also pay a price for this anonymity performance increase. The average fee more than doubles for some graphs, and the average hop count increases by more than $60\%$, which in turn causes the transaction success rate to drop by about $10\%$.

## 6 Discussion and further work

Although the anonymity increase seems to be quite substantial, it is not yet clear whether or not this anonymity increase is worth the cost efficiency price. The proposed routing protocol can result in paths that are drastically different from the optimal path. This is good for privacy, but it may be too drastic. The core of the problem is the deterministic nature of routing protocols. Perhaps modifying the routing only slightly (e.g. by introducing a few random hops) already introduces enough randomness to consider anonymity to be sufficient, without causing the massive increases in costs.

There are many avenues that are interesting for further research. First of all, the current path segment routing is still very simple, as the Dovetail node is most or less chosen at random. Creating a heuristic for finding good candidate Dovetail nodes can be useful here. As we have seen, more well-connected nodes seem to be better candidates. However, it could be that creating a bias towards well connected nodes might also give the adversary better heuristics.
The counterattack can also be explored in much more detail. Although we have opted to keep the adversarial attack limited to the two endpoints of a payment path, it may be possible to infer more information about the final sender or recipient from these endpoints.
Our protocol is intended to protect LN users against graph-based anonymity-compromising attacks. However, there are other types of attacks that LN is vulnerable to, such as timing attacks [15]. It might be interesting to investigate how the proposed protocol fares against these other types of attacks.
Another negative about our research is that all the data is simulated. The simulations may therefore show results that are not necessarily representative of the real Lightning Network. Extra research into simulating LN more accurately might be considered.
Finally, another very interesting field for further research is that of semi-source routing. This was briefly discussed in Section 3.2: when splitting the routing problem into multiple sub-problems, it might be possible to delegate some of the routing work to other nodes. This may have both immense anonymity benefits (for example, by allowing the use of remote private channels), but also enables new attack angles, such as when the delegated routing node is an adversary.

## 7 Research integrity and reproducibility

As mentioned before, the information shown in the Results section of this paper are all simulated. This has both posi-

tive and negative consequences. One positive aspect of this is that we are not dealing with the data of real users, and therefore, are not exposed to the risks of handling sensitive information. A negative aspect however, is of course that the data used might not necessarily be representative of the real Lighting Network. This should always be kept in mind when creating models as substitute for real things, and the Lightning Network is no exception.

One aspect of research which we value highly is that of reproducibility. All the information discussed in the Results section of this paper are acquired from the code base that is open-source and thus publicly available[10]. This should make it feasible for anyone to reproduce the results as shown in this paper, together with the parameters shown in Section 4.1. Although the simulation relies on randomness, all the data shown in Section 4 is created using the seed value 65, meaning these results can be reproduced easily by using this seed value. Even when the reader is unable to reproduce the data itself, it has also been uploaded to the github repository and can be found in the 'results' folder. For more information, see the README.

## References


[1] BOLT #7: P2P Node and Channel Discovery. Available at: https://github.com/lightningnetwork/lightning-rfc/blob/master/07-routing-gossip.md.

[2] lnchannels lightning network snapshot, 2021. Available at: https://ln.fiatjaf.com/.

[3] Andreas M Antonopoulos. *Mastering Bitcoin: Programming the open blockchain*. " O'Reilly Media, Inc.", 2017.

[4] Bitcoin Wiki. Hashed timelock constracts, 2019. Available at https://en.bitcoin.it/wiki/Hash_Time_Locked_Contracts.

[5] Rick de Boer, Stefanie Roos, and Satwik Prabhu Kumble. Improving blockchain anonymity using hop changes with partial route computation. 2021.

[6] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. Sok: Layer-two blockchain protocols. In *International Conference on Financial Cryptography and Data Security*, pages 201–226. Springer, 2020.

[7] Paolo Arash Kazemi Koohbanani, Stefanie Roos, and Satwik Prabhu Kumble. Improving the anonymity of layer-two blockchains adding random hops. 2021.

[8] Satwik Prabhu Kumble, Dick Epema, and Stefanie Roos. How lightning's routing diminishes its anonymity. In *Proceedings of the 16th International Conference on Availability, Reliability and Security*, pages 1–10, 2021.

[9] Lightning Labs. Lightning network daemon, 2016. Available at: https://github.com/LightningNetwork/lnd.


[10] https://github.com/jsheemskerk/Attacking-Lightning-s-anonymity


[10] Giulio Malavolta, Pedro Moreno-Sanchez, Clara Schneidewind, Aniket Kate, and Matteo Maffei. Anonymous multi-hop locks for blockchain scalability and interoperability. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019*, 2019.

[11] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008. Available at https://bitcoin.org/bitcoin.pdf.

[12] Mehmet Emre Ozkan, Satwik Prabhu Kumble, and Stefanie Roos. Improving the anonymity of blockchains: The case of payment channel networks with length-bounded random walk insertion. 2021.

[13] Mihai Plotean, Stefanie Roos, and Satwik Prabhu Kumble. Improving the anonymity of the lightning network using sub-optimal routes. 2021.

[14] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments, 2016.

[15] Elias Rohrer and Florian Tschorsch. Counting down thunder: Timing attacks on privacy in payment channel networks. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 214–227, 2020.

[16] Jody Sankey and Matthew Wright. Dovetail: Stronger anonymity in next-generation internet routing. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 283–303. Springer, 2014.

[17] Sergei Tikhomirov, Pedro Moreno-Sanchez, and Matteo Maffei. A quantitative analysis of security, anonymity and scalability for the lightning network. In *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 387–396. IEEE, 2020.

[18] Jin Y Yen. An algorithm for finding shortest routes from all source nodes to a given destination in general networks. *Quarterly of Applied Mathematics*, 27(4):526–530, 1970.