Improving Blockchain Anonymity Using Hop Changes with Partial Route Computation

Rick de Boer[1] , **Satwik Prabhu Kumble**[1] , **Stefanie Roos**[1]
[1]TU Delft

Abstract

The Lightning Network aims to solve Bitcoin's scalability problem by providing a way to transact with minimal use of the blockchain. Instead, payments are routed over payment channel networks. This routing is done by LN clients, which use cost functions to compute the optimal transaction path. With the use of onion routing, LN tries to hide the identity of transaction participants from each other. However, the cost functions of these routing protocols are currently too deterministic, making it possible for curious transaction participants to comprise the identity of sender and receiver by computing the optimal path themselves.
Here we show that we can increase the anonymity of this network by adding randomness to these routing algorithms. More specifically, during path computation we will randomly deviate from the optimal path by hopping to a random node and continue by computing a new optimal path from there. The unpredictability of this process improves the anonymity of the network, such that malicious nodes can identify the sender and recipient of transactions with negligible probability in most cases.

Keywords

Payment Channel Networks, Lightning, Routing, Anonymity

1 Introduction

Proof-of-work blockchains like Bitcoin have shown great promise to be used as a decentralized electronic payment method [12]. Although the implementations of their consensus algorithms have proven to be quite slow and unsustainable, as they involve a large number of miners doing difficult computations to validate even a single block [4]. This presents a scalability issue that will only become worse as the number of users increases. Other digital payment methods like Visa[1] can sustain thousands of transactions per second, while Bitcoin is currently able to sustain about ten transactions per second [5].

One promising solution to Bitcoin's scalability problem is the Lightning Network (LN) [15]. LN is a payment channel network (PCN) built on top of Bitcoin as a separate layer [5]. LN promises instant payments with low transaction costs while holding on to the security provided by the Bitcoin blockchain [17]. It accomplishes this by conducting its transactions on the PCN and only using the main blockchain in case of a dispute or during the opening/closing of a channel [5]. These transactions are often called "off-chain". Two users can open a payment channel by locking collateral on the blockchain [9]. From then on, these two users can transact with each other by redistributing this collateral. Once a user wants to close the channel, they can do so by publishing the last agreed-upon state to the blockchain such that the funds are correctly split [5]. Two nodes that aren't directly connected can still transact with each other when they are connected via other nodes. The public topology of the PCN makes it possible for the sender of a payment to find a route that connects them to their desired recipient. This is called source routing [5]. These routes are computed by LN clients, which use cost functions to find an optimal path of payment channels [9], also called the *Transaction Path*. Depending on the used client, this optimality is based on criterions like low transaction costs, time lock durations and channel reliability. LN employs onion routing [2] in an attempt to prevent information leakage to intermediary nodes. This way, an intermediary node will only be given knowledge of whom they got the payment from, and to whom they should forward it [7].

However, recent studies have shown that the network's deterministic path computation still allows malicious nodes to uniquely identify the sender or receiver for about 70% of observed transactions, thus diminishing the network's anonymity [9]. This work proposes a new routing algorithm that improves this anonymity by adding a non-deterministic element to the path computation. This new routing algorithm involves randomly deviating from the optimal route by adding a suboptimal node (hop change) to the path. When such a hop change occurs, we continue by computing a new optimal route starting from the hop. This way of routing generates transaction paths that make it harder for malicious nodes to find the sender and recipient of the payment, because

[1]https://usa.visa.com/content/dam/VCOM/download/corporate/media/VisaInc_factsheet_11012015%20(002).pdf

the randomness increases the number of potential sources and destinations. However, these suboptimal paths come with higher transaction fees and delays, so it is important to limit the number of random hops such that we do not undermine LN's high performance and efficiency.

This paper is structured as follows. Section 2 gives a detailed explanation of the inner workings of the Lightning Network. Here we will cover topics like payment channel creation, transaction routing and LN clients. In section 3 we will go over the attack strategy that adversaries can use to undermine the anonymity of the network. Next, in section 4 we will go over the new algorithm that aims to solve these anonymity problems with the use of random hops with partial route computation. In section 5 we will see how this new routing algorithm performs, by simulating it against custom deanonymization attacks and comparing it with the contemporary routing protocol. In section 6 we will discuss the results of the simulation and judge whether we improved the anonymity while still staying true to Lightning's low transaction costs and high performance. Next, in section 7 we reflect on how responsibly the research was conducted. Finally, in section 8 we will conclude and look if there is still room for improvement.

2 The Lightning Network

The Lightning Network is a layer-2 protocol constructed on top of the current Bitcoin protocol. This second layer provides a way to conduct off-chain transactions across the network without having to trust intermediary nodes that are part of the transaction [17]. The only time an LN user has to interact with the blockchain is during a dispute or to open/close a payment channel. LN is one of several other payment channel networks that operate on the same principles, so the concepts discussed here can also be applied to other PCNs. Being a PCN, LN is comprised of many users who are connected via payment channels. Here we model the Lightning Network as a graph, where the nodes represent the users, and the edges represent the payment channels.

2.1 Lightning Payment Channels

A Lightning payment channel can be opened by having two users deposit bitcoins into a 2-of-2 multi-signature address, such that any transaction would have to be signed by both of them [7]. This is also called the *Funding Transaction*. The coins that each user deposited represent their initial balance in the channel, and the sum of these coins represents the channel capacity. Users can spend coins as long as they have a positive balance in the channel, and the maximum amount of coins travelling through the channel can never exceed its capacity. Before the funding transaction is broadcast to the blockchain, each party has to sign a *Revokable Commitment Transaction* (RCT), which permits the other party to unilaterally close the channel and extract their current balance [17]. This protects users against the loss of funds when the other party becomes inactive. After the funding transaction has been broadcast, users can redistribute the initial balance by signing new RCTs. When a user decides to close the channel, they have to publish the last agreed-upon RCT. Violating this rule could result in the other party receiving all funds in the channel [15]. This measure demotivates someone from maliciously publishing an outdated RCT as an attempt to get more coins.

Up until now, we have only discussed instances where two users are directly connected via a payment channel, but more often than not this is not the case. When two parties are not directly connected, they have to route their payments through the Lightning Network. LN has a public topography, meaning that nodes are aware of other nodes on the network and their payment channels. Nodes are identified by their public key, and channels are associated with their channel identifier, capacity, and transaction fee [7]. Nodes charge these transaction fees whenever their channels are used as intermediaries. This fee consists of a base fee plus a fee that is proportional to the amount transferred:

$$fee = base\_fee + fee\_rate * amount$$

Nodes can charge different fees for their channels, and the fees can also differ when using the channel from the opposite direction. Based on this public knowledge, a sender can decide how to route their payments to their desired recipient (source routing). When routing a payment, the sender also uses onion routing to hide their identity from other nodes on the transaction path. A transaction is successful when every intermediary node passes the payment to its successor on the transaction path until the recipient is eventually reached. However, transactions can fail and one should not have to trust intermediary nodes on the transaction path. Thus, to ensure the atomicity of transactions and compliance of intermediary nodes LN enforces the use of *Hashed Time-lock Contracts* (HTLC)[2].

Hashed Time-lock Contracts

The use of HTLCs enables a sender to lock their money on the path of payment channels and only release it if they receive a certain secret only known by the receiver. The sender can set a time frame for this, which is calculated by summing the transaction delays of payment channels on the transaction path. If the sender does not receive the secret within this time frame, they can cancel the transaction and recover their funds. Below is an example of a *multi-hop-payment* using the HTLC protocol where Alice sends 1000 Satoshis to Bob, with Charlie being the intermediary node:

1. Bob sends an invoice to Alice, which contains the desired amount of coins and a randomly selected secret string hashed using the SHA256 hashing algorithm.
2. Alice computes a path to Bob and finds out that they are both connected to Charlie.
3. Charlie charges a fee of 10 Satoshis to route the payment from Alice to Bob, so Alice sends him a Sphinx packet [3] with an HTLC containing 1010 Satoshis. Alice has to make sure that the HTLC timelock is sufficiently high for the transaction to be completed.

[2] https://en.bitcoin.it/wiki/Hash_Time_Locked_Contracts

4. Charlie subtracts the fee from the total amount and sends Bob the packet with a new HTLC containing 1000 Satoshis. The new HTLC is made using the old one with a lower timelock since Charlie will receive the pre-image earlier than Alice.
5. Bob reciprocates by sending Charlie the pre-image of the hash, who can then finalize their HTLC.
6. Charlie sends the pre-image to Alice such that Alice releases the funds and the transaction is finalized.

2.2 LN Payment Routing

To conduct a transaction over the Lightning Network, one first has to compute a route to their desired recipient. These routes can be computed using a routing client. LN currently has three active clients: Lightning Labs' LND[3], ACINQ's Eclair[4], and Blockstream's c-Lightning[5]. All three of these clients use a variation of Dijkstra's algorithm, where the costs are computed by looking at transaction costs, channel capacity, lock time and previous failures of the channel. Routes are computed starting from the receiver, such that we can add the growing transaction fees to the total amount for every subsequent node. These routing clients have different *cost functions*, meaning they have different priorities when judging the optimality route. Below we will go over each client and their cost functions:

LND: This is by far the most popular routing client used within LN, as it is used by approximately 92% of the network [9]. Its cost function is given by:

$$cost = amount * timelock * rf + fee + bias$$

The amount equals the funds traveling through the channel, and the timelock describes how long those funds will be locked. rf is given by a constant risk factor of $15 * 10^{-9}$, and the bias is a value that is related to recent failures of the channel.

c-Lightning: Similarly to LND, c-Lightning focuses on minimizing the timelocks and fees, but also adds some random fuzz to the calculation. This randomness is introduced via a scaling factor, where $scale = 1 + random(-fuzz, fuzz)$. c-Lightning's cost function is:

$$cost = (amount + scale * fee) * timelock * rf + bias$$

where rf and $bias$ are related to the path length, initially set to 10 and 1 respectfully. When the path length exceeds 20, rf is set to a value close to 0, transforming the search into a shortest path problem.

Eclair: Instead of looking for one optimal path like the other clients, this client uses Yen's k shortest path algorithm [18] and randomly selects one of the results. This procedure adds randomness to the computation in an attempt to confuse attackers. As for its cost function, it first normalizes the timelock, capacity and age of channels (n_{tl}, n_{cap}, n_{age}). It then multiplies these values with default weights, resulting in the following cost function:

$$cost = fee * (n_{tl} * tl_r + (1 - n_{cap}) * cap_r + n_{age} + age_r)$$

[3]https://lightning.engineering/

[4]https://acinq.co/

[5]https://blockstream.com/lightning/

3 Attacking the Lightning Network

Now that we have a basic understanding of how the Lightning Network operates, let us look into how some of its properties can be exploited to undermine the network's anonymity. First we will define the goals and capabilities of an adversary in the network, and then we will go over the attack itself.

3.1 Adversary Model

First, we assume that the network contains a set of evenly spread adversaries with at least two active payment channels. We do not consider private channels here, as they would not be visible when using source routing. When such an adversary participates in a transaction, it becomes their goal to determine the sender and the receiver of the payment. Furthermore, we assume that the adversaries are aware of the LN topology and that they can use the HTLC payment information, like timelocks and payment amount, to their advantage. Because of onion routing, the adversaries only know the previous and next nodes on the transaction path, let us call them *PRE* and *NEXT* respectively. With this information, the adversaries can rule out all of the nodes that can not possibly be a sender or a receiver, ending up with an *Anonymity Set* of potential senders and receivers. When the sizes of the anonymity sets are low, this means that the network has low anonymity, but the reverse is not necessarily true [16]. We also assume that multiple adversaries will not be able to use each other's anonymity sets to narrow down the potential senders and receivers, because the security of HTLCs will eventually be improved such that you will not be able to tell that you are part of the same transaction [10]. Finally, we assume that adversaries will not have any knowledge of previous transactions, meaning that every node has an equal probability of being a sender or receiver.

3.2 LN Attack Strategy

The attack [9] can be divided into the following two phases:

Phase I

During the first phase of the attack, we will use the information inside the HTLC to rule out any nodes that can not be reached with the remaining timelock and payment amount. Then, we consider every possible loopless path, starting from *NEXT*, and excluding the adversary node and *PRE*. Here we start by considering every path with depth one, incrementing until we find all of the combinations. Because of the exponential growth of the attack's time complexity, we only allow a maximum depth of four for these paths. Now for every path found we run phase II.

LN also supports *Shadow Routing*[6], where the sender adds an additional value to the total timelock. This procedure makes it unreliable for an adversary to use the total timelock to predict the destination of the payment. Consequently, the timelock constraint is removed from the search when this technique is applied, resulting in larger destination anonymity sets.

[6]https://github.com/lightningnetwork/lnd/issues/1222

Phase II

Let $P = \{p_1, p_2, ..., p_{rec}\}$ be a path found during phase I, with p_{rec} being a potential receiver. We start by prepending the adversary node and *PRE* to this path, resulting in path $P' = \{PRE, A, P\}$. Next, we execute the following steps:

1. First we determine whether this path could be the result of one of the routing clients, so we can include p_{rec} in the destination anonymity set. We do this by using the previously mentioned routing clients to compute optimal paths from every node in the network to p_{rec} (When using Eclair, we compute the 3 best paths). For every node v, we check if it is part of P'. If so, we compare v's computed path P' with the subpath $\{v, ..., p_{rec}\}$ of P'. If these paths do not match we can conclude that P' could not have been computed by the routing client, so we exclude p_{rec} from the destination anonymity set and try phase II for another path.
2. Once we come across PRE during this pathfinding, we again check if PRE's computed path matches with P'. If these do not match, it means that PRE could not have been an intermediary. But we know for certain that PRE is part of the transaction path, so we conclude that PRE has to be the sender if p_{rec} is the recipient. This conclusion can be made because a sender does not have to pay fees for using their own channel, which could result in a different transaction path. If the paths do match we just include PRE in the source anonymity set and continue looking for other potential senders. When using Eclair, we conclude that PRE has to be the sender when none of the 3 paths match with P'.
3. Once we have come across both PRE and A, we start looking at new nodes differently. Now for every node v that we visit, we check if P_v contains PRE. If so, this means that v could have been the first intermediary node, so we add all of v's neighbors (except the ones in P_v) to the source anonymity set. For Eclair, if any of the 3 paths contains PRE we add the neighbours of v to the source anonymity set.

4 New Routing Algorithm

Thus, because of the deterministic nature of the current routing clients, transactions are easily traceable by adversaries. c-Lightning and Eclair contain a bit of randomness in their path computation, but their routes are still predictable in practice [9]. Here we will go over the proposed routing model that aims to solve this issue. One of the goals of this new model is that it holds up to the previously mentioned attack. Furthermore, we assume that the adversaries are aware of the new routing algorithm, so some features of the existing attack have to be rewritten. Those changes will also be documented here.

4.1 Routing Model

There are many ways one could go about adding a non-deterministic element to LN routing [13; 6; 14; 8]. We will approach this by adding (more) randomness to the path computation. This randomness is added using the following procedure: we start by computing the optimal transaction path, but during this process we randomly add a suboptimal node to the path, called a hop change. After adding this hop change, we restart the process by finding a new optimal route, starting from the hop, until it is time for another random hop. This partial route computation is repeated until we have successfully connected the sender and receiver. When finding a route, the chance of adding a hop change is related to the degree of the node most recently added to the path. This results in a larger variance of the possible paths, making the routing even less predictable. In fig. 1 we compare this new model with LN's current way of routing on a small PCN where an adversary happens to be part of both transactions.

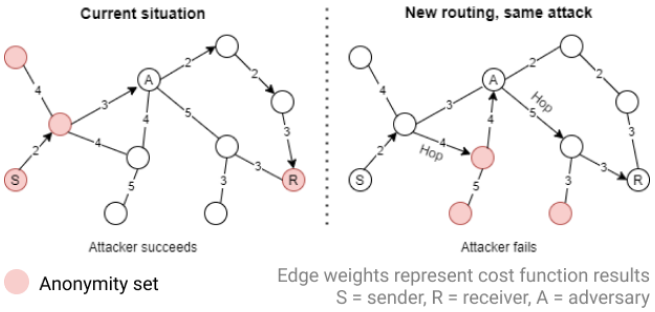

Figure 1: Comparing LN's current routing to the routing model

As expected, the adversary successfully comprises the identity of the sender and receiver when using LN's current routing algorithms on this particular graph. However, the attack fails when simulated with the new model. This happens because the attack (from section 3) expects the payment to follow the optimal path, so the use of this suboptimal path results in the sender and receiver being excluded from the anonymity set, which is exactly what we wanted. When an adversary wants to attack this model, they would have to change the original attack to be more inclusive when picking potential senders and receivers, resulting in larger anonymity sets.

4.2 The Algorithm

There are different ways to design an algorithm for this model. One could argue for a simple approach, where for every node that we add to the path, we have a chance to also add one of this node's unvisited neighbors to the path. This approach would suit our needs, but it would have a pretty severe impact on the performance, since this random node could charge very high transaction fees or could come with a large delay. Furthermore, this approach could motivate an adversary to charge ridiculous fees in the hopes of their channel being included as a random hop. The performance results of such an algorithm can be found in appendix A.

As for the approach we are actually going to be using, let us have a look at algorithm 1 and go through it step by step. Keep in mind that this is a simplified version of the real model[7], but it should give you a general idea of how it works. This pseudocode does not take transaction delays into account, and it ignores the fact that senders do not have to pay transaction fees when using their own channels.

Algorithm 1 Random Hops with Partial Route Computation

```
1: function ROUTING(G, source, target, amt)
2:     paths ← {}
3:     visited ← new Set()
4:     pq ← new PriorityQueue()          ▷ Prioritize low costs
5:     d ← {}                            ▷ Maintain distance to nodes
6:     a ← {}                            ▷ Maintain amount + fees
7:     paths[target] ← [target]
8:     d[target] ← 0
9:     a[target] ← amt                   ▷ Initialize
10:    pq.add(0, target)                 ▷ Start from receiver
11:    while !pq.isEmpty() do
12:        cost_u, u ← pq.get()
13:        if u == source then
14:            return paths[u]           ▷ Path found
15:        end if
16:        if cost_u > d[u] then
17:            continue
18:        end if
19:        visited.add(u)
20:        for v in G.neighbours(u) do
21:            if visited.contains(v) then
22:                continue
23:            end if
24:            cost ← d[u] + cost_function(G, a[u], u, v)
25:            if (random_float(0, 1) ≤ 0.1
26:              & len(paths[u]) ≤ 25) then
27:                cost ← cost * 0.2√len(paths[v])  ▷ Hop
28:            end if
29:            if cost < d[v] & capacity(u, v) ≥ a[u] then
30:                pq.add(cost, v)
31:                paths[v] ← [v] + paths[u]
32:                a[v] ← fee(u, v) + a[u]
33:                d[v] ← cost           ▷ Update values
34:            end if
35:        end for
36:    end while
37:    return []                         ▷ No path found
38: end function
```

Like other routing protocols, this model uses Dijkstra's algorithm to find the shortest path starting from the receiver. Here, Dijkstra's path length is based on the results of a cost function. This new routing model is designed to be backward compatible with any of the other clients in the Lightning Network, so the cost function on line 24 can refer to any client's cost function. When using Eclair, we execute this function k times and pick a random result. The non-deterministic element is written on line 25. The way this works is that for every unvisited node, we pick a random floating-point number between 0 and 1. If this number is smaller than 0.1, we multiply the computed cost with a value related to the current path's length, starting from 0.2 and growing sublinearly. This multiplication essentially constitutes a hop change, since it will significantly reduce the computed distance to that node and as a result, make it way more likely to be included in the path. The relation to the path length is necessary because leaving it out would make long paths very random, dramatically increasing the delay and transaction costs. Like c-Lightning, this method has a bias towards long paths. When the path length becomes greater than 25, we stop applying this multiplication because the multiplied value would else become greater than 1, as $0.2 * \sqrt{25}$ equals 1. The values 0.1 and 0.2 were found to be optimal after testing many different combinations.

[7] https://github.com/rejdeboerTU/LN-attack-simulation

4.3 Modified Attack Model

Now let us revisit the attack from section 3 and adapt it to this new routing model. We will be looking at each phase and see what needs to change to counter the new model:

Phase I: For this phase we do not really need to change anything, since this part of the attack does not exclude any nodes from being a potential sender or receiver. Although depending on the used hardware, one might want to lower the maximum depth allowed for finding a path, since this new attack will be considerably more computationally intensive than its predecessor as we will soon discover.

Phase II: The old attack relied a lot on the assumption of paths computed by the routing clients being (close to) optimal. However, with the use of the new model this assumption becomes invalid, requiring us to make the following changes to this phase's steps (the complete phase II modification can be found in appendix B):

1. Previously we would exclude a node from being a potential recipient when the path found in phase I did not match with the path(s) found in phase II. This assumption now becomes false, since paths found during phase I do not have to be optimal anymore. Instead, we have to continue and find potential senders for every path found in phase I.
2. Here we would normally conclude that PRE was the sender if P_{pre} did not match $P^{'}$. This assumption also becomes false, because now it is possible that the inclusion of PRE could have been the result of a hop change. Instead, we include both PRE and its neighbors (except for A and $NEXT$) to the source anonymity set.
3. For this step it is no longer valid to require that PRE is inside P_v, as this would rarely be the case considering the randomness. Instead, we make this requirement more inclusive by making it sufficient for A or $NEXT$ to also be inside P_v.

Most of these changes entail getting rid of some properties of the old attack. Previously, these properties allowed us to greatly narrow down our potential senders and receivers, so losing that ability will significantly increase the sizes of the

anonymity sets. Furthermore, previously we were also able to prematurely stop phase II. Now that we can not do this anymore, this phase will take a lot more time, significantly increasing the time complexity of the attack.
This modification will result in large anonymity sets, but it is the best that we can do. One could make the argument for an attack where we use the new routing model for finding paths in phase II, but the randomness would result in a completely different path being generated in the majority of cases. Using a similar attack strategy like the one used against Eclair would also not work, since the number of possible paths that the model can generate is simply too large, requiring one to try a lot of different routes which would take a lot of time while also not even being effective.

5 Evaluation

The evaluation of this new model will be done as follows: First we will go over the metrics that are used to measure the performance and anonymity of the routing models. With these metrics in mind, we will be able to go over the results and start reasoning whether this new model has improved anonymity over the contemporary model, while still maintaining high performance. In addition, this section will contain information about the simulation framework and the dataset that was used to generate the results.

5.1 Metrics

Anonymity Metrics

- PA: The percentage of transactions that were attacked by at least one adversary.
- PS: The percentage of attacks where the adversary successfully finds the sender and recipient pair.
- $|A_s|$: Average source anonymity set size.
- $|A_d|$: Average destination anonymity set size.
- S_s: The percentage of transactions where the source anonymity set only contains the sender.
- S_d: The percentage of transactions where the destination anonymity set only contains the recipient.
- fp_s: The percentage of attacks where the adversary did not manage to link the source to the recipient, given that the recipient was found (false positives).
- fp_d: The percentage of attacks where the adversary did not manage to find the recipient (false positives).
- Cor_s: The correlation between the size of the source anonymity set and the distance between the sender and adversary.
- Cor_d: The correlation between the size of the destination anonymity set and the distance between the adversary and receiver.

Performance Metrics

- H_{avg}: Average number of intermediary nodes per transaction.
- FR_{avg}: Average fee percentage (fee / total amount).
- D_{avg}: Average delay per transaction.
- *fail*: Percentage of failed transactions.

5.2 Dataset

The simulation uses a recent snapshot of the Lightning Network[8], which contains data from all publicly published nodes and channels. The framework omits channels with no capacity and nodes that do not publish their timelocks and transaction fees, resulting in a total of 4791 nodes and 28997 channels being considered. The adversary role is equally distributed to nodes based on their centrality. Having an adversary set with varying centralities is important for generating reliable results because the centrality of an adversary highly influences their attack results. For instance, an adversary with high centrality will be connected to more nodes, resulting in them being an intermediary of more transactions. This means that they would be able to perform more attacks than an adversary with low centrality. On the other hand, being connected to more nodes also means that there are more possible routes that a payment could go, resulting in larger anonymity sets. We calculate the centrality of a node based on the following four metrics: betweenness, eigenvector, closeness and degree. For each of these metrics, the Networkx Python library[9] was used to generate a list of data for every node. For the simulation we consider a total of 20 adversaries, of which 10 are picked randomly, 5 have high centrality, and 5 have low centrality.

5.3 Simulation Framework

The simulation framework[10] is written in Python and provides a way to test routing models on a snapshot of the Lightning Network. The simulation differs from LN in the fact that it assumes that transactions are instant, ignoring the impact that locked collateral can have on the network. When the simulation is run, it starts manually assigning the adversary role to nodes according to their centrality. After selecting the adversaries, the simulator starts conducting transactions by pairing two random nodes as the sender and receiver, followed by the execution of the pathfinding algorithm using the sender's routing client. When using Eclair, the framework uses a more general version of Dijkstra's algorithm to find the k shortest paths, because Yen's algorithm is slower than Dijkstra when we are not dealing with negative weights [9]. The transaction amounts are exponentially distributed between 1 and 100000 Satoshis. These transactions can fail when the pathfinding algorithm did not find a valid transaction path where the channel balances were sufficient to forward the payment. This simulation does not take any of LN's cryptographic features into account, as it is not necessary for the results. For every adversary on a transaction path, the simulation runs the deanonymization attack. The results of these attacks, together with the performance results of the transaction, are written to a JSON file. A script can be run to read this file and convert it to useful data, which can be used to evaluate both the routing model and the modified attack model.

[8] https://ln.fiatjaf.com/

[9] https://networkx.org/documentation/stable/reference/algorithms/centrality.html

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

5.4 Results

Below you can see the performance results gained by simulating 5000 transactions. The left column enumerates the earlier mentioned performance metrics, and the other columns compare the values obtained from simulating the old and the new routing algorithm.

Performance results		
Metric	Old routing	New routing
H_{avg}	2.43	11.95
FR_{avg}	5.38%	6.52%
D_{avg}	95.27	106.12
$fail$	8.73%	11.45%

The anonymity results were gained by simulating 1000 transactions, which is considerably less than the performance results because of the longer simulation duration. The results below are shown in a similar style to the performance results except for the extra column on the right. This column shows the results of simulating the new algorithm with the modified attack model.

Anonymity results			
Metric	Old routing	New routing	New attack
PA	38.35%	66.94%	66.94%
PS	99.0%	8.38%	54.64%
$\|A_s\|$	298.36	8.12	1135.30
$\|A_d\|$	51.90	55.41	131.63
S_s	42.46%	3.51%	0.0%
S_d	57.82%	22.97%	22.84%
fp_s	0.0%	83.19%	24.48%
fp_d	1.40%	68.65%	61.35%
Cor_s	0.82	0.04	0.33
Cor_d	0.40	0.04	-0.07

Performance Evaluation

When looking at the performance results, we observe the following: First of all, the hop changes increase the average number of hops per transaction by a substantial amount. This is caused by the fact that most hop changes originate from a central node, which can not be revisited. So the lack of these central nodes forces the pathfinder to use nodes with lower centrality, increasing the hop count. We also observe a significant increase in transaction failures as a result of the added hop changes. This happens because sometimes the pathfinder will not be able to find a transaction route where every channel has sufficient balance to forward the payment. Luckily, this new routing algorithm does not seem to dramatically impact the transaction fees and delays.

Anonymity Evaluation

As expected, we see a large success rate when attacking the old routing algorithm using the attack model from section 3. This success rate dramatically plummets when using the new routing algorithm. This happens because the old attack will stop searching when encountering a suboptimal path, which is not very effective when most of the paths found by the new model are suboptimal. These results are also reflected by the low anonymity set sizes when using the old attack. With the use of the new attack, we observe that both average anonymity set sizes increase by a lot. The new model also considerably decreases the number of singular anonymity sets, even getting rid of singular source anonymity sets. Because of the increase in the average number of hops, there is a larger chance of an adversary participating in the transaction, which increases the number of attacks.

We also see that the use of the new model increases the number of false positives by a lot, even when using the new attack. As for the old attack, the adversary makes false conclusions in terms of the potential sender and destination nodes. Although one would expect those false positives to go down when using the new attack. The reason these false positives did not decrease is because the adversary only tries to find recipients that are at most 4 nodes away from them, so the increase in average path length makes the recipient unreachable in a lot of cases. Additionally, the increased complexity of the attack makes it too hard to find all of the potential sources within a reasonable timeframe, causing an increase in the source false positives.

However, there are still relatively many attacks that end up with a singular destination anonymity set, even after the routing and attack change. This is bad, but making changes to the routing model will not further decrease these numbers, because an adversary will always be able to gain such a set when they are the penultimate node on the transaction path. In such a situation, it becomes trivial for them to conclude that $NEXT$ has to be the destination since they can use the delay information inside the HTLC.

As for the correlation metrics, there is a decent correlation between the size of the anonymity sets and the distance of the sender and receiver from the adversary when using the old routing model. This happens because short paths are relatively easy to predict by an adversary, becoming increasingly difficult as the path grows. The new routing completely nullifies this correlation for the old attack, because the false assumptions result in small and close to constant anonymity set sizes. With the modified attack, we see a growth in the correlation between source anonymity set size and adversary distance. This happens because the modified attack model gets rid of the false assumption, resulting in anonymity sets that become increasingly difficult to predict as the adversary distance increases. The modified attack also results in a slightly negative correlation between the destination anonymity set size and adversary distance. This is again caused by the fact that most recipients are located outside of the adversary's reach, resulting in a smaller number of potential recipients as the adversary distance grows.

6 Discussion

After evaluating these results, we can conclude that the use of the new routing model brings both advantages regarding and disadvantages. Here we will be putting these results in a broader context by going over these positives and negatives and giving some clarification where necessary.

6.1 The Drawbacks

The drawbacks of using this new routing algorithm become immediately apparent after looking at the performance results. The increase in the average number of hops is a negative, because more intermediary nodes mean more potential points where a transaction could fail, and it increases the amount of locked collateral on the network. Furthermore, it raises the chances of an adversary participating in the transaction. Next, the increase in the number of payment failures is also clearly negative, but there are ways to remedy this. For instance, we could simply run the algorithm again in the hopes of a valid route being found, or we could turn down the randomness for the second run. Another way to combat these failures would be to split your payment up into smaller payments such that there is a higher chance of channels being able to forward it. As for the other performance metrics, we only see a small increase in the average transaction fees and delay over the contemporary model.

6.2 The Benefits

However, what we lose in performance, we gain in anonymity. The introduction of the new routing model had a great impact on almost every anonymity metric. First of all, we see a big increase in the anonymity set sizes, indicating that it is now harder for an adversary to narrow down their potential senders and receivers. Because of this, the source anonymity set becomes pretty useless for an adversary, since they are just too big to conclude anything from. This also happens in some cases for the destination anonymity sets, and in other cases the recipient is too far away to be found by the adversary. Even though there are still many cases where the adversary uniquely identifies the correct recipient, there is not really anything that we could change to the routing to fix this. Using a technique like shadow routing, where we add an additional value to the timelock, could prove useful here. This would result in the adversary not being able to reliably use the delay information, forcing them to be more inclusive when picking potential recipients. This might bring down the number of singular destination anonymity sets found by the adversary, but it would also introduce some other problems. The reason why shadow routing has not seen much use in LN is because the additional timelock values increase the damage that timing attacks can do to the network. This is where an adversary intentionally lets payments fail at the end of the delay timelock in an attempt to clog up the network with locked collateral [11].

7 Responsible Research

Now let us reflect on this research and argue if it was conducted responsibly. We will do this by going over some privacy and environmental issues correlated to this field, followed by a section on the reproducibility of the research.

Environmental Issues

One of the main goals of the Lightning Network is to make Bitcoin more scalable and sustainable by providing a way to do off-chain transactions. Bitcoin's current network traffic mostly consists of on-chain transactions which have to be validated by miners. This process costs a lot of power, thus releasing tons of CO2 into the atmosphere. The Lightning Network has a more sustainable approach where one would only have to interact with the blockchain in case of a dispute, or during the creature/closure of a payment channel. The increase in anonymity by this routing algorithm might motivate more people to start using the Lightning Network for their Bitcoin transactions, decreasing the number of on-chain transactions.

One could argue that increasing the number of intermediary nodes for off-chain transactions would also increase the number of disputes. This would result in more on-chain transactions, decreasing the earlier mentioned sustainability provided by LN. However, the number of blockchain transactions caused by disputes is relatively small when looking at Bitcoin's network traffic as a whole, so the effect of this path length increase can be seen as negligible.

Privacy Considerations

During this research we did not actually interact with the Lightning Network. The transactions used for testing were all simulated, so it did not cause any disruptions to the network. Furthermore, the used snapshot contained data that was already known to the public, so there was no usage of any sensitive information.

Reproducibility of the Research

After reading this paper, one should have the necessary knowledge to reproduce this research. With the code repository that was shared earlier in the paper, one will be able to simulate the attack and come to the same conclusions. Every transaction was simulated with the same set of adversaries, which is defined in the code.

The only issue here is that running the simulation on the snapshot requires powerful hardware. The results of this research were produced with the aid of DAS [1], a powerful supercomputer. Because few people have access to such a system, the framework also offers a way to simulate attacks on smaller graphs. These graphs posses some of LN's properties, like varying centrality, so simulations will produce results similar to those of the snapshot.

8 Conclusions and Future Work

It can be concluded that this new routing model does indeed improve the anonymity of the Lightning Network. The model provides great resilience against deanonymization attacks that would previously be very effective, and it even holds up against attacks that are designed to counter it. Thus, the added random hops make it infeasible for an adversary to deanonymize the sender and the receiver in most cases. Although this anonymity comes at the cost of some hits in performance: besides the small increase in transaction costs and delay, the longer transaction paths introduce more potential points of failure, so one should ask themselves if

anonymity is worth the trouble of having to retry transactions. Consequently, even though this routing model would be great for someone who often conducts large-scale transactions and wants to reduce the risk of being targeted by an attack, there is no real need to use this model for small transactions that do not require anonymity.

Future research should be conducted to resolve the two main issues that this model possesses: the long transaction paths and the vulnerability of a recipient when an adversary is the last intermediary on the transaction path. As we know, the increase in path length is a result of the random hops. At the moment, it is almost guaranteed for a hop to happen when encountering a central node because of its high degree. When hopping from a central node, it forces the pathfinder to use many less central nodes, resulting in a longer transaction path. To fix this, one might want to reduce the chance of hopping to a node, depending on the degree of the current node. This way, there is still a chance that we follow a more optimal path when the current node has a high degree.

The recipient vulnerability could potentially be solved with shadow routing. With this technique, an adversary would not be able to use the delay information inside the HTLC to conclude that their neighbour node has to be the recipient. Although the additional delay value added by shadow routing should not be too large, as that would expose a weakness to other attacks.

References

[1] Henri Bal, Dick Epema, Cees de Laat, Rob van Nieuwpoort, John Romein, Frank Seinstra, Cees Snoek, and Harry Wijshoff. A medium-scale distributed system for computer science research: Infrastructure for the long term. 2016.

[2] Chen Chen, Daniele E. Asoni, David Barrera, George Danezis, and Adrian Perrig. Hornet: High-speed onion routing at the network layer. *CCS'15: The 22nd ACM Conference on Computer and Communications Security*, 2015.

[3] George Danezis and Ian Goldberg. Sphinx: A compact and provably secure mix format. *2009 30th IEEE Symposium on Security and Privacy*, 2009.

[4] Arthur Gervais, Ghassan O. Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. *CCS '16: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[5] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, PatrickMcCorry, and Arthur Gervais. Sok: Layer-two blockchain protocols. *International Conference on Financial Cryptography and Data Security*, 2020.

[6] Joran Heemskerk, Stephanie Roos, and Satwik Prabhu Kumble. Improving anonymity of the lightning network using multiple path segment routing. 2021.

[7] George Kappos, Haaroon Yousaf, Ania Piotrowska, Sanket Kanjalkar, Sergi Delgado-Segura, Andrew Miller, and Sarah Meiklejohn. An empirical analysis of privacy in the lightning network. 2020.

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

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

[10] Giulio Malavolta, Pedro Moreno-Sanchez, Clara Schneidewind, Aniket Kate, and Matteo Maffei. Anonymous multi-hop locks for blockchain scalability and interoperability. 2019.

[11] Ayelet Mizrahi and Aviv Zohar. Congestion attacks in payment channel networks. 2020.

[12] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

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

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

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

[16] Andrei Srjantov and George Danezis. Towards an information theoretic metric for anonymity. *International Workshop on Privacy Enhancing Technologies*, 2002.

[17] Giovanni Di Stasi, Stefano Avallone, Roberto Canonico, and Giorgio Ventre. Routing payments on the lightning network. *IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom) Green Computing and Communications (GreenCom)*, 2018.

[18] Jin Y. Yen. An algorithm for finding shortest routes from all source nodes to a given destination in general networks. 1970.

A Alternative Routing Results

Performance results	
Metric	Results
H_{avg}	4.70
FR_{avg}	11.67%
D_{avg}	122.44
$fail$	13.42%

As you can see, the performance is considerably worse when using this routing method. The only advantage in performance is that this approach has a lower average hop count. However, this lower hop count also results in transaction paths that are easier to attack, which has a negative impact on the anonymity. Furthermore, since this approach has a chance to ignore the transaction fee of a channel, it could motivate malicious nodes to charge ridiculous routing fees, further decreasing the performance.

B Complete Phase II Modification

1. We start by using the LN clients LND, Eclair and c-Lightning to compute optimal paths from every node in the network to p_{rec} (When using Eclair, we compute the 3 best paths). We keep computing these paths till we have eventually visited PRE and A.
2. Once we come across PRE during this pathfinding, we add both PRE and its neighbours, except for A and $NEXT$, to the source anonymity set. We add these neighbours, because PRE could have been a random hop from any of those nodes. We also consider PRE to be a possible sender, because A could have been the first intermediary node.
3. Once we have come across both PRE and A, we start looking at new nodes differently. Now for every node v that we visit, we check if P_v contains PRE, A or $NEXT$. If so, this means that v could have been the first intermediary node, so we add all of v's neighbors (except the ones in P_v) to the source anonymity set. For Eclair, if any of the 3 paths contains PRE, A or $NEXT$, we add the neighbours of v to the source anonymity set.