Improving the Anonymity of the Lightning Network using Sub-Optimal Routes

Mihai Plotean[1] , **Stefanie Roos**[1] , **Satwik Prabhu Kumble**[1]
[1]TU Delft

Abstract

The Lightning Network is a second layer payment protocol built on top of Bitcoin, which is scalable and has reduced transaction fees. It does so by eliminating the need to broadcast every transaction to the whole network. When one user wants to send a payment to another, the routing protocol generates a path between them that is always fast and cost efficient. The low degree of randomness in the existing routing protocols during path selection allows an adversary to compromise the anonymity of the sender and recipient.

In this work, we propose a new routing algorithm that is less predictable when creating a transaction path. We show that this increases the anonymity of the users of the Lightning Network by creating an attack on the new routing protocol. The attacker tries to identify the potential source and recipient of a transaction. Our results suggest that there is a trade-off between the offered anonymity and transaction fees; it is not possible to get higher anonymity at no cost by designing a non-deterministic routing algorithm.

1 Introduction

Blockchain-based payment systems promise fast peer-to-peer transactions, worldwide payments and low processing fees. However, the process of passing on transactions, broadcasting them to each participant and storing them has proven to be slow. Contrary to centralized payment networks such as Visa, which supports over 47,000 transactions per second, the limit for Bitcoin [9], the largest cryptocurrency in the world, is 7 transactions per second [11]. This problem appears due to the decentralized nature of blockchain.

Lightning [11] is a second layer (also called off-chain layer) added on top of Bitcoin which has the main purpose of solving its scalability issue as well as lowering processing fees. The main idea behind is that transactions are not recorded in the blockchain, which is expensive and time-consuming. Instead of directly interacting with it, two users first open a channel and lock an amount as collateral. Then they locally conduct transactions until someone decides to close the channel, after which the distribution of funds takes place. Direct interaction with the blockchain takes place just in the following situations: opening, closing a channel or resolving disputes in the network. When sending a payment, a direct channel with the receiver is not necessary. A payment can be routed via multiple intermediaries from the source to the destination. This route is determined by the source using a cost function that depends on the routing protocol and is based on multiple criteria such as fees and delays. There are three main routing protocols in Lightning: LND, c-Lightning and Eclair. Each of them chooses one of the best paths in terms of some globally known network properties. Furthermore, onion routing is used to improve anonymity, which makes it harder for an intermediary to track off-chain payments. The Lightning Network was also created to reduce privacy concerns. Opposite to on-chain transactions, payments in Lightning are not broadcasted and stored by everyone on the network, making it more private. The high transaction throughput, speed and low processing fees make it the most promising solution to accepting cryptocurrency payments in daily life.

However, studies have shown that the anonymity of users is under question because of the highly deterministic nature of the protocols that Lightning uses for payment routing. By acting as an adversary in a transaction, one can gain sensitive information such as who are the possible senders and recipients, which could also disclose payment habits. In [12], the author states that the endpoints of a transaction can be deanonymized using timing-based attacks. The results show that an adversary controlling more than 10 nodes could deanonymize transaction sources and destinations with a precision of over 50%. A paper where it is assumed that the adversary is located right after the sender or right before the recipient also demonstrates that the anonymity of the sender and recipient is under risk [13]. Furthermore, another work that does not make any assumptions about the location of the adversary shows that if any of the nodes in the transaction path is controlled by an adversary, there is a 70% probability of uniquely identifying the sender or recipient and an 8% probability of uniquely identifying both the sender and recipient [8]. This is considering that the adversary only makes use of globally known properties such as timelock and routing protocol.

The problem tackled in this research project is to improve on Lightning's anonymity issue by making it harder to infer the sender and the recipient, as illustrated in Figures 1 and 2. We design a new non-deterministic routing protocol and create a simulation where every participant in the network uses this routing strategy. By simulating transactions on two graphs generated to resemble the Lightning Network and introducing attackers that observe a high proportion of all transactions, our results suggest that we can reduce by a factor of 2.6 the probability of uniquely identifying the receiver and bring the probability of uniquely identifying the sender close to zero. However, this leads to an increase in the average hop count by 66-67% and an increase in fees by 107-110%.

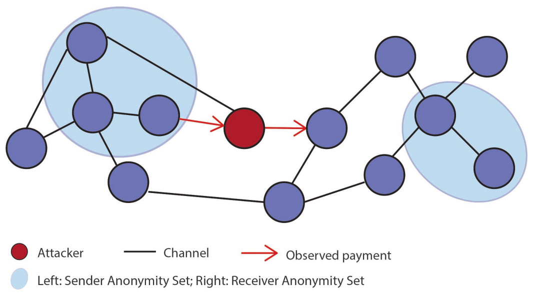

Figure 1: Inferring the sender and recipient anonymity sets considering that the route is created by an existing routing protocol.

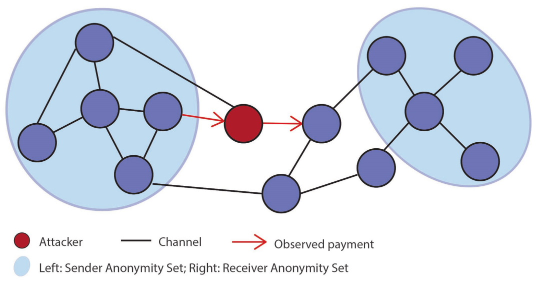

Figure 2: Inferring the sender and recipient anonymity sets considering that the route is created by a new non-deterministic routing protocol. The anonymity sets become larger compared to Figure 1.

The remainder of the paper is structured as follows. In Section 2, we first analyze how routing works in Lightning and what are the existing Lightning implementations. In Section 3, we design a new non-deterministic routing algorithm that uses sub-optimal routes with the objective of improving anonymity. To evaluate the protocol, we create an attack on the new routing strategy in Section 4, followed by an analysis based on various metrics from the performance and anonymity aspects in Section 5. These are compared to the metrics from a current protocol. Section 6 concludes how the anonymity was impacted and discusses future points of improvement.

2 The Lightning Network

The paper describing how the Lightning Network works was published in 2016 [11], and since its launch in 2018, it has gained over 12.000 active channels and 50.000 nodes as of 21 June 2021[1]. Lightning is a peer-to-peer payment channel network where parties can make payments in bitcoin without recording them on the bitcoin blockchain. The main blockchain architecture is called layer-1. Lightning is called a layer-2 solution as it is built on top of layer-1 and does not require any changes to the underlying blockchain.

In this section, we describe how the routing in Lightning works and what are the existing routing protocols. We then describe how Hash Time Locked Contracts work as they make multi-hops payments possible in the absence of a trusted party.

2.1 Routing

The Lightning Network can be represented as a graph. Each participant represents a node in the graph and each channel between two participants is denoted by an edge. When a payment is made, the funds are transferred directly from the sender to the receiver in one hop if there is a direct channel between both parties. The more interesting and common case is when a direct channel is absent. This is usually the case as opening a channel requires interaction with the blockchain layer, which incurs high fees. A payment from node A to node B can be routed via multiple nodes, which we call intermediaries. All intermediaries must have sufficient collateral to forward the payment. This type of payment is called a multi-hop transaction and allows payments between all members of the network. When a multi-hop transaction takes place, edges are chained together by the source to form a path to the destination. The way this path is created depends on the routing protocol that the source is using. One of the crucial properties with respect to privacy that the routing algorithms should satisfy is that routing paths should be found without disclosing transaction values, the sender and the receiver [5]. Currently, there are 3 main routing protocols in Lightning: LND, c-Lightning and Eclair. Each of these is similar in the fact that they determine the path using a shortest path algorithm. For LND and c-Lightning this algorithm is Dijkstra and for Eclair it is Yen's algorithm [15]. However, the main difference is that each of them uses a different cost function. The factors considered in the cost function can be channel capacity, which is equal to the sum of the balance held by each participant in the channel, charged fees, delays, and past channel failures.

LND

LND is the most common Lightning implementation with over 90% of the nodes in the network using it. It favours paths with low timelocks and low fees. Furthermore, LND tries to avoid paths that contain channels that led to payment failures in the past. The algorithm does so by first finding the channel where the failure occurred and then assigning a *bias* value to it. The *bias* decreases as more time passes from the point of failure.

c-Lightning

C-Lightning also gives priority to low fees and delays. At the same time, it adds some randomization when selecting paths. This is done through a parameter named *fuzz* in the

[1]https://1ml.com/statistics

cost function. Another parameter, *bias*, has the purpose of disfavouring long paths. The path is calculated using Dijkstra's shortest path algorithm.

Eclair

Eclair is the only of the three algorithms that does not always choose the best path based on the cost function. It chooses randomly one of the three best paths, which are computed using Yen's Algorithm [15]. Yen's Algorithm computes k shortest paths by using any efficient shortest path algorithm with Dijkstra in the case of Eclair. Additionally to low timelock and low fees, the capacity and the age of the channel are considered.

2.2 HTLC

Multi-hop payments in Lightning where no third party is trusted are possible due to Hash Time Locked Contracts (HTLCs) [14]. A HTLC is based on two components: hashlocks and timelocks. The hashlock prevents a party from getting its funds until a specified piece of data is revealed. The timelock allows a party to get their funds locked by the HTLC back in case the beneficiary does not acknowledge the receipt of the payment before a predetermined time. The main idea behind is that the amount to be paid is locked and it is not given until a secret hash is provided. If the time that the parties agreed on in the HTLC expires, the locked funds are given back to the sender.

Suppose Alice wants to transfer an amount equal to 100 satoshis[1] to Bob. Let Carol be the only intermediary in the transaction. Bob, the recipient, generates a random number, computes its hash, and sends it to Alice. Alice uses this hash to set up a HTLC with Carol - her successor on the path computed by the routing algorithm. The created contract states that Alice will pay Carol 105 satoshis if the latter will obtain the random number, called preimage, that was used to generate the hash sent by Bob. The extra 5 satoshis is a fee charged by Carol for acting as an intermediary and forwarding the payment. The amount of 105 satoshis becomes locked in the HTLC and nobody can use them for now. To make sure everyone can get their locked funds back in case the hash is never provided, there is also a timelock associated with the contract. If Carol fails to provide Alice the preimage in that timelock, Alice gets her 105 satoshis back. Now, a new HTLC is set up between Carol and Bob. The amount of 100 satoshis is locked and subtracted from Carol's balance. The conditions of the contract remain the same as in the HTLC between Alice and Carol, only the amount and timelock being different. Since Bob was the one to actually create the hash, he has the preimage and thus uses it to unlock the funds sent by Carol and obtain 100 satoshis. By doing so, the preimage becomes available to Carol. Carol uses it to receive her 105 satoshis as stipulated by the HTLC. The same steps are illustrated in Figure 3.

As demonstrated, the transaction took place without a direct channel between Alice and Bob. Nobody had to trust each other in the path. Furthermore, each participant would get their money back in case somebody would not fulfill their job in the path.

[1]A satoshi is equivalent to 100 millionth of a bitcoin.

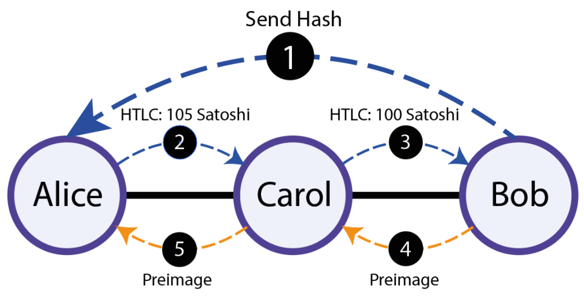

Figure 3: Alice sends Bob 100 satoshis with no direct channel between them. Besides the amount to be sent, HTLCs save the timelock to keep track if a payment has expired.

3 Designing the Routing Algorithm

To design a routing protocol that improves anonymity, we need to know the capabilities and the objective of the adversary that tries to compromise it. For this reason, we first introduce the adversarial model. Then, we enumerate the possible options when designing a new non-deterministic routing strategy and describe the option that we chose.

3.1 Adversarial Model

An adversary **A** has the goal of determining the sender and recipient of a transaction it is a part of. We use the same assumptions as the adversary model in [8]. **A** knows the design of the protocol and its parameters. The adversary is a computationally bounded node in the network that has access to the same information as any other node. Furthermore, **A** is an honest-but-curious attacker, meaning that the adversary forwards the payment to the next node in the path as needed, but also uses publicly available information from the transaction and the network to find out two sets: the set of potential senders and the set of potential recipients. Note that it is possible for two or more adversaries to combine their observations by computing the intersection of the anonymity sets that they obtained. This would lower the size of the anonymity sets, making it easier to determine the actual sender and receiver.

3.2 Routing Algorithm

The three main Lightning implementations described in Section 2 choose the best or one of the best paths in terms of fees, delays, previous node failures and other properties. The route computation of LND and the low degree of randomness introduced by c-Lightning and Eclair is not enough to make sender and recipient anonymity sets large enough as described in [8]. Our objective is to create a new non-deterministic routing protocol that will improve the anonymity of Lightning. At the same time, we make the algorithm compatible with any of the cost functions from the existing routing protocols.

When designing a non-deterministic routing protocol with the use of sub-optimal routes, there are multiple possibilities: adding Random Hops [7], Hop Change with Partial Route Computation [3], Length-bounded Random Walk Insertion [10], Multiple Route-Segments using a "Dovetail" Node [6].

We believe that even a simple routing strategy can significantly improve the anonymity of both the sender and recipient considering the specified adversarial model. In this section, we create a protocol that combines two simple strategies that are applied consecutively. Both of these would not require major changes to the existing Lightning implementations. In the first step, *n* random hops from the source are performed. In the second one, we randomly choose one of the best *k* paths from the node we arrived at in *Step I* to the destination. Note that we cannot rule out that performing the steps the other way around would provide even higher anonymity.

Step I - Random Hops

The first step starts by generating a random number *n* between 0 and *max_hops*. The impact of the variable *max_hops* on anonymity is studied in Section 5. However, it should be low as increasing it would result in longer paths and thus higher fees for the sender. Then, *n* random hops are taken starting from the source. In case cycles exist, they are removed after the path computation. The algorithm goes on to the second step. An example of this process is illustrated in Figure 4.

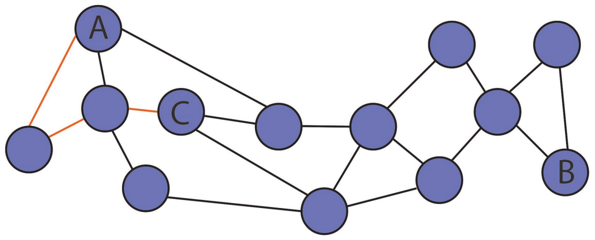

Figure 4: Step I of the algorithm. A wants to make a payment to B. The random hops are represented by the orange edges. C is the node after 3 hops.

Step II - *k* Best Paths

In the second step, Yen's algorithm is used to compute *k* shortest paths to the recipient. The starting node is the final one from the path generated by the random hops in *Step I*. From *k* paths, we randomly choose one of them. The impact of the variable *k* on anonymity is also studied in Section 5. If the starting node was the source, we simply return the path. Otherwise, the path in *Step II* is appended to the one computed in the first step, giving the final path of the transaction. We check for cycles once again and remove them if present. Finally, we make sure that we have obtained a valid path by checking that the total capacities of the channels on the path are enough to forward the payment. This cannot be done in *Step I* as random hops start from the source, but the amount to be forwarded by the node where *Step II* starts, which includes the fees, has to be calculated from the recipient. Thus, if the obtained path is not valid, we start again from *Step I*. A full path computation example is illustrated in Figure 5.

3.3 Runtime complexity

The first step has time complexity $O(n)$ as we essentially get a random node *n* times, where *n* is the number of random hops to perform. The second step involves Yen's algorithm, which

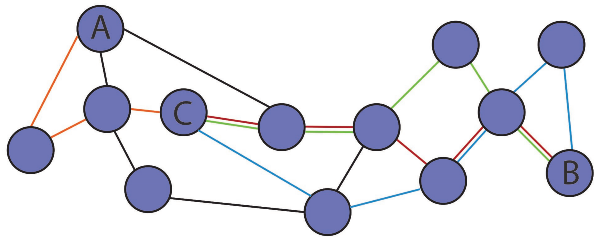

Figure 5: Step II. Three best paths are computed starting from node C to the destination (Node B) - denoted by red, blue and green. From these, one random path is chosen.

computes the *k* best paths by finding the shortest path first and then computes *k* - 1 deviations of the shortest path. The runtime depends on the shortest path algorithm that it uses. We assume it to be Dijkstra algorithm. Dijkstra algorithm has a time complexity of $O((V+E)log(V))$ when using a priority queue using binary heaps. In each iteration, Yen's algorithm calls Dijkstra at most *V* times. Considering that there are *K* iterations, Dijkstra is called *KV* times in the worst-case [2]. Thus, the time complexity of the first and second steps of the routing algorithm together is $O(KV(V+E)log(V))$.

4 Attacking anonymity

To show that the routing protocol described in Section 3 improves the anonymity of the participants in the Lightning Network, we need to design an attack on it. We first describe the attack step-by-step and then discuss its run-time complexity.

4.1 Attack design

An adversary **A** can start the attack when it observes a transaction as an intermediary. The attack is split into two phases. *Phase I* of our attack does not differ from the one described in [8]. The only information the adversary has about the transaction path is which node is its predecessor, and which node comes next after itself. We denote these nodes as *PRE* and *NEXT*. In *Phase I*, the attacker finds all loopless paths starting from *NEXT* where the sum of subsequent timelocks in the path add up to the timelock value it knows. It also makes sure that the total capacity of every channel in the path is enough to forward the payment. The attack continues in the next phase.

The main idea behind *Phase II* is that the adversary tries to find out at which node did the routing algorithm stop the random hops and chose one of the *k* best paths. Let *X* be this node. Furthermore, we denote as $P_i = \{p_1, p_2, ..., p_r\}$ a path found in *Phase I*, where $p_1 = NEXT$ and $p_r = rec_i$ is the receiver in the path. Prepending *PRE* and **A**, the nodes that come before *NEXT*, we get $P_i' = \{PRE, \mathbf{A}, p_1, p_2, ..., p_r\}$. Thus, P_i' represents a potential sub-path of the observed transaction that goes from the node preceding the attacker, to the potential receiver. There are three possible cases:

1. There were no hops and one of the best *k* paths was chosen starting from the source.
2. *X* is located before *PRE* in the transaction path.
3. *X* is one of the nodes in P_i'.

In *Phase II*, the attacker determines if P'_i could be a path that our routing algorithm would choose, and consequently include rec_i into the set of potential receivers. If it is, the attacker tries to find out who are the potential senders of the transaction corresponding to that path. One important property that is used to find *X* is that any subpath of a shortest path is itself a shortest path. Note that when there were no hops made, this property cannot be used to directly determine a potential sender as the attacker does not know the balance of the first channel that was used to calculate the path. However, in this case, it can be used to find the potential second node after the sender.

- **1**: Let *N* be a node in P'_i and let $P'_i[N :]$ be a sub-path of P'_i containing all the nodes from *N* to rec_i. We also denote the *k* shortest paths from *N* to rec_i as $P^N = \{P^N_1, P^N_2, ..., P^N_k\}$.

  The attacker first computes the *k* shortest paths from *A* to rec_i, i.e. $P^{\mathbf{A}} = \{P^{\mathbf{A}}_1, P^{\mathbf{A}}_2, ..., P^{\mathbf{A}}_k\}$. If any of the paths $P^A_i \in P^A$ equals $P'_i[\mathbf{A} :]$, then **A** is a possible intermediary in the shortest path chosen by the routing algorithm. The attack goes to **2** to determine if **A** is *X*, *PRE* is the sender or *X* is located before **A**.

  If none of the paths equals $P'_i[\mathbf{A} :]$, it means that *X* can only be located after the adversary. Furthermore, if this is the case, the routing algorithm performed at least 2 random hops, which are: from *PRE* to **A**, and from **A** to *NEXT*. If the *max_hops* value is less than that, P'_i is discarded and rec_i cannot be the destination. Otherwise, the attack goes to **3** to check which node after the adversary could be *X*.

- **2**: The attacker computes P^{PRE} - the *k* shortest paths from *PRE* to rec_i. If none of the paths $P^{PRE}_i \in P^{PRE}$ equals P'_i, there are two possibilities: either no hops were made and *PRE* is the source, or **A** is indeed *X*. Thus, since it is known that one of the hops is from *PRE* to **A**, the possible senders are *PRE* and all the nodes that can be reached within *max_hops - 1* hops from *PRE*. If any of the paths equals P'_i, then *X* is located before **A**, or some of the random hops were identical to a part of one of the best paths. Since the attacker cannot distinguish the latter from the former case, the possible senders are all the nodes from which the shortest path to the recipient contains P'_i as a sub-path, as well as all the nodes that are up to distance *max_hops* from them.

- **3:** The adversary computes P^N, where N is in the range starting from the second up to including *max_hops*-th node in P'_i. The nodes after are not considered as the random hops cannot start after *PRE*. If any of the *k* best paths equals to $P'_i[N :]$, then N could be *X*. The possible senders are all the nodes that up to *max_hops - index* hops away from *PRE*, where *index* is the index of N in P'_i. That is because all the hops up to *N* are known. Note that it could happen that some of the random hops were identical to a part of one of the best paths, which would allow considering nodes that are at a closer distance from *PRE* than *max_hops - index*. However, since the attacker cannot know that this actually happened, it still needs to consider as potential sources all the nodes that are up to *max_hops - index* hops away from *PRE*. In case that no equal path is found for the nodes in the range, P'_i is discarded and rec_i cannot be the destination.

There is no proof that the attack we propose is the strongest. Hence, with a better attack, we might find the potential sources and recipients even more accurately, making the anonymity lower.

4.2 Attack Complexity

Phase I of the attack has a time complexity of $O((Deg_{max})^d)$ [8], where *d* is the maximum hop count that we consider starting from *NEXT* to find the potential recipients, and Deg_{max} is the maximum degree of any node in the network.

In *Phase II*, in the worst case we have to calculate *k* best paths for every node besides the last in the sub-path obtained from *Phase I*, which is equal to *d* + 2 times. Considering that we use Yen's algorithm, which as discussed in Section 3.3 takes $O(KV(V+E)log(V))$ time, it means that the run-time complexity is $O((d+2)KV(V+E)log(V))$. Furthermore, we have to calculate *k* cheapest paths from all nodes to the destination. This can be done with a generalized version of Dijkstra, which has a time complexity of $O((V+E)log(V))$. Thus, the total time complexity for all the destinations is $O((d+2)KV^2(V+E)log(V))$. The complexity of both *Phase I* and *Phase II* is the product of their complexities, that is $O((Deg_{max})^d(d+2)KV^2(V+E)log(V))$

5 Evaluation

In this section, we show how the anonymity and efficiency of Lightning are impacted if our new non-deterministic routing algorithm is used. Most importantly, we study how the source and recipient anonymity sets change and what are the trade-offs. The routing algorithm described in Section 3 contains two parameters: *k* and *max_hops*. We show how the change of these parameters impacts various metrics.

5.1 Metrics

For attack evaluation we use the following metrics from [8]:

- Number of attacked transactions divided by the total number of transactions, which we denote as R_{att}.
- The correlation $CorrD_S$ of the size of the sender anonymity set to the distance between the adversary and the sender. $CorrD_R$ is a similar metric, but for the receiver.
- The proportion of attacks $Sing_S$ which returned a singular source anonymity set, singular recipient $Sing_R$, both singular sender and recipient $Sing_{both}$.
- The proportion of the attacks that completed *Phase I* of the attack $Comp_I$.

In addition to these, we also use the following metrics:

- Average size of the recipient anonymity set $Avg_{|R|}$ and sender anonymity set $Avg_{|S|}$.

- The proportion of attacks where the correct source was not included in the source anonymity set, denoted as FP_S. Similarly, for the recipient, we have FP_R.
- The average fee Avg_{fee} that was paid by the sender.
- The average number of hops Avg_{hops} in the transaction path.

5.2 Simulation Model

To simulate the transactions and our attack, we use as a basis the code[2] written in Python[3] and the simulation model described in [8]. It allows using either a snapshot of the Lightning Network or a randomly generated graph for simulation. Furthermore, the code contains all three routing algorithms discussed in Section 2.1. We extend[4] the present generalized version of Dijkstra that computes 3 best paths to any number of *k* best paths. To find the nodes that are at a maximum distance of *max_hops*, we implement the Breadth-first search algorithm.

5.3 Dataset and Parameters

To study how the fees and the number of hops are impacted by the new routing algorithm, we used a snapshot of the Lightning Network from lnchannels[5]. It contains all the public information available in Lightning such as which are the connected edges, total capacities, delays and charged fees. The simulation model that we use removes all the inactive nodes and channels that did not publish their policies or have been closed. After this operation, we obtain a network with 4.791 nodes and 28.997 channels. 92% of the nodes use LND as the routing algorithm.

Running the attack on a snapshot of Lightning is time-consuming. Given the limited time available that we have in hand, we use two different randomly generated small graphs to resemble the Lightning Network: one according to the Barabási–Albert preferential attachment model [1] and an Erdős-Rényi graph [4]. These are generated using the NetworkX[6] Python library. Each of the graphs contains 200 nodes and around 800 edges. The properties that are globally known in Lightning are also generated uniformly and randomly. The base fee that the node charges is between 0.1 and 1. The fee rate is between 0.0001 and 0.001. The timelocks are between 10 and 100 and the channel balances are between 100 and 10000 satoshis.

For our evaluation, we consider that all the nodes use the routing algorithm that we described in Section 3 combined with the LND cost function. This is a good baseline as almost all nodes use LND as their client. We simulate 300 transactions between random senders and recipients. The amount to be sent is distributed uniformly with a minimum of 1 and a maximum of 1.000 satoshis. In the attack, we consider as potential recipients all the nodes that are at a distance of at most 4 from the adversary. Increasing the depth is possible, but would lead to a significant increase in the computational complexity, making the attack unfeasible considering the computational resources that we have in hand.

As adversaries, we assign 10 nodes with the highest centrality. Nodes that have a high centrality are more likely to be chosen as intermediaries and thus observe a transaction.

5.4 Results

We first check the impact on anonymity of each of the two strategies that our routing protocol combines. We use the Barabási–Albert graph to illustrate the results in Figures from 6 to 10. On the left, we keep $k = 1$ fixed, meaning that after the random hops are performed, the best path is chosen. When $max\_hops = 0$, we get the same transaction path as in LND. On the right, we keep $max\_hops = 0$ fixed, meaning that we skip the random hops step and randomly choose from the source one of the *k* best paths. Hence, when $k = 1$, the transaction path is the same as in LND. If we used Eclair cost function, we would get the same transaction path as in Eclair for $k = 3$. Furthermore, we provide the 95% confidence interval $CI_{0.95}$ for the data samples that were used to generate each graph.

In Figure 6, we can see that the proportion of destination anonymity sets that are singular decreases approximately by a factor of 2, from 60-65% to 30%, when $max\_hops = 7$ or $k = 10$.

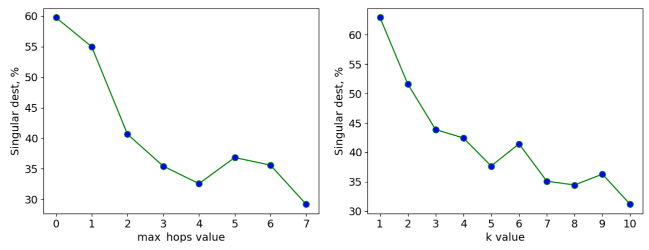

Figure 6: The change of the proportion of singular destination anonymity sets. Left: $CI_{0.95} = [32.9, 49.9]$, Right: $CI_{0.95} = [34.92, 48.47]$

In Figure 7, we plot the same graphs for singular source anonymity sets. The decrease is much more significant compared to the one depicted in Figure 6. A *max_hops* value of 5 is enough to almost eliminate the source anonymity sets consisting of only one node. There is also a decreasing trend for the right graph. The proportion of source anonymity sets consisting of one node is around 2% when $k \geq 5$.

In Figure 8, we see that the proportion of attacks that returned both only one sender and recipient decreases since there is a decreasing trend for both these metrics separately. The metric is close to zero for the maximum value that we have chosen for *max_hops* and around 1 for the maximum value of *k*.

In Figure 9, we represent the average number of hops in the transaction path. As the number of hops in the first step of the routing is generated from a uniform distribution, the expected

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

[3]https://python.org/

[4]https://gitlab.ewi.tudelft.nl/cse3000/2020-2021/rp-group-51/rp-group-51-mplotean

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

[6]https://networkx.org/documentation/stable/reference/generators.html

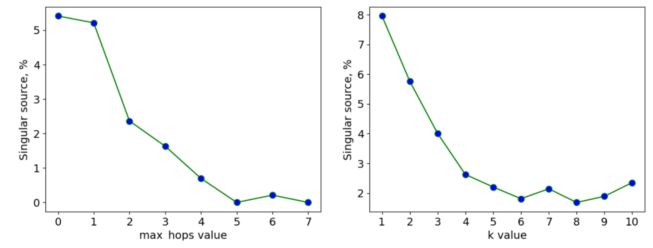

Figure 7: The change of the proportion of singular source anonymity sets. Left: $CI_{0.95} = [0.07, 3.81]$, Right: $CI_{0.95} = [1.75, 4.73]$

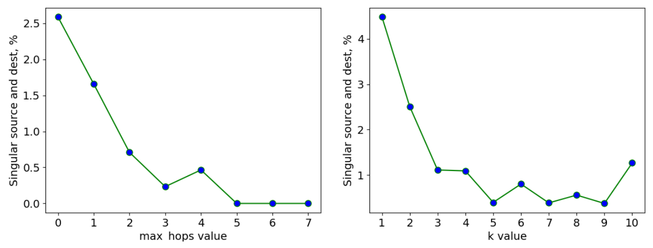

Figure 8: The change of the proportion of both singular source and destination anonymity sets. Left: $CI_{0.95} = [0.00, 1.5]$, Right: $CI_{0.95} = [0.37, 2.22]$

value is $max\_hops/2$. However, since when performing random hops we do not keep track of the visited nodes and we remove the cycles only after the path is computed, the actual average number of hops from the random hops step is lower than $max\_hops/2$. When no hops are performed and the best path is chosen, the destination can be reached in approximately 4 hops. Adding more random hops before choosing the best path increases the hop count. In this case, it goes to around 5.75 when $max\_hops = 7$. In the case of increasing *k*, we see that the average number of hops increases from 4 when $k = 1$ to 5.2 when $k = 10$. Even though a cheap path is not always one with a low number of nodes, this is usually the case. Consequently, the hop count tends to increase as we choose randomly of the *k* best paths. A higher hop count means that there is a bigger chance of payment failure. Another downside of longer path lengths is that there is a bigger probability that there can be an adversary in the path which could try to deanonymize the sender and recipient.

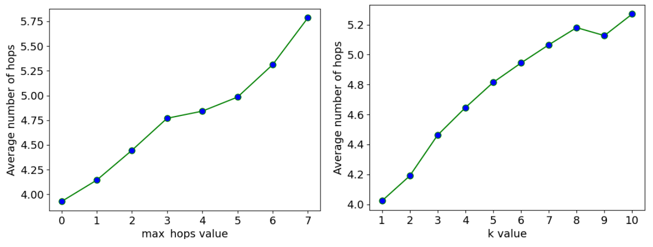

Figure 9: The change in the amount of hops in the transaction path. Left: $CI_{0.95} = [4.29, 5.15]$, Right: $CI_{0.95} = [4.46, 5.08]$

In Figure 10, we show that the increased anonymity comes at a cost. When choosing the best path directly, we see that the average fee is around 1.3. As *max_hops* increases, it reaches a value between 2.3 and 2.4 when $max\_hops = 7$. Similarly, the average fee is around 1.8 when $k = 10$. For the same path length, performing random hops leads to a higher fee than directly choosing one of the *k* best paths. For instance, the number of hops is around 4.4 for both $max\_hops = 10$ and $k = 6$ as can be seen in Figure 9, but the fee is lower for $k = 6$ with the difference being approximately 0.3. This is because the charged fees is one of the main criteria in the cost function used to compute the best paths. When performing random hops, the fees are ignored, which leads to a more expensive transaction.

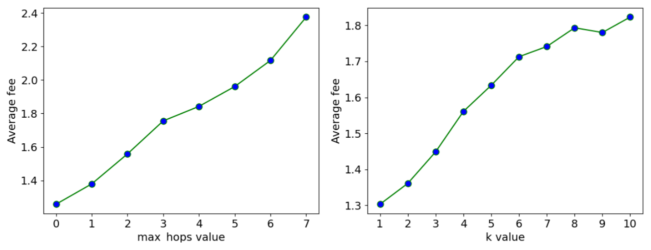

Figure 10: The change in the fees charged by the nodes in the transaction path. Left: $CI_{0.95} = [1.47, 2.05]$, Right: $CI_{0.95} = [1.48, 1.75]$

Now, in Tables 1 and 2 we compare metrics of 3 different parameter pairs based on the two randomly generated graphs. The first pair corresponds to the metrics we would obtain if all the nodes used LND as their client for routing. The last pair showcases high values for *max_hops* and *k* that could be considered reasonable. The second one represents an in-between parameter pair. The key results that can be observed are as follows:

- The number of attacked transactions increases as we choose higher parameter values. This is most likely linked to the increasing average number of hops.
- The proportion of attacks that have a singular sender anonymity set decreases by a factor of 12 for $max\_hops = 3$, $k = 5$, and goes to zero for $max\_hops = 7$, $k = 10$ in case of the Barabási–Albert graph. For the Erdős-Rényi graph, the decrease is by a factor of 2 and 2.3 for $max\_hops = 3$, $k = 5$ and $max\_hops = 7$, $k = 10$ respectively.
- The proportion of attacks that have both a singular sender and recipient anonymity set is 0 or close to 0 for the pair $max\_hops = 7$, $k = 10$.
- For both graphs, the proportion of attacks that have a singular recipient anonymity set decreases by a factor of 2 for $max\_hops = 3$, $k = 5$ and by 2.6 for $max\_hops = 7$, $k = 10$. The decrease between the second and third pairs is smaller than between the first and second due to the low distance between the adversary and the recipient. We found out that the adversary is located right before the recipient in 80% of the cases when a singular recipient anonymity set is returned. In 95% of the cases, it is at a distance of at most 2.
- In case of the Barabási–Albert graph, the average size of the recipient anonymity set increases by a factor of 4.5

Metric	$max\_hops = 0$ $k = 1$	$max\_hops = 3$ $k = 5$	$max\_hops = 7$ $k = 10$
R_{att}	0.87	0.91	0.95
$CorrD_S$	0.63	0.24	0.06
$CorrD_R$	0.41	0.44	0.36
$Sing_S$	0.06	0.005	0.00
$Sing_R$	0.60	0.32	0.23
$Sing_{both}$	0.26	0.002	0.00
$Comp_I$	0.33	0.13	0.07
FP_S	0.02	0.07	0.14
FP_R	0.04	0.14	0.24
$Avg_{\|R\|}$	2.03	9.2	9.64
$Avg_{\|S\|}$	20.5	19.1	17.7
Avg_{hops}	3.93	5.28	6.55
Avg_{fee}	1.26	1.96	2.65

Table 1: Attack metrics for 3 parameter pairs based on the Barabási–Albert graph.

Metric	$max\_hops = 0$ $k = 1$	$max\_hops = 3$ $k = 5$	$max\_hops = 7$ $k = 10$
R_{att}	0.52	0.62	0.61
$CorrD_S$	0.55	0.29	0.21
$CorrD_R$	0.48	0.49	0.34
$Sing_S$	0.03	0.015	0.013
$Sing_R$	0.59	0.30	0.23
$Sing_{both}$	0.03	0.005	0.004
$Comp_I$	0.44	0.23	0.15
FP_S	0.06	0.16	0.19
FP_R	0.08	0.24	0.31
$Avg_{\|R\|}$	1.49	3.08	3.56
$Avg_{\|S\|}$	26.63	50.8	43.5
Avg_{hops}	4.49	6.15	7.52
Avg_{fee}	1.57	2.30	3.25

Table 2: Attack metrics for 3 parameter pairs based on the Erdős-Rényi graph.

for $max\_hops = 3$, $k = 5$, and by a factor of 4.8 for $max\_hops = 7$, $k = 10$. The increase is less significant for the Erdős-Rényi graph. The factors are 2 and 3 for the same parameter pairs.

- There is an increase in the false positives for both the sender and recipient, which could be linked to the decrease in the proportion of attacks that completed Phase I. This means that an adversary is less likely to include the true sender and recipient in the anonymity sets.
- The correlation between the size of the sender anonymity set and the distance between the adversary and the sender becomes negligible for the second and thirds pairs.
- For both graphs, the average hop count increases by 35-36% in the second pair and by 66-67% in the third pair. The average number of hops increases as the expected value of the random hops that are performed increases. Furthermore, by increasing *k*, the random path chosen is likely to be longer.
- The increased anonymity comes at a cost. The average fee increases by 46-55% in the second pair, and by 107-110% in the third pair.

Last, we show how our new non-deterministic routing strategy impacted the average fee and hop count based on the snapshot of the Lightning Network. The results are represented in Table 3. We can see that these are slightly different from the results we got based on the random graphs. The fee increase from $max\_hops = 0$, $k = 1$ to $max\_hops = 3$, $k = 5$ is equal to 72%. From $max\_hops = 0$, $k = 1$ to $max\_hops = 7$, $k = 10$, it is equal to 130%. The average hop count increases by 60% in the case of the second parameter pair, and by 107% in the case of the third.

5.5 Ethical Implications

The anonymity evaluation of the routing protocol described in Section 3 requires designing and performing an attack on the Lightning Network. To achieve more accurate results, a live attack on the network is preferred. However, this could disclose the anonymity and payment habits of participants in the network.

Metric	$max\_hops = 0$ $k = 1$	$max\_hops = 3$ $k = 5$	$max\_hops = 7$ $k = 10$
Avg_{hops}	3.27	5.24	6.78
Avg_{fee}	1.41	2.18	3.26

Table 3: Average hops and fees based on the Lightning snapshot.

We perform the attack on randomly generated graphs. These graphs have the purpose of simulating a real network, where balances, fees, delays, and other globally known properties are random. A more realistic scenario is performing the attack on a snapshot of the Lightning Network, where the aforementioned properties are real, which leads to a more realistic scenario. In both cases, the transactions are generated so that the sender, receiver, and amount to be paid are random. Thus, we believe that due to this randomness, we do not put at risk the anonymity of any participant of Lightning.

6 Conclusions and Future Work

In this paper we have studied how the anonymity of the Lightning Network can be improved by introducing a new non-deterministic routing protocol. Our evaluation indicates that combining one or two simple routing strategies can significantly improve the anonymity of the participants in the network, but leads to an increase in the fees.

Our results are promising and should be validated by considering other graphs and the possibility of colluding attackers. Given that we only considered two types of random graphs of small size, an open question remains how does the anonymity we got differ from the one we would obtain by running the same attack on larger graphs, a snapshot of Lightning, or even in a real-world scenario. Moreover, we have not investigated the anonymity in case of colluding attackers. Multiple adversaries could combine their observations, making the anonymity lower. Further studies are also needed to estimate what are the optimal parameters in our routing algorithm that offer the best trade-off between fees and anonymity. Finally, there could be a better algorithm that could provide a better trade-off; for instance, one that performs the steps of our strategy the other way around.

References

[1] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[2] Eric Bouillet, Georgios Ellinas, Jean-François Labourdette, and Ramu Ramamurthy. *Path routing in mesh optical networks*, pages 128–129. John Wiley & Sons, 2007.

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

[4] P. Erdös and A. Rényi. On random graphs. *Publicationes Mathematicae Debrecen*, 6:290, 1959.

[5] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. Sok: Layer-two blockchain protocols. In *Financial Cryptography and Data Security (FC)*, 2020.

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

[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*, 2021.

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

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

[11] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments. 2016. Available at: https://lightning.network/lightning-network-paper.pdf.

[12] 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*, 2020.

[13] 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)*, 2020.

[14] Bitcoin Wiki. Hashed timelock contracts. 2019. Available at: https://en.bitcoin.it/wiki/Hash_Time_Locked_Contracts.

[15] 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), 1970.