Improving the Anonymity of Layer-Two Blockchains Adding Random Hops

Paolo Arash Kazemi Koohbanani[1] , **Stefanie Roos**[1] , **Satwik Prabhu Kumble**[1]
[1]TU Delft

Abstract

The Bitcoin Lightning Network is a layer-two solution that promises instant payments, scalability, and low transaction fees on top of the Bitcoin blockchain. In case there is no direct channel between the sender and receiver, the routing algorithm uses source routing and a shortest path algorithm to determine the hops in a transaction. However, the lack of randomness in the routing decision allows an attacker to de-anonymize either sender or receiver, if they happen to be one of the nodes in the transmission path. The guarantees offered by the onion routing style algorithm are not enough to ensure anonymity when little to no randomness is used when choosing the path. Here we show how it is possible to modify the path finding algorithm keeping backward compatibility. It increases anonymity between the sender and receiver adding random hops to the already computed shortest path. Anonymity and efficiency metrics are then analysed with respect to an adversary that is aware of the full protocol implementation. Furthermore, assuming a protocol-aware adversary, an attack is designed, and it is concluded to be successful at most 53% of the time and singularly de-anonymizing both parties in 1% of the cases. The average number of hop counts increases by approximately two and the average fee paid by the sender increases by 4.77 times. Our results suggest a possible increase in the anonymity offered without a significant impact on the complexity of the lightning protocol implementation. However, transaction fees and payment success ratio should be analyzed further, especially for low-value transactions.

1 Introduction

Proof-of-work (PoW) blockchains gained significant adoption in the past years as more people started using them for different purposes such as investment, store of value, or payments [1]. The average amount of transactions per second for the Bitcoin [2] blockchain is between 3 and 7 in contrast to the thousands of transactions that traditional centralized systems can achieve already [3]. Because of the limited block size, 1MB, the number of transactions per second cannot be easily extended above 10. A proposed solution to this problem, that allowed to scale the blockchain without a change in the consensus algorithm, was to create a layer-two blockchain on top of the main blockchain, now considered layer-one. Multiple implementations of layer-two blockchains exist such as channels, commit-chains, or refereed delegation [4]. The channel implementation only requires interaction with the blockchain when opening, closing a channel, or one of the parties involved in a transaction decides to dispute it. A dispute happens whenever two nodes disagree on the state of the network or the current balance of the channel. For example, one node might try to close the channel distributing all the remaining balance to himself, which will generate a dispute from the other party. The layer-one blockchain will then be used to resolve disputes as that has already been proven to be a reliable mean to agree on a global state between multiple untrusted parties.
The Bitcoin Lightning Network (LN) [5] is a layer-two solution that promises instant payments, scalability, and low costs on top of the existing Bitcoin blockchain. Transactions executed on the LN are not recorded on the layer-one blockchain except for the opening and closing of the 2-of-2 multisignature address [6] when locking and releasing the collateral, and resolving disputes. Transactions can occur between two direct channels that agree on opening a channel between themselves or they can be routed along the network to reach the final destination. In both cases, disputes can be resolved on the Bitcoin blockchain by carrying out the disputed part of the transaction directly on it [7].
Opening or closing a channel requires sending a transaction on the layer-one blockchain. This can be an expensive operation if transaction fees are high. Channels can then be either private or public, the latter are broadcast to the entire network and they can be used by other nodes to process payments. Routing a payment along the network makes it possible to avoid the opening cost but requires a way to determine how to route the payment across the network to reach the correct destination [8]. When routing a payment the sender and receiver should not be known to the intermediary nodes, only the previous and next nodes in the path should be revealed at each hop. This is implemented using an onion routing style algorithm where the packets that are transported only reveal the minimum amount of information to process the payment

to the next node [9]. The LN uses source routing, the path is determined by the sender of the transaction who decides the hops that will occur when routing it. Every public node actively broadcasts information about the fee that it keeps when routing a transaction and the time limit after which the funds will be returned in case of a failed payment. The sender can then choose the nodes to use to route its payment and will usually optimize for lower fees especially when the transaction amount is low, one of the use cases for the LN [10].
As such different implementations have been developed to optimize the path finding algorithm, the three most used lightning implementations use modified versions of Dijkstra algorithm [11] with different cost functions [12]. LND [13] is the most used lightning implementation and uses Dijkstra algorithm with an additional penalty for nodes that recently failed to route a payment. C-lightning [14] also uses Dijkstra shortest path algorithm but to reduce failures it adds a bit of randomization in the cost function. Eclair [15] instead uses Yen's Algorithm [16] where it randomly chooses from the 3 shortest paths available. Overall the variations use a path finding algorithm with little to no randomness added in the decision making, resulting in very few possible paths to choose from. Although onion routing style has been successful in maintaining privacy for the web, it was shown that this is not enough to guarantee privacy and anonymity on the LN [12; 17].
Whenever a transaction has to go through a node, the amount has to be locked until the transaction is completed or it expires. To enforce time expiration of payments, a Hashed Time Locked Contract (HTLC) [18] is used. The contract cryptographically ensures that if the sender does not send the funds in time, the locked funds can be recovered safely. It works by generating a random SHA256 [19] value of which the receiver has the preimage, the hash value is then propagated to all nodes in the transaction. To redeem the funds, the receiver of the contract needs to provide the preimage of the hash value determined at the start of the transaction [20]. Every public channel sets its own expiration time in terms of number of blocks to wait. This information is publicly broadcast so that it can be factored in when calculating the shortest path. The current HTLC time is known by each node in the chain so that it can be verified whether it expired or not. A low expiration time means that the number of remaining hops for that transaction are low with higher probability.
There have been already attempts to attack the LN taking advantage of the HTLC and their susceptibility to timing attacks. This allows an attacker to gain information about the parties involved in specific transactions by controlling nodes on the path. In the case of [21], the time difference between accepting to process a payment and its completion is calculated. Subsequently, a maximum likelihood estimator is constructed for the possible destinations matching the calculated time. Sources are estimated sending a failed payment message and calculating the time difference when the payment is retried. Additionally in [12], the knowledge about different routing algorithm implementations is used along with the HTLC to discard possible pairs, singularly de-anynomizing the parties involved in 8% of the cases. The routing implementation and its predictability have also been vulnerable to Denial-of-Service attacks and given that most transactions pass through a limited number of nodes they could be easily exploited [22].
Our work focuses on evaluating the possibility to modify the routing algorithm to add enough randomness such that being a node in a transaction and knowing the HTLC value will not be enough to determine a singular anonymity set with a probability higher than 5%. The research will focus on analysing the attack to the routing algorithm, explaining a modification to it that will introduce randomness into the path finding process, and showing two possible attack strategies implemented against it.
In the following sections, we will start by describing how the current de-anonymization attack [12] works in section 2. Section 3 will then be dedicated to model the modification to the routing algorithm and its integration within our simulation framework, followed by the design of a new attack taking into account the introduced randomness. In section 4 we will list the evaluation criteria for our simulation and critique the results obtained after the modification to the algorithm. Section 5 will discuss the results obtained and compare them to other alternatives analysing their advantages and dis-advantages, following Section 6 will focus on the conclusions of the work and future research. Section 7 will analyse the ethical aspects and reproducibility of the research being done.

2 Attacking Lightning

This section will be dedicated to explain the workings of recent attacks to lightning's anonymity. For a more complete description we refer to [12].

2.1 De-anonymization Attack

The attack assumes there is a set of adversaries spread across the network with multiple public channels opened. Private channels are not taken into consideration as in the case of source routing the sender needs to be aware of them. Moreover the adversaries are not able to determine whether they are on the same transaction path. It is assumed that the contracts used are anonymous multi-hop locks [23], generating a different hash value for each hop in the chain. Far apart nodes have no way of knowing whether they are on the same transaction path. This means that if the same transaction is attacked multiple times, the adversaries cannot share the information obtained and take an intersection of the anonymity sets. The attack starts when the adversary happens to be on the transaction path and it is divided into two phases. In the first phase, the attacker collects all possible destinations reachable while the second phase is used to discard invalid paths and create a set of potential source and destination pairs.

Phase I

When the adversary receives a transaction to route, the current HTLC is known, that is, how many blocks are left until the transaction timeout. The previous and next hops are also known. Starting from the next node, a search is performed to go through all possible nodes that could be a destination of the transaction, until there are no more nodes or the depth of the path is higher than four. The average hop count per transaction is less than seven, therefore stopping at depth four should be large enough to reduce the number of false positives [24]. During the search, the advertised time lock value of each hop encountered on the possible constructed path is subtracted from the previous one. The sum of all the time locks along the constructed path will therefore be equal to the HTLC value the adversary has. If the recipient HTLC value goes below zero, the hop cannot be a possible destination because a transaction would always timeout in the worst case, the path is discarded. A similar reasoning can be done for values higher than zero, the time lock value would be more than the one needed, causing a transaction to wait for a longer time in the worst case. However, if shadow routing [8] is used the time lock value will become greater than zero for most transactions. In this case, the sender decides to add an offset to the HTLC value to reduce the chances of nodes guessing the intendent recipient of the transaction. If shadow routing is used, a search can be performed only using the graph connections without considering the time lock, we will assume it is not for the rest of the paper.

Phase II

Once all possible destinations are found, the search for possible sources that could match the destination is started. Starting from the previous node, we go through all possible connected channels that might be a source for this transaction and run the routing algorithm to see if there is a possible match. Because the routing algorithms are known, as explained in the previous section, we can take the possible computed path and check if it would be the one chosen by any of the implementations. In the case of Eclair the path will be discarded if it does not match any of the top three shortest paths. Running the routing algorithm enables to discard many paths that would not be cost effective reducing the anonymity set size. Nodes can publicly broadcast the implementation they are using, this information can be used by the attacker to only run the advertised implementation when discarding paths. However, the broadcast information might not be reliable, therefore an adversary should consider all implementations to reduce the possibility of false positives.

3 New Routing Algorithm

This section is dedicated to explaining the simulation framework we used to model the lightning network and the attacks executed on it as well as detailing how the modification to the routing algorithm is applied.

3.1 Simulation Framework

The simulation framework is written in Python and it is used to create the network, start transactions between nodes, route payments across the network, attack transactions, and evaluate results. The network of nodes is initialized from a snapshot of the lightning network where only public channels that broadcast a public policy are used. The rest is discarded along with channels with no capacity. The adversaries along the network are chosen manually based on their centrality. A mix between high and low values are taken to have different views on the effectiveness of the attack. Transactions are simulated by choosing two nodes at random and initiate a payment between them. The HTLC packets are not encrypted due to the attack not requiring it, only the time lock value has to be taken into account. Routing a payment amount is an instantaneous operation as well as attacking it in case an adversary happens to be on the path. When routing a payment, the path finding algorithm corresponds to the advertised implementation that the node broadcasts. When using the advertised implementation a modified Dijkstra algorithm is also used in the case of Eclair to return the top three paths, this is done to simplify the implementation complexity. The obtained paths should not differ from the ones that Yen's algorithm finds especially for low values of k. The same generalized version will be used to model the modification of the algorithm to make sure all implementations are taken into account and a possible attack would be less effective.

Attacking payments

The routing algorithm locking the funds and moving the amount across the network is also responsible to attack the transaction in case an adversary occurred in the path. The attack is started whenever the adversary receives the information regarding the current time lock and the amount of the transaction it needs to route. Transactions are processed one after the other, therefore the simulation does not consider concurrent payments. Chances of locking funds and failing to process a payment decrease, however, in a healthy network of nodes most of the transactions should succeed. It should still be taken into consideration that multiple payments could be routed from the same node at the same time. The cost function used for LND discards the notion of bias and failed payments, which should not meaningfully impact the routing decisions made by the nodes. It is still possible to conduct the attack even when a transaction fails as long as the transaction reaches the adversary node giving it the HTLC information to compute the anonymity sets.

3.2 Routing Algorithm Modification

The algorithm modification we created is backward compatible with any current implementation of the LN. Adoption does not require consensus from a majority of the network because the LN uses source routing. This means, the sender determines all hops the transaction will go through when deciding to route a payment across the network. The decision to use a modification or a different implementation to compute the path is left to the senders and cannot be known unless they voluntarily broadcast it.

Once one or more short paths between the sender and receiver are calculated the hops are added substituting edges of the path at random. The algorithm goes over all edges of the found path and randomly decides whether or not to add an extra hop. If a hop needs to be added, it goes through all possible nodes that connect the extremes of the edge and randomly chooses one of them to form a new connection. For example, if there is an edge $A \rightarrow B$, a hop can be added if there exists a C for which a payment can flow from $A \rightarrow C \rightarrow B$, considering all channels have enough capacity. Each time a random hop is added it is required to check for possible loops that were created. The minimum number of random hops that needs to be added is set to two, this should increase the average hop count of a transaction by at least two, ensuring that every transaction will be randomized. An edge case of a transaction with only one hop will have a total of three hops, making it more difficult to identify small anonymity sets.

Algorithm Pseudocode

Next, we will describe the pseudocode which can be found in algorithm 1, illustrating how the modification can be implemented in a LN implementation after having computed a $path$ with a given graph G. In the case of a direct channel the algorithm will not apply any changes returning the same path, the identity of the sender or receiver cannot be revealed to any possible adversary. To add an additional level of randomness, the number of possible hops that can be added is chosen to randomly be between 2 and the maximum hops in the transaction, line 5. The loop in line 7 goes through each node in the path, computing the hops that can be added, and returning the modified path when the last one is reached. For each node, we calculate the hops connected to it in the graph G and at line 13 we check whether all conditions to be a hop are met, first checking that it is not equal to the next hop on the path. The other conditions are checked through functions that are assumed to be implemented, we will not detail their exact implementation as depending on the data structures used to store the LN graph they will have different implementations. The function $hasEdge$ will return true only if in graph G there exists a direct channel connecting $hop \rightarrow nextNode$. This means the hop can be inserted on the path because it connects to the next node present in the original path. The function $noLoops$ will return true only in the case that adding the hop will not create any loop on the modified path and the provided one. Finally, the function $hasEnoughCapacity$ makes sure the two new channels that will be used have enough capacity to forward the transaction. For simplicity, the exact fee calculation is excluded from the pseudocode, but in a realistic implementation it should be calculated to correctly check the capacity of the channels. To implement the fee calculation it is advised to reconstruct the path backwards using the original amount as a starting point, this will make sure the recipient will get the same amount and decrease the possibility of failed payments. If there are still hops to be added and there are multiple possible hops which can be used, the algorithm will choose between them at random, the function $random$ at line 17 is assumed to take an array of values and randomly return one of them. In some edge cases of the network, it might not be possible to add any random hop causing a transaction to use the shortest path that was found. In this case, we advise to compute it on a suboptimal path for which at least 1 random hop is added. Given that the timestamps of a transaction can be known, the randomness used when calculating the choices should not be predictable, therefore a cryptographically secure random algorithm should be used [25].

Runtime Complexity

The worst case algorithm complexity is $O(V * E)$ for a path that spans the entire network of V nodes where each node has E connections that need to be checked. The if condition at line 13 should be amortized $O(1)$ considering the graph is stored to easily retrieve edges between nodes and hash tables are used to check for loops. Moreover in a realistic implementation, the path length should grow at most $O(logV)$ [26] and each node will have a constant number of channels connected [27]. Therefore it can be concluded that the algorithm complexity is a lower bound with respect to Dijkstra algorithm making it a viable option to adopt.

3.3 Attacking the Modification

If the modification is used when calculating the path to route a payment, the attack to the LN anonymity will not be as successful due to the fact that it will discard paths with more hops than the optimal one. If an adversary is not aware of it, the anonymity will definitely increase as it will not be able to realize that the paths that are being used are not the shortest ones. However, we will design a new attack considering an adversary that is aware of the complete implementation. The objective of the attack will be to reduce the number of false positives while keeping the computation complexity feasible in terms of runtime. A false positive is defined as having a singular anonymity set which does not contain the actual sender or receiver of the transaction.

Phase I of the attack described in the previous section will not be changed as the search for possible destinations works the same way using the remaining time lock value. Although knowing that the path length increases by at least two hops, an attacker should try to raise the depth limit if computation power permits. Due to limited computation power, the depth limit of the attack will remain unchanged. Phase II instead has to be changed to account for hops that might have been added by the modification and should not discard pairs early in the search. We will detail two different strategies to attack a transaction during Phase II and evaluate their results later on.

The first strategy uses the same search for sources as in the original attack, the condition for discarding a pair is slightly modified. The pseudocode for a function checking whether a suboptimal path might be generated by the modification can be found in algorithm 2. Whenever an optimal path containing the adversary is found, the first condition that needs to be met is that the optimal path should not contain more nodes than the suboptimal one, this is achieved by subtracting the size of the sets in line 2. If the first condition is met, a walk is started from the first node in both paths. At each point, the same hop should be found for both paths, if this is not the case then a random hop might have been added in that spot. To test whether a random hop was added, the algorithm compares the next hop in the suboptimal path to the current optimal path hop. If they are equal, that means a random hop was added, otherwise the function returns discarding this subpath. In the original attack, a singular source set containing the previous node was returned whenever a more optimal path was found by the adversary. This is because a suboptimal path is used only if the sender does not have enough forward balance to use the best path, due to the nature of randomly choosing hops this assumption was removed from the new attack.

The second strategy instead goes through all nodes in the LN and checks whether they can be a source for all possible des-

Algorithm 1 Adding random hops to a given path

```
1:  function ADD RANDOM HOPS(G, path, amount)
2:      if $length(path) \leq 2$ then return $path$
3:      $modifiedPath \leftarrow \{\}$
4:      $len \leftarrow length(path)$
5:      $hopsToAdd \leftarrow randomBetween(2, len - 1)$
6:      $addedHops \leftarrow 0$
7:      for all $node \in path$ do
8:          $nextNode \leftarrow next(path, node)$
9:          if $nextNode = \emptyset$ then
10:             $modifiedPath \leftarrow modifiedPath + node$ return $modifiedPath$
11:         $possibleHops \leftarrow \{\}$
12:         for all $hop \in edgesOut(G, node)$ do
13:             if $hop \neq nextNode \wedge hasEdge(G, hop, nextNode) \wedge noLoops(modifiedPath, path, hop) \wedge$
                    $hasEnoughCapacity(G, node, hop, nextNode, amount)$ then
14:                 $possibleHops \leftarrow possibleHops + hop$
15:         $modifiedPath \leftarrow modifiedPath + node$
16:         if $addedHops < hopsToAdd \wedge length(possibleHops) > 0$ then
17:             $modifiedPath \leftarrow modifiedPath + random(possibleHops)$
18:             $addedHops \leftarrow addedHops + 1$
```

tinations that were found in Phase I. For each sender and recipient pair, the path finding algorithm is executed, if the attacker appears on the found path it is added to the anonymity set. If the attacker is not found, the algorithm checks for each connection the nodes on the path have, and if the adversary is connected, it is added to the anonymity set. Checking all nodes for every destination is a slower strategy, but it should reduce the number of false positives to a minimum. If the search during Phase I has completed, the anonymity set should contain both pairs in 100% of the cases. The next best possible attack available would be to compare all destinations against all sources, bruteforcing all possible paths that are available. Due to its runtime complexity the exhaustive search will not be analysed. We would like to note that for adversaries with high centrality, constructing paths from all nodes in the network will result in adding the adversary to many of them due to their strategic position.

Attack Complexity

The conclusion of [12] regarding the attack complexity of the first phase remains unaltered. The attack goes through all possible hops after the adversary that could be a destination up to a max depth of four. Therefore, the runtime complexity of Phase I will still be $O((Deg_{max})^d)$ where Deg_{max} is the maximum degree of the LN and d is the depth after which the search is stopped.
In the second phase of the first strategy, the attack goes through all possible destinations and computes a set of sources, the worst case scenario for this is when all nodes have to be checked with $O(V)$ complexity. For each destination a generalized version of Dijkstra algorithm which does not stop at the best path is used. The algorithm is going to stop only after having analysed enough suboptimal paths that could be generated from the modification. The modification limits the maximum number of nodes to be added therefore the value of suboptimal paths to be considered will be at most constant k. Yen's algorithm to find the top k subpaths, can be seen as a generalization of adding random hops along the path. For this reason, the runtime complexity of Yen's algorithm will be used, which is $O(kV(E+VlogV))$ [28]. Combining everything together the overall runtime complexity of the first strategy is $O((Deg_{max})^d kV^2(E+VlogV))$.
The second strategy worst case scenario would be to run Dijkstra algorithm with both source and destination set corresponding to the entire network. The runtime complexity is $O(V^2)$ for a search spanning the entire network and $O((E+V)logV)$ for Dijkstra algorithm implemented using a binary heap. Checking the path length for the adversary will have a worst case complexity of $O(logV)$ [26] which is a lower bound with respect to Dijkstra algorithm. Hence, the overall runtime complexity for the second strategy is $O((Deg_{max})^2 V^2(E+V)logV)$.
While the first strategy has worse runtime complexity, in the average scenario, the number of sources that will be analysed will not be in the order of $O(V)$. On the other hand, the second strategy forces the attack to go through all existing nodes each time impacting the runtime of every attack.

Algorithm 2 Checking a subpath contains random hops

```
1:  function IS MODIFIED PATH(G, subpath, optimal)
2:      if $len(optimal - subpath) > 0$ then return $False$
3:      $i \leftarrow 0$      $j \leftarrow 0$
4:      while $i < len(optimal) \wedge j < len(subpath)$ do
5:          if $optimal[i] \neq subpath[j]$ then
6:              $j \leftarrow j + 1$
7:              if $j \geq len(subpath) \vee$
                    $optimal[i] \neq subpath[j]$ then
8:                  return $False$
9:          $i \leftarrow i + 1$      $j \leftarrow j + 1$
        return $j >= len(subpath)$
```

4 Evaluation Metrics and Simulation Results

In this section we will first describe the evaluation criteria used to asses the impact on the anonymity and efficiency of the modification. Following we will illustrate the dataset used for the simulation and explain the results obtained after simulating it.

4.1 Anonymity and Efficiency Criteria

The purpose of the evaluation is to analyse whether it is possible to improve the anonymity of the LN and measure the cost of improving it. For this reason, we selected anonymity metrics to measure the degree to which users can be de-anonymized in the network. On the other hand, the efficiency metrics focus on the usability and scalability aspects of the LN.

Anonymity metrics

- Size of source and destination anonymity sets.
- The ratio R_{att} between the transactions attacked and the total number of transactions.
- The average ratio Av_{att} of transactions that were attacked by multiple adversaries.
- The correlation between the size of source or destination anonymity sets with their distance, respectively $CorrD_S$ and $CorrD_R$.
- The percentage anonymity sets with a correct singular source $Sing_S$, destination $Sing_R$ or both $Sing_{Both}$.
- The percentage of the attacks that completed phase I of the attack $CompI$.
- Percentage of transactions for which the recipient and the sender were included in the anonymity set thus de-anonymizing both parties $Success_{att}$.
- The percentage of false positives with regards to source $FalsePos_S$, destination $FalsePos_R$ or both $FalsePos_{Both}$, meaning a singular set was found but it contained the wrong node.

Efficiency metrics

- The runtime complexity of the routing algorithm with the modification.
- The percentage of failed transactions $Fail$.
- The average number of hops Av_{hop} excluding sender and receiver as they are not considered hops.
- The average fee Av_{fee} paid by the sender of the transaction.
- The average fee normalized by the transaction amount N_{fee}.

4.2 Dataset

The snapshot of the LN is taken from lnchannels [29] which contains a list of all publicly broadcast information about live nodes and channels. When populating the graph for the simulation, channels with no capacity are excluded as well as the ones that did not broadcast a policy containing their fee rate and time locks. The number of nodes being tested are 4791 having 28997 channels, transactions between them are randomly initiated with amounts ranging between 1 and 100000 satoshis. The amounts are uniformly distributed for each order of magnitude. Due to limited computation power, the snapshot of the LN is used to only evaluate the efficiency of the modification. The evaluation of the anonymity aspect is done through smaller constructed graphs using Barabasi & Albert construction (BA) [30] and Erdos & Renyi (ER) construction [31]. Using BA method two graphs are constructed, one with 100 nodes and 2 edges each (BA_100), and the other one with 500 nodes and 5 edges each (BA_500). ER method is used to construct one graph with 500 nodes and a probability of edge creation of 0.02 (ER_500). For each graph, the adversaries are chosen to be the top ten nodes with the highest betweenness centrality metric [32]. The channels are populated with random values for their fee calculation, time lock, balance, and age. 5000 transactions are executed to obtain the efficiency metrics and 1000 for the anonymity ones.

4.3 Results

Table 1 displays the efficiency metrics previously defined, the two columns respectively represent the results obtained before and after the modification.

Parameter	Before Modification	After Modification
$Fail$	0.08	0.1
Av_{hop}	2.41	4.57
Av_{fee}	2.85	13.78
N_{fee}	0.09	0.43

Table 1: Efficiency metrics for the LN snapshot

The ratio of failed transactions increased from 8% to 10% mostly due to the increase in hop counts. Increasing the number of nodes involved in a transaction raises the probability of one channel not having enough balance to forward it or a node going offline. In line with the minimum number of hops that should be added, the average hop count increased by 2.16. This means the average transaction is able to add two hops, but not necessarily all transactions can, especially in less connected parts of the LN. The average fee increased by almost five times from 2.85 to 13.78. To consider low-value transactions, we normalize the fee calculation to the transaction amount, this still shows an increase in fee of 4.77 times.

Table 2 and 3 display the anonymity metrics, respectively, for the first and second strategy. Each column represent a different graph structure and the rows correspond to one of the metrics defined before.

The first strategy shows a success rate between 29% and 50% for the different graph structures. With the increasing number of nodes, the completion of Phase I decreases from 8% in BA_100 to 6% in BA_500, consequently the success rate drops with an increase in destination false positives. The ER method constructs a more sparse graph for which the completion of Phase I is 23% but the number of attacked transactions is much lower at 17%. In all simulations, the number of destination false positives is higher than 4% therefore the algorithm to discard pairs early in the search seems

Parameter	BA_100	BA_500	ER_500
R_{att}	0.9	0.73	0.17
Av_{att}	2.38	1.75	1.09
$CorrD_S$	0.4	0.46	0.35
$CorrD_R$	0.36	0.3	0.34
$Sing_S$	0.01	0.0	0.0
$Sing_R$	0.42	0.19	0.28
$Sing_{Both}$	0.0	0.0	0.0
$Comp_I$	0.08	0.06	0.23
$Success_{att}$	0.36	0.29	0.5
$FalsePos_S$	0.01	0.0	0.0
$FalsePos_R$	0.06	0.08	0.04
$FalsePos_{Both}$	0.0	0.0	0.0

Table 2: Anonymity metrics for the first attack strategy with three different graph structures (BA_100, BA_500, ER_500)

Parameter	BA_100	BA_500	ER_500
R_{att}	0.88	0.76	0.21
Av_{att}	2.2	1.76	1.07
$CorrD_S$	-0.02	-0.29	0.05
$CorrD_R$	0.55	0.49	0.65
$Sing_S$	0.01	0.0	0.02
$Sing_R$	0.33	0.16	0.24
$Sing_{Both}$	0.01	0.0	0.01
$Comp_I$	0.12	0.09	0.31
$Success_{att}$	0.44	0.43	0.53
$FalsePos_S$	0.02	0.0	0.01
$FalsePos_R$	0.0	0.0	0.0
$FalsePos_{Both}$	0.0	0.0	0.0

Table 3: Anonymity metrics for the second attack strategy with three different graph structures (BA_100, BA_500, ER_500)

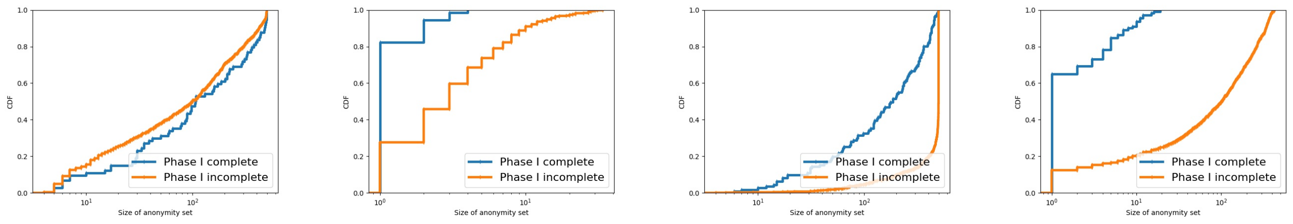
to discard valid destinations in multiple occasions.
The second strategy reduces the destination false positives by analysing all possible sources and therefore avoiding the chance of returning early with a smaller anonymity set. The destination false positives for all graph structures is in fact 0%, although in some cases when Phase I does not complete, a false positive source might be added. For all graph structures, the success of an attack increased with respect to the first strategy. In 1% of the cases for the BA_100 and ER_500 graphs, it is possible to singularly de-anomymize both the sender and the receiver of the transaction. Singular anonymity sets for sources were found at best in 2% of the attacks in the case of ER_500. On the other hand, singular destinations were found in at least 16% of the attacks making them more exposed to de-anonymization.
The randomization involved when generating transactions can be seen in the percentage difference of the transactions being attacked between the first and the second run. The second attack strategy seems to be more successful, although it should be noted that the approach is much slower than the first one if many transactions have to be attacked. In the average scenario the first strategy will not consider the entire network as a possible source set, which makes the first strategy more affordable in case of limited computation power.

We also note that the average number of attacks per transaction is bigger than one, thus some transaction paths were long enough to go through multiple adversaries. In this case, a set intersection could be used to reduce the size of anonymity set even more if anonymous multi-hop locks are not used [23].

In Fig. 1 we plot the size of the anonymity sets as a cumulative distribution function for both source and destination sets. The plots only show the run on the BA_500 graph for both the first and second strategy, false positives are excluded. The blue line corresponds to attacks for which Phase I has completed, this happened 6% and 9% of the time for the first and second runs. In both cases a completion of phase I produces an anonymity set size of less than 10 for the destinations. Failing to complete Phase I shows the first strategy returning smaller anonymity sets while the second one, considering all nodes, creates much larger anonymity sets. On the other hand, the sender anonymity sets appear to be smaller than 10 only in less than 20% of the cases for the first strategy, indicating a similar growth whether Phase I has completed or not. Completion of Phase I is instead important to create small anonymity sets in the second strategy.
When considering the entire network as a possible source, the strategic positioning of adversaries can be seen by the larger anonymity sets. The high centrality of each adversary causes them to be included in many of the transactions, ultimately creating large sets. It is important to note that a correct sender and recipient pair was found in 29% and 43% of the cases, respectively, for the first and second strategy. This means, the size of the anonymity set is not necessarily indicating the degree of the user anonymity because in more than 50% of the cases they were not included, making the set meaningless to the anonymous pairs.

5 Discussion

The results obtained are promising towards a successful introduction of randomness into routing algorithms used by the LN, although additional consideration regarding the increased delay and cost should be made as well as the complexity of the implementation. We are going to discuss what problems might arise when adopting the described methodology to calculate the payment path as well as alternatives that can be used instead.
In a live implementation of the LN, there might be the need for some users to transact relatively low amounts that do not require a high degree of anonymity and would prefer to pay less in transaction fees. By design, this is possible as the random hops are added only after having computed a short path. Due to the increased number of hops that are required to move the funds, higher chances of slow payments arise, especially in the case of adversaries that might want to dispute them causing an increase in layer-one transactions. In particular the chances of a node going offline or not responding for a long period of time could slow down the fast payments the LN promises. The impact of larger fees and the higher chances of having to resolve disputes were not discussed in depth and are left for future research developments.



(a) BA_500 Sources First Strategy

(b) BA_500 Destinations First Strategy

(c) BA_500 Sources Second Strategy

(d) BA_500 Destinations Second Strategy

Figure 1: Size of anonymity sets for the first and second attack strategy simulated on the BA_500 graph structure

The strategy described to add random hops focuses on randomization rather than optimization. This has the direct consequence of increasing transaction fees and raising the chances of payment failure. The possibility for a malicious party to randomly place expensive channels along the network should be considered. A strategy that could be adopted would be to choose only hops with lower transaction fees. This would most likely decrease the average fee, but it could open up for attacks where the adversary will be able to reduce the number of possible added hops due to the determinism involved when choosing them.

Although random hops seem to be a feasible way to increase the anonymity while not modifying complexity of implementation, there are other possibilities that can be used. As an example of a more complex implementation inspired by the dovetail protocol [33] would be to use multi path segment routing [34]. A node called dovetail would be randomly chosen and the transaction is routed from sender to the dovetail and then to the receiver. This creates two different shortest paths for which an adversary who happens to be on the path is not able to distinguish between increasing the anonymity of source and destination. Similar to adding random hops, it could be possible to insert more than one hop as in the case of partial route computation [35] or length-bounded random walks [36]. The disadvantage when inserting multiple hops is that the fees might increase by larger amounts and the possibility of failed transactions too. A more general approach, implemented in part by Eclair, is to use k sub-optimal paths which can be considered to be a simple although efficient solution [37]. Based on the chosen k the algorithm will probably find one of the paths that more specific random hops implementation might generate. Hence, making it more difficult to design a specific attack for an adversary aware of the implementation.

6 Conclusions and Future Work

In this work, we describe attacks developed by recent research [12] that enable adversaries to de-anonymize the parties involved in a transaction, if they happen to be on the payment path. We show how a change to the randomness involved in the path finding algorithm could increase the anonymity. The modification involves taking a computed short path and then add random hops reconstructing a new one.

Two attacks are designed to account for the modification considering suboptimal paths that might have been generated after adding the hops. The first strategy focuses on determining whether suboptimal paths can be generated by the modification, while the second one tries to exhaustively search for all possible sources that can match a destination.

Estimating the anonymity degree of large sets is difficult when the probability of them containing the correct pairs is low. For this reason further research should be done on analysing the possible ways to estimate the anonymity of the users involved. As an example, Shannon entropy [38] could be used, although additional consideration should be made on how to estimate the probabilities in a complex graph structure such as the LN.

The performed simulation did not take into consideration the possibility for adversaries to communicate and create intersections of their anonymity sets in the case a transaction was attacked multiple times. This happens more often due to the increased hop counts. The assumption was that the cryptographic properties of HTLC will be improved in the future [23], removing the chance that this attack would be feasible. It is still a possibility and ground for further research on the topic. Simulating multiple concurrent transactions should be tested with the modification to assess the impact of an increase in hop counts on the number of failed payments and their delays. We considered mainly anonymity metrics, but the efficiency of the LN should be taken into consideration if there is a chance that scalability might be impacted because of the added random hops.

The increase in transaction fees is noticeable especially for users sending really small amounts such as 1 satoshi. To remedy this increase a trade-off could be made between the chances of getting de-anonymized and the fees that are paid by the user. As an example if a 1 satoshi transaction is already composed of 5 hops, adding more random hops would only increase the fee paid by the sender who probably is not concerned about the anonymity of the transaction he sent. A solution would be to have an automatic optimization that adds random hops based on an anonymity parameter that can be chosen by the user. This makes it possible to have higher security levels paying a higher price. An adversary will not be aware whether random hops were added thus he will have to create an attack in which the user might or might not have added random hops into the routing path.

7 Responsible Research

This work is intended to provide an overview of the current limits on the anonymity of the Lightning Network and illustrate how a possible modification to its specification would make an improvement possible. All attacks were executed inside an offline simulation thus they never had a real impact on the anonymity of any person that is using the main network. All the assumptions made were taken from a realistic snapshot of the LN from which a randomly generated simulation was designed. The data being used is therefore a recent representation of the live version of the network but the simulated transactions were never actually executed on it. The list of sent payments is not public information, it would not be possible to use it as a base dataset, attempting to obtain a list of real transactions using the attacks specified above is neither a feasible solution. All the simulations were started by randomly picking nodes in the network and selecting an amount that should be processed without consideration to possible connections the nodes might have. We conclude that the anonymity of any past or current user of the LN who might have been included in the downloaded snapshot was never affected. Moreover, it should not be affected in the future if steps are taken as previously suggested to reduce the chances of de-anonymization by possible adversaries.

Reproducing the attack using a similar dataset should not produce completely different results than the one obtained. Nevertheless the randomization involved when creating the graph and generating transactions can impact the end results as can be seen in the difference between the two analysed runs. The modification of the algorithm can be reproduced by implementing the pseudocode in algorithm 1 and modifying the initial attack to account for the added randomness in the routing selection. Hence it should be possible to carry out the same experiment on any up to date version of the LN and confirm its anonymity impact. We would like to note that simulating the live snapshot of the LN requires to allocate enough computational power to obtain the results in an appropriate time. A repository containing the code, the dataset, and the results obtained from the simulation is also available at https://github.com/paolokazemi/Lightning-Network-Anonymity.

The environmental impact of Proof-of-Work blockchains should also be taken into consideration. The LN has a lower impact due to the fact that most of the transactions should be processed off-chain, except for the opening or closing of a channel and handling disputes. The modification explained in this work increases the average hop count that a transaction needs to take to reach its destination. This has the consequence that a potential longer route could have problems with nodes taking more time to unlock the funds and higher chances of payment failures. In these cases, a user might decide to execute the transaction on the layer-one blockchain instead, increasing its load. However, we argue that this would happen only if most of the hops involved in a transaction decided to dispute it which would have the same consequences on the existing version of the LN. In fact, the base principle of the LN is to connect nodes by means of contracts that can be validated using the Proof-of-Work of the blockchain, but this should be done only in specific circumstances.

References

[1] R. Houben and A. Snyers, "Cryptocurrencies and blockchain," *Bruxelles: European Parliament*, 2018.

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system." https://bitcoin.org/bitcoin.pdf, 2008. [Online; Accessed: Jun. 27, 2021].

[3] G. Di Stasi, S. Avallone, R. Canonico, and G. Ventre, "Routing payments on the lightning network," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 1161–1170, IEEE, 2018.

[4] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais, "Sok: Layer-two blockchain protocols," in *International Conference on Financial Cryptography and Data Security*, pp. 201–226, Springer, 2020.

[5] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments." https://lightning.network/lightning-network-paper.pdf, 2016. [Online; Accessed: Jun. 27, 2021].

[6] M. Araoz, R. X. Charles, and M. A. Garcia, "Structure for deterministic p2sh multisignature wallets." https://github.com/bitcoin/bips/blob/master/bip-0045.mediawiki, 2014. [Online; Accessed: Jun. 27, 2021].

[7] "BOLT #5: Recommendations for On-chain Transaction Handling." https://github.com/lightningnetwork/lightning-rfc/blob/master/05-onchain.md, 2016. [Online; Accessed: Jun. 27, 2021].

[8] "BOLT #7: P2P Node and Channel Discovery." https://github.com/lightningnetwork/lightning-rfc/blob/master/07-routing-gossip.md, 2016. [Online; Accessed: Jun. 27, 2021].

[9] "BOLT #4: Onion Routing Protocol." https://github.com/lightningnetwork/lightning-rfc/blob/master/04-onion-routing.md, 2016. [Online; Accessed: Jun. 27, 2021].

[10] B. Vu, "Exploring Lightning Network Routing." https://blog.lightning.engineering/posts/2018/05/30/routing.html, 2018. [Online; Accessed: Jun. 27, 2021].

[11] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[12] S. P. Kumble, D. Epema, and S. Roos, "How lightning's routing diminishes its anonymity," in *Proceedings of the 16th International Conference on Availability, Reliability and Security*, pp. 1–10, 2021.

[13] "Lightning Network Daemon." https://github.com/lightningnetwork/lnd, 2021. [Online; Accessed: Jun. 27, 2021].

[14] "c-lightning: A specification compliant Lightning Network implementation in C." https://github.com/ElementsProject/lightning, 2021. [Online; Accessed: Jun. 27, 2021].

[15] "Eclair." https://github.com/ACINQ/eclair, 2021. [Online; Accessed: Jun. 27, 2021].

[16] J. Y. Yen, "An algorithm for finding shortest routes from all source nodes to a given destination in general networks," *Quarterly of Applied Mathematics*, vol. 27, no. 4, pp. 526–530, 1970.

[17] G. Kappos, H. Yousaf, A. Piotrowska, S. Kanjalkar, S. Delgado-Segura, A. Miller, and S. Meiklejohn, "An empirical analysis of privacy in the lightning network," *arXiv preprint arXiv:2003.12470*, 2020.

[18] "Hash Time Locked Contracts." https://bitcoin.it/wiki/Hash_Time_Locked_Contracts, 2019. [Online; Accessed: Jun. 27, 2021].

[19] H. Handschuh, *SHA Family (Secure Hash Algorithm)*, pp. 565–567. Boston, MA: Springer US, 2005.

[20] "BOLT #3: Bitcoin Transaction and Script Formats." https://github.com/lightningnetwork/lightning-rfc/blob/master/03-transactions.md, 2016. [Online; Accessed: Jun. 27, 2021].

[21] E. Rohrer and F. Tschorsch, "Counting down thunder: Timing attacks on privacy in payment channel networks," 2020.

[22] S. Tochner, S. Schmid, and A. Zohar, "Hijacking routes in payment channel networks: A predictability tradeoff," *arXiv preprint arXiv:1909.06890*, 2019.

[23] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei, "Anonymous multi-hop locks for blockchain scalability and interoperability," in *26th Annual Network and Distributed System Security Symposium, NDSS 2019*, 2019.

[24] I. A. Seres, L. Gulyás, D. A. Nagy, and P. Burcsi, "Topological analysis of bitcoin's lightning network," in *Mathematical Research for Blockchain Economy* (P. Pardalos, I. Kotsireas, Y. Guo, and W. Knottenbelt, eds.), (Cham), p. 4, Springer International Publishing, 2020.

[25] J. Kaltz and Y. Lindell, "Introduction to modern cryptography: principles and protocols," 2008.

[26] D. Coppersmith, D. Gamarnik, and M. Sviridenko, "The diameter of a long-range percolation graph," in *Mathematics and computer science II*, pp. 147–159, Springer, 2002.

[27] "Lightning Network Statistics — 1ML - Lightning Network Search and Analysis Engine - Bitcoin mainnet." https://1ml.com/statistics, 2021. [Online; Accessed: Jun. 27, 2021].

[28] E. Bouillet, G. Ellinas, J.-F. Labourdette, and R. Ramamurthy, *Path routing in mesh optical networks*, pp. 128–129. Wiley Online Library, 2007.

[29] "lnchannels." https://ln.fiatjaf.com/, 2021. [Online; Accessed: Jun. 27, 2021].

[30] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[31] P. Erdős and A. Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, vol. 5, no. 1, pp. 17–60, 1960.

[32] U. Brandes, "A faster algorithm for betweenness centrality," *Journal of mathematical sociology*, vol. 25, no. 2, pp. 163–177, 2001.

[33] J. Sankey and M. Wright, "Dovetail: Stronger anonymity in next-generation internet routing," 2014.

[34] J. Heemskerk, S. P. Kumble, and S. Roos, "Improving anonymity of the lightning network using multiple path segment routing," 2021.

[35] R. de Boer, S. P. Kumble, and S. Roos, "Improving blockchain anonymity using hop changes with partial route computation," 2021.

[36] M. E. Ozkan, S. P. Kumble, and S. Roos, "Improving the anonymity of blockchains: The case of payment channel networks with length-bounded random walk insertion," 2021.

[37] M. Plotean, S. Roos, and S. P. Kumble, "Improving the anonymity of the lightning network using sub-optimal routes," 2021.

[38] A. Serjantov and G. Danezis, "Towards an information theoretic metric for anonymity," in *International Workshop on Privacy Enhancing Technologies*, pp. 41–53, Springer, 2002.