Improving the Anonymity of Blockchains: The Case of Payment Channel Networks with Length-bounded Random Walk Insertion

Mehmet Emre Ozkan , Satwik Prabhu Kumble , Stefanie Roos

June 27 2021

Abstract

The LND is currently the most popular routing algorithm used in the Lightning Network, the second layer solution to Bitcoin's scalability. Despite its popularity, recent studies demonstrate that its deterministic nature compromises the anonymity of the Lightning Network. In other words, threatening parties present in the transaction path can guess the sending and receiving parties of transactions easier than in the absence of such strong determinism. As a solution, we propose augmenting the LND with a length bounded random walk insertion to include randomness into the transaction path and regain anonymity. Most importantly, we found that for generated network simulations and the snapshot network, including the random walk into the transaction path improves anonymity. In simulations with LND routing, attackers could identify senders or receivers for 70% of transactions. For simulations of networks with 100 nodes and an average of 2 edges per node with the weighted random walk insertion, attackers could identify senders or receivers around 65% of the time. However, for simulations of networks with 500 nodes and an average of 10 edges per node with the weighted random walk insertion, attackers could never identify senders or receivers. Besides, for the snapshot simulation, they could only identify either in around 4% of transactions. Thus, we overall believe that the random walk insertion into the LND algorithm addresses the anonymity issue of the unmodified algorithm.

1 Introduction

Lightning Network is a second-layer solution to bitcoin's scalability problem, which is a payment channel network. Recent studies discovered that the Lightning Network is not entirely anonymous, meaning attackers could reveal the senders' or receivers' identities. As anonymity is important, this research will try to improve the anonymity of the Lightning Network with different routing algorithms. First of all, to understand the problem, the Lightning Network will be introduced. Afterward, some of the reasons for this anonymity loss will be explained. Lastly, random walk algorithms will be introduced, which will be used in our method.

1.1 Lightning Network

The Lightning Network is a second-layer solution to Bitcoin's scaling problem. Bitcoin's bottleneck is mainly the number of transactions that can be stored on the blockchain at any given moment. The Lightning Network, a payment channel network, tackles this issue by eliminating the requirement for each transaction to be stored on the blockchain.
In the Lightning Network, every transaction passes through the payment channel. The channel is created by storing the opening transaction on the blockchain, and each node (user) can open a channel to another node. Each channel has fee-related information, the total balance, and timelock value.

The blockchain is only used when creating and removing the channels. When a user wants to send a payment, the user will compute the path of the transaction and send the payment using the onion routing protocol through channels [2]. Onion routing protocol hides the route of the transaction from the adversaries nodes by encapsulating the route[12]. Each user can compute its payment route but in the Lightning Network around 92% of the network uses LND routing [1]. Lightning transactions are signed with a new sighash type that allows transfers to occur between untrusted parties. In the event of uncooperative or hostile participants, lightning transactions are enforceable via broadcast over the bitcoin blockchain, through a series of decrementing timelocks [2].

1.2 Anonymity Issue

One of the weak points of the Lightning Network is low anonymity, because of the popular routing algorithms. The most popular routing algorithms in use are focused on reliability and efficiency, because of that they are mostly deterministic [4]. Since the routing is mostly deterministic, an attacking node, a node part of the route may identify the sender or the receiver even though onion routing is being used [1]. This discovery of the path is possible because of two main pieces of information being available. One of them is publicly available timeclocks, which give information about how much time the transaction has left. The other one is the routing algorithm, attacking nodes do not know which routing algorithm is used in the transaction. However, since around 92% of the users use LND routing, the attacking node can guess many of the senders or receivers [1].
Instead of using those highly deterministic routing algorithms, a routing algorithm with undetermined characteristics such as random walk insertion can be promising to increase the anonymity in the network.

1.3 Random Walk Algorithms

A random walk describes a path consisting of a succession of random steps in the mathematical space [6]. Random Walk algorithms are useful algorithms that are used in many scientific fields [6]. There are 2 main types of Random Walk algorithms. One of them is completely random walk, where each possible step has an equal probability of being chosen, only depending on the random number generator. Another way of using random walks is by changing the probabilities of steps. For example, an ant simulation defines a step as a choice where the ant should go, if a step is chosen by many ants, the coming ants should be more likely to choose that step too [7], which is a random walk algorithm with weights, prioritizing the already visited nodes.
So in the case of the Lightning Network, a random walk algorithm would create the path to the receiver by adding random hops to the path until it finds the receiver node. The big problem with a completely random walk routing algorithm is the long routes. Unnesecerray long payment paths increase the fee's for a transaction and increases the network load which makes denial of service attacks easier [5]. Because of these reasons, a length bounded random walk insertion algorithm will be used when creating the transaction route, so just adding a small random walk in between the efficient routes.

In the next section, a little about the related work will be mentioned. Afterward, in methodology, the routing algorithms implemented will be explained, along with the simulation and the attack on the new routing algorithm. And in the next section, the routing algorithm will be evaluated. After the evaluation, we will talk about the conclusion.

2 Related Work

There is not a lot of research into the anonymity of the Lightning Network. From recent researches, we learned that Lightning Network anonymity can be compromised [1]. There are some known ways to

increase the anonymity with shadow routing [8] since it hides the timelock information. There is also research on the cost of attacking the Lightning Network [10], which explains it would cost around 3 million dollars to have 2 percent of the nodes in the network. There is also some research on increasing the efficiency and the anonymity of the network with different types of routing algorithms [11]. Unfortunately, there was no research into analyzing the anonymity of routing algorithms that use random walk insertion.

3 Methodology

3.1 Random Walk Insertion Algorithms

This paper considers three different random walk algorithms. All algorithms have a different way of choosing the route. The first one we will talk about prioritizes efficiency over anonymity, which is called Random Hop Insertion. The second algorithm (Random Walk Insertion) creates routes that are much more randomized that improves anonymity but decreases efficiency. And the third algorithm increases the anonymity even more than the second algorithm, which is called Weighted Random Walk Insertion. And all of these algorithms are length bounded but in different ways.

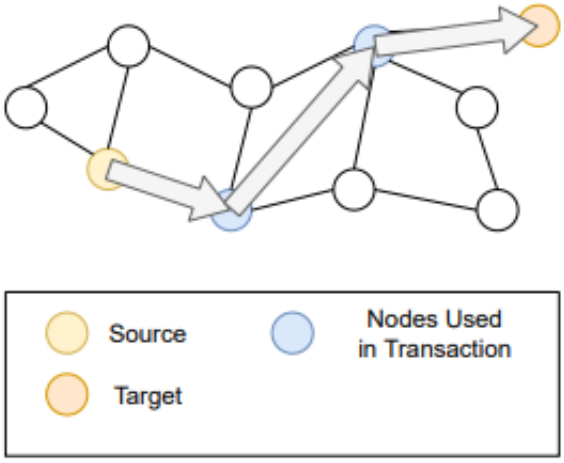

Figure 1: Example of Optimum Path

3.2 Algorithm 1: Random Hop Insertion (RHI)

Random Hop Insertion algorithm as the name suggests, adds random hops to the most optimum route. The sender chooses the length n which is the number of channels used in the route. One by one n many random hops will be inserted into the path as long as the length of the complete path is smaller than the input length n. Therefore, if the length given is already smaller than the most optimum route, there will not be any random hops added. The most optimum route is found by using the LND routing algorithm. This random hop is inserted from a node that is randomly chosen in the optimum route, which is called a random source. Then the optimum route is computed from the randomly hoped node to the receiver. This random hop insertion is repeated until enough hops are added n times or the path length is equal to n. Each time before choosing a random source, it is not allowed to choose any of the nodes in the path before the last random source. This restriction gets rid of the possibility of removing the randomly added hops.

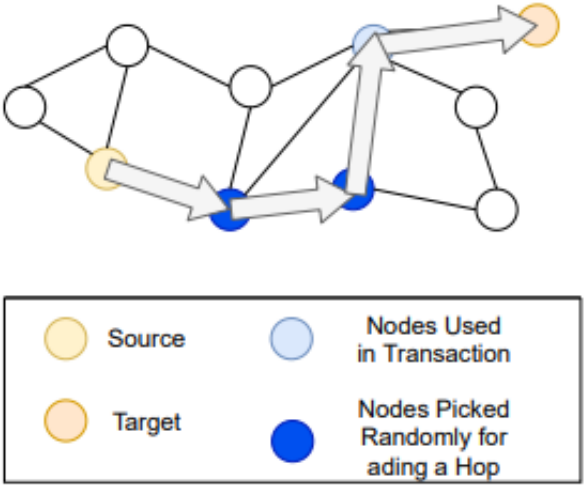

Figure 2: Example of Algorithm 1, length 5

3.3 Algorithm 2: Random Walk Insertion (RWI)

Random Walk Insertion algorithm was developed after implementing and testing the Random Hop Insertion algorithm. The routing algorithm is simple as the name suggests, and the sender chooses the length n of the random walk. First, it will get the optimum route with the LND routing algorithm. Then a random node in the path is chosen except the source and the sink. From the chosen node, a random walk is created with the given length n. Once the length of the random walk finishes, the optimum route is taken to the target. As the name suggests, it inserts a random walk from a random node with a given length, and when the random walk is finished, the optimum route to the target is taken.

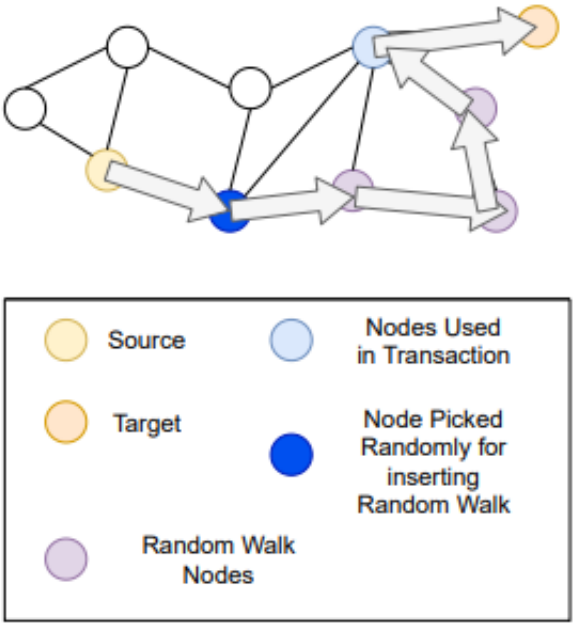

Figure 3: Example of Algorithm 2/3, length 3

3.4 Algorithm 3: Weighted Random Walk Insertion (WRWI)

Weighted Random Walk Insertion algorithm is just like Random Walk Insertion algorithm, the only difference being the random walk is weighted. This algorithm works as follows. First, it will get the optimum route from source to sink with LND. Then a random node in the optimum path is chosen except the source and the sink. However, this random choice is not completely random, it is weighted with the degree of the nodes. The degree of nodes is the number of connections a node has, implying that the more connections a node has, the more likely it is to be chosen. This adds more randomness to the path as higher degree nodes have more nodes to hop. From the randomly chosen node, a degree weighted random walk with the given length n is inserted. After inserting the random walk, the optimum route to the sink is taken with LND routing.

All of these algorithms have different ways of adding randomness to the route. While the length given in the first algorithm bounds the length of the complete path, others only use the length in the random walk. However, all of them are optimized with LND routing, so they might show the same weaknesses. And none of the algorithms will add any protection to the receiver's identity if the attacking node is after the random insertions.

3.5 Simulation

In this section, the simulation will be explained. Assumptions are made in the simulation and there are a few different types of simulations being used. The differences between the simulations are only the starting network.

For testing and understanding, the simulation is preferred on the generated networks, as it has a shorter computing time. Both generated networks are created with networkx library [14]. The small graph is created with "networkx.barabasi_albert_graph(100,2,65)" and the bigger network is created with "networkx.barabasi_albert_graph(500,10,65)". After generating the network each channel is assigned with random values for the fee, balance, and timelock just like in figure **??**.

Algorithm 1: Network generation

```
for edge in EdgesInNetwork do
    edge.Delay ← 10 * RandomIntegerBetween(1,10)
    edge.BaseFee ← 0.1 * RandomIntegerBetween(1,10)
    edge.FeeRate ← 0.0001 * RandomIntegerBetween(1,10)
    edge.Balance ← RandomIntegerBetween(100,10000)
end
```

After generating the network, attacker nodes are chosen concerning centrality, nodes with the highest centrality are chosen as attacking nodes. Central nodes are chosen because they are a better fit to attack as they are more likely to pass transactions, and the number of attacking nodes depends on the percentage of attacking nodes wanted, which changes in our experiments.

There are also simulations on the network built from a Snap Shot of the Lightning Network. Which allows the simulation to be as close to the real network as possible. All of the transactions are generated with random sender and receiver. The amount of the transactions differ a third being smaller than 10, a third of the transactions has an amount between 10 to 100, and the last third of the transactions have an amount that is between 100 to 1000. This allows the network to be simulated with transactions that carry different amounts.

Algorithm 2: Transaction Generation

```
iters ← 0 while iters < 100 do
    source ← pick a random sender node from the network
    sink ← pick a random sender node from the network
    amount ← 0
    if iters mod 3 == 0 then
        amount ← RandomIntegerBetween(1,10)
    end
    if iters mod 3 == 1 then
        amount ← RandomIntegerBetween(10,100)
    end
    else
        amount ← RandomIntegerBetween(100,1000)
    end
    transaction ← Routing(source, sink, amount, length)
    if transaction successful then
        iters ← iters + 1
    end
end
```

One of the main assumptions in the simulation is each transaction is instant, this is mainly related to the amount being transferred. Meaning normally transactions are not instant, and as the transactions are sent through nodes the amount is locked in the channel. This blocks the channels and the upcoming transactors choose their routes depending on this traffic. However, in our simulation, since all transactions are instant, there is nothing such as the locked amount in channels, which is important as the free amount in channels can fluctuate in the real network [2], making it harder for attacking nodes to find the sender.
Another assumption is related to the use of shadow routing, which is changing the timelock of the routing [8]. Hash Time Locked Contract (HTLC) allows the payment to be canceled if the payment does not get to the target node in the given time. When sending a transaction, the sender can choose this timelock, it is usually chosen to be the minimum time on the decided route, but by choosing the timelock bigger, the receiver can be hidden. The remaining timelock is used in the attacks and plays a big role in finding the target node. In our simulation, shadow routing is not used.

3.6 Attack

First, the same attack as in [1] will be used and evaluated to see how much the new routing algorithm improved against that attack. Afterward, the attack will be changed to attack the new routing algorithm.

The way the attack on LND routing works is simple. A set of possible destinations are created up to 4 hops with the information of the left timelock and the amount being transferred.

Algorithm 3: Getting the possible receivers

Input : timelock left, Cost function used to calculate the cost, The Network, Amount of the transaction, Previous, Attacker, and the next Node,

Output: Possible receivers with the possible senders

```
ReceiversToSenders ← dictionary
level ← 0
Nodes[level] ← [next]
flag ← True
while flag do
    Level ← level + 1
    if level is 4 then
        break
    end
    for current in Nodes[level] do
        for  neighbours of current do
            if neighbor != previous AND
            neighbor != adversary AND
            timelock remaining >= 0 AND
            transaction is possible then
                Nodes[level] ADD neighbor
                Previous[level] ADD current
            end
        end
    end
    if Nodes[level] IS empty then
        flag ← False
    end
end
level ← level - 1
while level >= 0 do
    for node in Nodes[level] do
        if timelock of node is 0 then
            path ← build the path from previouses
            if there are no duplicate nodes in the path then
                sources ← GETSENDERS(Network,receiver,path,amount,cost function)
                if sources is not Empty then
                    ReceiversToSenders[receiver] ← list(sources)
                end
            end
        end
    end
    level ← level - 1
end
RETURN ReceiversToSenders
```

After having possible receivers, the possible sender set is created by going through the possible receivers and checking which nodes would have the most optimum path to the possible sink, while the path includes the attacking, previous, and the next node.

Algorithm 4: Getting the possible senders

```
Input  : Possible receiver: target, Cost function used to calculate the cost, The Network,
         Amount of the transaction, path,
Output: Possible senders to the given target
pque ← priority queue of cost and node
pque.put((cost to target,target))
while pque is not empty do
    current cost, current ← pque.get()
    for neighbours of current do
        if the transaction to neighbor is possible then
            cost ← calculate the cost of going to neighbour from current
            if there is no other route to the neighbor cheaper then cost then
                Path[current] ← neighbour + Path[current]
                pque.put((cost,neighbor))
            end
        end
    end
    current is in path  current is attacker flag1 ← 1 if current is sender to the attacker then
        sources.append(previous)
        flag2 ← 1
    end
    if flag1 is 1 AND flag2 is 1 then
        if Previous is in Path[current] then
            for neighbors of current do
                if neighbor not in Path[current] then
                    sources.append(neighbor)
                end
            end
        end
    end
end
RETURN set(sources)
```

Attack on the new routing algorithm can be improved. The attack on the LND will not be best suited as the optimum route is mixed with random walks in new routing algorithms. However, since the routing algorithms still use the LND routing in some ways, if the attacking node is after the inserted random walk, the unmodified attack may successfully find the target node. Therefore, we will consider both attacks, the modified one and the attack used in the LND routing algorithm [1].

The modified attack first gets the potential sources and targets, just like in the existing attack. In the process creation of the possible sources and targets was changed to be less strict and only considered if the transaction was possible to send regarding the fee and the amount. However, the less strict version inflated the possible sender and receiver sets when run on the snapshot in table 4. Therefore, the strict version of creating the possible senders and receivers is better. After creating a possible set of sources and targets, the very same routing algorithm is used to generate some paths.

After generating 10 new paths for each source and target combination, each of the possible paths is added to a list. Then the most popular path that includes the attacking node in it will be chosen as a guess. If the source and/or the target match with the real sender receiver, then the attack will succeed. This assumes that the attacking nodes do not share information. However, in reality, there might be more than 1 attacking node.

Therefore, there is the second part to this attack. The second part assumes that all attacking nodes share all the information they have. Thus, for a transaction, all attacking nodes in the path of that

transaction will share their generated paths with the possible sources and sinks. Afterward, from all those generated paths, the most popular one including all attacking nodes will be chosen as a guess. This new attack takes a probabilistic guess by using the information about adversaries, and the weighted random walk insertion. However, it has some flaws although the first one is it increases the possible set of sources and targets, this could be overlooked as there is a random walk insertion in the routing. The modified attack can still be used with the conservative way of choosing the possible senders and receivers, which performs better in bigger networks which will be mentioned in section 4.1 Results. In the modified attack, possible paths are generated, but for generation, the amount getting transferred and the length used in the routing algorithm needs to be known. Thus, we will consider the worst case where the attacker knows the amount and the length that users had used. The length that is used in the algorithm can not be known, but the amount used in the transaction can easily be guessed, so it is accurate to assume that the attacker knows the amount. Therefore, this attack will be evaluated under different scenarios, with different lengths, and with different routing algorithms.

4 Evaluation

For evaluation, what metric is used, and which metric corresponds to which attack need to be explained.

- "Average fee per transaction" is important for the user and depends on the routing algorithm. Most of the time, the gain in anonymity is paid with the average fee per transaction.
- Another price of anonymity is "Average number of hops per transaction", the higher this number is the busier the network, which makes it more prone to attacks like denial of service [5].
- "Average anonymity source size" is the average number of possible senders per transaction. As mentioned in section 3.6 the bigger this average number the harder it is to find the real sender, since the sender set is used in generating possible paths.
- "Average anonymity sink size" is the same as Average anonymity source size and it is the average number of possible receivers per transaction.
- "Clean any singular ratio" gives the ratio of how many of the possible senders, or receiver sets had only 1 node in it. It just means the attacker knows the sender or the receiver with 100% certainty. This metric is specifically important for the unmodified attack on the LND routing from [1], because it is essentially the success rate of that attack.
- "Clean all singular ratio" is just like Clean any singular ratio, but instead of success defined as knowing the sender or the receiver, rather success is defined as knowing the sender and the receiver.
- Finally our main metrics to evaluate the attack are defined as "single attack and/or", and "combined attack and/or". The single attack is when the attacking nodes do not share information in between each other and guess the sender and the receiver depending on only their knowledge as explained in section 3.6. Therefore, "single attack and" is the ratio of correctly guessed sender and receiver, while "single attack or" is when the single attack guessed the sender or the receiver correctly. The combined attack is when attacking nodes share their information with each other and guess together. "Combined attack and" is the ratio of attackers guessing the sender and the receiver correctly, and "combined attack or" is the ratio of attackers guessing the sender or the receiver correctly.

4.1 Results

First, the modified attack needs to be checked whether it is good against weighted random walk insertion (WRWI) as it gains most of the anonymity against the LND attack in our simulations on the small network. The data from this attack is performed on a randomly generated network to ease the computing. As explained in section 3.6, it is hard for the attacker to guess the length being used in WRWI routing, so the worst-case scenario is assumed first, the attacker and the user are using the same length as input.

Table 1: Modified attack on routing algorithm 3, Weighted Random Walk Insertion with different lengths, attack knowing the length. Simulation on the 100 node network with 10% attacking nodes

Parameter	length 2	length 3	length 4	length 10
Clean any singular ratio	0.39%	0.0%	0.0%	0.0%
Clean all singular ratio	0.0%	0.0%	0.0%	0.0%
Average number of hops	5.18	5.83	7.12	11.43
Average fee	2.26	2.39	3.42	5.57
Average anonymity source size	12.63	14.09	15.16	18.00
Average anonymity sink size	5.46	4.54	3.57	1.93
Single attack and	4.25% ±3.95%	3.30% ±3.50%	2.45% ±3.03%	1.35% ±2.26%
Combined attack and	15.53 ±7.10%	5.50% ±4.47%	7.69 ±5.22%	7.69% ±5.22%
Single attack or	38.22% ±9.52%	26.73% ±8.67%	21.80% ±8.09%	10.36% ±5.97%
Combined attack or	61.17% ±9.55%	69.72 ±9.00%	64.42% ±9.38%	56.73% ±9.71%

From Table 1, it can be seen that most of the anonymity gained is diminished with the modified attack. Overall, when all attackers communicate, attackers know either the sender or the receiver around 60% of the transactions. It is also clear that when the length increases, anonymity also increases. However, when the length of the routing algorithm is 10 the fee and the network traffic are double. So, any length longer than 10 might not be preferred or might introduce service problems. However, having 10% of the network is hard [10], the more realistic scenario is single nodes attacking, and in that case, 10% to 40% of the transactions had their sender or receiver leaked, depending on the length that the routing algorithm uses.

In table 1 there was the assumption that the attacker guesses the length the user was using correctly, so how effective this attack will be with not matching lengths. In table 2, length is the length that the user used to generate the path with WRWI and the g is the attacker WRWI length which is used to generate possible paths as explained in section 3.6.

Table 2: Modified attack on routing algorithm 3, Weighted Random Walk Insertion with different lengths, attack doesn't know the right length. "g" is the length used in attack. Simulation on the 100 node network with 10% attacking nodes

Parameter	length 2, g = 3	length 3, g = 2	length 4, g = 2	length 10, g = 4
Clean any singular ratio	0.36%	0.0%	0.0%	0.0%
Clean all singular ratio	0.36%	0.0%	0.0%	0.0%
Average number of hops	5.31	6.35	7.60	11.43
Average fee	2.31	2.69	3.40	5.56
Average anonymity source size	13.53	16.67	15.71	18.00
Average anonymity sink size	5.29	4.15	3.54	1.93
Single attack and	4.30% ±3.98%	2.42% ±3.01%	2.28% ±2.93%	1.35% ±2.26%
Combined attack and	7.41% ±5.13%	6.73% ±4.91%	0.98% ±1.93%	7.69% ±5.22%
Single attack or	32.62% ±9.19%	25.81% ±8.58%	19.29% ±7.73%	10.51% ±6.01%
Combined attack or	75.0% ±8.49%	69.23% ±9.05%	68.63% ±9.09%	45.19% ±9.75%

From table 2 it can be seen that the length used in attack mismatching the length used in the routing does not affect the success of the attack substantially. As it can be seen as long as the length used in the attack is close to the length used by the user the anonymity does not get affected much. However, even when there is a big difference between two lengths, the success of single attacks does not get affected; and the combined attack can still guess the sender or the receiver with relatively high probabilities.

From table 2 and 1 it is clear that the anonymity of the WRWI routing algorithm is reduced with the modified attack. Thus, a solution to this could be making a new routing algorithm that randomly chooses one of the routing algorithms. In table 3, the anonymity of other routing algorithms against the modified attack is measured.

Table 3: Modified attack on different routing algorithms, attack uses the Weighted Random Walk Insertion with length 4. Simulation on the 100 node network with 10% attacking nodes

Parameter	Algorithm 0:LND	Algorithm 1:RH len 4	Algorithm 2:RWI len 4
Clean any singular ratio	0.65%	1.17%	0.35%
Clean all singular ratio	0.0%	0.0%	0.35%
Average number of hops	3.62	3.75	7.10
Average fee	1.05	1.19	3.41
Average anonymity source size	12.13	11.15	10.87
Average anonymity sink size	5.6	6.15	2.53
Single attack and	3.92% ±3.80%	2.92% ±3.30%	3.52% ±3.62%
Combined attack and	4.95% ±4.25%	6.54% ±4.85%	9.71% ±5.80%
Single attack or	62.75% ±9.47%	62.57% ±9.49%	25.7% ±8.56%
Combined attack or	82.18% ±7.50%	75.7% ±8.41%	59.22% ±9.63%

From table 3 it can be seen that the modified attack is more revealing against other routing algorithms implemented and LND compared to WRWI. This means that the idea of implementing a routing algorithm that chooses randomly one of the routing algorithms that was talked about would not be effective.

All of these results were generated with a randomly generated network with 100 nodes, instead of the snapshot of the Lightning Network. In table 4, the results were run on the snapshot of the Lightning Network, which is a much bigger network than the randomly generated one. Since this network is much bigger and the possible sender, receiver sets were created with fewer restrictions, there was a change in attack, the generated paths were only generated from the first 500 possible senders and receivers.

Table 4: Modified attack on routing algorithm 3, Weighted Random Walk Insertion with different lengths. Simulated on snapshot of the Lightning Network, with 0.4% attacking nodes

Parameter	length 4, g = 10	length 4, g = 4
Clean any singular ratio	4.26%	4.26%
Clean all singular ratio	0.0%	0.0%
Average number of hops	7.42	7.42
Average fee	15.38	15.38
Average anonymity source size	1297.95	1297.95
Average anonymity sink size	525.9	525.9
Single attack and	0.0%	0.0%
Combined attack and	0.0%	0.0%
Single attack or	0.0%	0.0%
Combined attack or	0.0%	0.0%

From table 4 it can be seen that the modified attack did not work, and only 4% of the sources or sinks were revealed. The main reason for the attack not working is because the connectivity and the number of nodes in the Lightning Network are much bigger than the randomly generated network. It should not be forgotten that the anonymity sizes were reduced to get the results with the limited time. However, the network is huge and highly connected helps anonymity a lot.
This attack did not perform well on the snapshot because it is a highly connected big network. A simulation on a network with 500 nodes and every node being connected to 10 nodes would be a faster way to test for the results. When the attack was performed on the snapshot, there were only 20 highly connected attacking nodes, but the number of these attacking nodes is important too. Thus, by testing with the different number of attacking nodes on the bigger network, we can make assumptions about how much of the network is needed to attack the network successfully. In table 5 over 10% adversary results were generated with the modified attacks, but choosing the anonymity sets conservatively like in the unmodified attack, as can be seen by the anonymity set size drop.

Table 5: Modified attack on routing algorithm 3, Weighted Random Walk Insertion with different lengths, Simulated on a 500 node network with a degree of 10. The percentage is the percentage of attacking nodes

Parameter	2% lnd, g = 4	2% length 4, g = 4	10% length 4, g = 4
Clean any singular ratio	20.07%	0.0%	4.17%
Clean all singular ratio	14.07%	0.0%	14.58%
Average number of hops	2.71	3.28	3.675
Average fee	0.59	0.94	1.11
Average anonymity source size	97.47	100.87	1.94
Average anonymity sink size	144.28	202.05	37.66
Single attack and	0.0%	0.0%	16.67% ±7.31%
Combined attack and	0.0%	0.0%	17.5% ±7.45%
Single attack or	0.0%	0.0%	18.23% ±7.57%
Combined attack or	0.0%	0.0%	23.33% ±8.29%

From table 5 we can see that the attacks are not successful at all on a highly connected network with a low percentage of adversaries. But considering the high percentage of adversaries anonymity is compromised, although only around 20%. Also having a 10% adversaries is quite high and expensive from recent researches [10]. However, these results were on a simulated network because there was not enough time to test them on the snapshot of the network.

5 Conclusion and Future Work

In conclusion, the question was "Could we improve the anonymity of blockchains in the case of payment channel networks with length-bounded random walk insertions?" From the simulations with the randomly generated networks that we ran there was not a substantial anonymity gain by using random walk insertions. However, from the simulations that were run on the snapshot of the Lightning Network, it is safe to say that weighted random walk insertion substantially increases the anonymity of users, as long as the network is large and highly connected. In the beginning, the random walk insertions gained anonymity against the attacks that were designed to crack LND. However, after modifying the attack against the weighted random walk insertion routing algorithm, most of the anonymity gained was diminished in the small randomly generated network. However, these results were not able to be translated to the snapshot simulation.
Moreover, we hypothesize that anonymity can be gained back with shadow routing [8], where the sender does not give the expected HTLC value to the transaction. This would help substantially because the attack is much more effective at finding the target node, then the source and shadow routing will decrease the chance of finding the target substantially since the available information decreases.
Another way to increase anonymity might be by using different routing algorithms. Instead of more than 90% of the network using some type of LND, if there were different types of optimum path algorithms, the anonymity could increase. The reasoning behind this is that the random walk insertion algorithms were using LND, they were open to the same attacks. And we hypothesize that a random walk insertion algorithm not based on LND, but a different alternative to it could increase anonymity against this modified attack.
For future work, the anonymity of shadow routing could be tested. Other random walk insertion algorithms based on other efficient routing algorithms, could be tested against the modified attack. But most importantly a way to simulate the channels getting locked like in the real network would give more accurate results for these experiments. It is also important to research these anonymity issues with different types of networks as we saw from the results of the simulation.
In the future, it would also be interesting to research the anonymity with changing percentage of adversaries and consider the costs of possible attacks.
Overall, the Lightning Network must use different routing algorithms, with different cost functions and shadow routing. However, as it is right now, weighted random walk insertion algorithms are highly likely to protect your identity as a sender, and likely to protect the identity of the receiver depending on where the attack is performed. And from our simulations on the snapshot of the network, only around 4% of the transactions had an identifiable sender or receiver.

6 Responsible Research

This work was done to explore the anonymity of the Lightning Network. In the process, instead of using the real network, simulation of the network was used. Using the simulation not only helped to get better results but also protected the network. Creating many transactions in the real network would cause traffic in the network, and would cost a lot of money because of the fees. The transactions were all generated, so nobody's anonymity was diminished. Since everything was simulated, there was no harm to anyone and the network, however, there were some assumptions in the simulation. The assumptions were also talked about in section 3.5, but mainly in the real network the anonymity revelations from the attacks might be lower because of the channels getting locked and routing being sub-optimal. There is another research about the sub-optimal routes [9].
Another concern of proof-of-work for cryptocurrencies like Bitcoin is the environmental impact, because of high electricity consumption. From the recent news about how much carbon emission bitcoin is creating, Layer 2 solution like the Lightning Network is promised to decrease this emission. Since in lightning not every transaction needs to be saved on the blockchain. Although the routing algorithms

we implemented overall increased average hops per transaction, meaning more energy will be used per transaction, it is a much smaller impact since the transactions on the Lightning Network uses much less energy than the transactions on the layer one chain.
When it comes to the reproducibility of the results, all the algorithms were explained in detail and the source code is public [13]. Meaning these results are highly reproducible except for small differences, because of all the randomness coming from the generations of the networks, transactions, and routing algorithms. The seed to the randomness could be shared but unfortunately, it was forgotten, and there was no time left to rerun the simulations. But to compensate for that mistake, we gave 95% of the confidence interval in the tables, and in the future a mistake like this won't happen.

References

[1] Kumble, Satwik Prabhu and Epema, Dick and Roos, Stefanie. (2021). How Lightning's Routing Diminishes its Anonymity. Proceedings of the 16th International Conference on Availability, Reliability and Security (pp. 1-10).

[2] Joseph Poon and Thaddeus Dryja. (2016). The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. https://lightning.network/lightning-network-paper.pdf

[3] Chen Chen* and Adrian Perrig. (2017). PHI: Path-Hidden Lightweight Anonymity Protocol at Network Layer. https://www.youtube.com/watch?v=aVfTm9U5Qb4

[4] G. D. Stasi, S. Avallone, R. Canonico, G. Ventre. (2018). Routing payments on the Lightning Network.

[5] Elias Rohrer and Florian Tschorsch. (2020). Counting Down Thunder: Timing Atacks on Privacy in Payment Channel Networks.

[6] Feng Xia, Jiaying Liu, Hansong Nie, Yonghao Fu, Liangtian Wan, Xiangjie Kong. (2020). Random Walks: A Review of Algorithms and Applications.

[7] Cameron Musco, Hsin-Hao Su, and Nancy A. Lynch. (2017). Ant-inspired density estimation via random walks.

[8] "BOLT #7: P2P Node and Channel Discovery.". (2016) https://github.com/lightningnetwork/lightning-rfc/blob/ master/07-routing-gossip.md

[9] Mihai Plotean, Stefanie Roos, Satwik Prabhu Kumble. (2021). Improving the Anonymity of the Lightning Network using Sub-Optimal Routes.

[10] S. Tikhomirov, P. Moreno-Sanchez and M. Maffei. (2020). A Quantitative Analysis of Security, Anonymity and Scalability for the Lightning Network.

[11] C. Grunspan, G. Lehéricy, and R. Pérez-Marco. (2020). Ant Routing scalability for the Lightning Network.

[12] D. Goldschlag, M. Reedy, and P. Syversony. (1999). Onion Routing for Anonymous and Private Internet Connections.

[13] "Lightning's Anonymity with Length Bounded Random Walk Insertions". (2021). https://github.com/emre6943/Attacking-Lightning-s-anonymity/blob/main/README.md.

[14] "Networkx Documentation". (2020). https://networkx.org/documentation/stable/index.html.