DoS-Resilient Onion Message Routing in the Lightning Network

by

Amin Bashiri

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Engineering

Department of Electrical and Computer Engineering
University of Alberta

Abstract

Onion messages (OMs) are private messages sent between nodes in the Lightning Network (LN) using onion routing. While they are intended to enable interesting applications such as reusable invoices, refunds, and asynchronous payments, OMs may also be used for unintended applications such as streaming data or spam. LN nodes can impose a rate limit on forwarding OMs to mitigate this. However, if not carried out carefully, the rate limit can expose the network to a denial of service (DoS) attack, where an adversary may disrupt or degrade the OM service by flooding the network. This DoS threat is particularly concerning because, under current specifications, a single OM can traverse hundreds of nodes, affecting all the nodes on its way. In addition, the adversary can hide their true identity thanks to the privacy-preserving nature of onion routing. We propose a simple solution to address this threat with two main components. The first component limits the distance over which OMs can travel. For this purpose, we introduce two methods: a hard leash and a soft leash. The hard leash strictly limits how far OMs can travel, while the soft leash makes it exponentially harder for OMs to traverse long distances. While the first method requires changes in the message format, the second method can be easily adopted without altering OMs. The second component of our solution consists of a set of simple yet effective forwarding and routing rules. We demonstrate that when these rules and the proposed leashes are applied, an adversary cannot significantly degrade the onion messaging service, assuming that the adversary does not control a significant fraction of funds in the network.

Preface

This thesis was conducted in the Department of Electrical and Computer Engineering at the University of Alberta under the supervision of Dr. Majid Khabbazian. The main content builds upon a paper [1] accepted at the Twenty-Eighth International Conference on Financial Cryptography and Data Security (FC).

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Majid Khabbazian, whose guidance, patience, and expertise were instrumental throughout this research endeavor. His mentorship enhanced my understanding of the subject matter and inspired me to strive for excellence in academic pursuits. I am truly fortunate to have had the opportunity to work under his supervision, and I am sincerely thankful for his dedication and invaluable advice.

Table of Contents

List of Tables

List of Figures

List of Symbols

Latin

$\mathcal{E}_{drp}$	An event that shows a path contains an adversarial node.
$\mathcal{E}_{sat}$	An event that shows a path contains an honest link saturated by the adversary.
$\mathfrak{b}$	The sum of all the rates the adversary can impose on honest links in the network.
E_C	The peer communication links between nodes.
E_L	The set of channels in the Lightning Network.
F_A	Total funds of the adversary.
F_H	Total funds of honest nodes.
F_i	The fund of honest node U_i.
f_i	The fraction of funds in the entire network that belongs to honest node i.
F_{avg}	The average fund held by any single honest node.
F_{max}	The maximum fund held by any single honest node.
G_C	The communication graph of the Lightning Network representing nodes and peer connections.
G_L	The graph of the Lightning Network containing nodes and channels.
$I_{i,j}$	A variable that is equal to one if the link from node i to node j is saturated by the adversary.
L	The maximum number of hops an Onion Message can travel.
l	Path-length parameter used in the path-selection algorithm.
n	The total number of honest nodes in the network.
P	The random path through the network, selected by a path-selection algorithm.
U_d	Honest destination node of an OM.
U_i	Node i.

U_s Honest source node of an OM.

V The set of nodes in the Lightning Network.

Greek

α_A The rate-limit coefficient of node A.

δ The ratio of the maximum fund held by any single honest node to the average fund across all honest nodes in the network.

γ The fraction of funds in the entire network that belongs to the adversary.

Abbreviations

BOLT Basis of the Lightning Technology.

DoS Denial of Service.

HTLC Hashed Timelock Contract.

LN The Lightning Network.

OM Onion message.

PoS Proof of Stake.

PoW Proof of work.

Glossary of Terms

Node's Funds The sum of capacities of channels owned by a node.

Chapter 1

Introduction

Bitcoin [2], introduced in 2008 by an anonymous entity known as Satoshi Nakamoto, was the first peer-to-peer electronic cash system. In simple terms, it allows people to send and receive money (Bitcoins) directly without needing a bank or any other intermediary. This decentralized nature is one of Bitcoin's key advantages.

Another significant benefit of Bitcoin is its enhanced security. Transactions are recorded on a public ledger, the blockchain, secured by cryptographic algorithms. This makes it extremely difficult for anyone to alter past transactions or counterfeit the currency. Additionally, Bitcoin can provide financial services to people who do not have access to traditional banking systems. For instance, someone in a remote area without a bank can still send and receive Bitcoins, with the only requirement being internet access.

These features have driven Bitcoin's adoption and popularity, establishing it as a revolutionary financial technology. However, despite its potential, Bitcoin faces significant challenges, particularly regarding scalability and transaction processing time. The network's current infrastructure can handle around 7 transactions per second [3, 4], which is insufficient for Bitcoin to fulfill its promise of becoming a global payment system when compared with centralized solutions like Visa [5, 6]. Additionally, the transaction processing times are quite high, with the current necessary time of around an hour for a transaction to be finalized. Addressing these issues is crucial for Bitcoin

to realize its potential as a decentralized global digital currency.

The Lightning Network (LN) [7, 8] is a payment channel network (PCN) [9, 10] created to address Bitcoin's scalability issues. In LN, two users (nodes) establish a bidirectional payment channel by locking funds in a funding transaction on the Bitcoin blockchain. This means opening a channel requires sending a transaction to the underlying blockchain (on-chain). However, once the channel is open, the two parties can send transactions back and forth directly to each other without sending any transactions to the underlying blockchain (off-chain).

Two nodes with an open channel between them can adjust the balance of funds in the channel to one side or the other by signing new transactions, called commitment transactions, that reference the initial funding transaction. However, they do not need to publish these updated commitment transactions on-chain. When they are finished transacting, they close the channel and broadcast the last commitment transaction to the Bitcoin blockchain, which reflects the last agreed-upon balance. This method enables instant and low-cost off-chain transactions, enhancing Bitcoin's scalability.

While a single payment channel enables two parties to transact directly and efficiently off-chain, its utility is limited to those two participants. However, the real potential of payment channels is unlocked when multiple channels are interconnected to form a network. This network allows users to route payments without a direct channel to the recipient. Known as multi-hop payments, this concept enables a payment to traverse several channels, with the payer paying a small fee to intermediary nodes to incentivize them to forward the transaction. As long as a path exists between the sender and receiver, funds can be securely transferred, effectively creating a decentralized payment system that scales beyond the limits of individual channels.

The current payment process on LN operates with unidirectional communication before executing each payment. In this process, the payee creates an invoice that contains all the necessary details to complete the payment and sends it to the payer. The invoice is typically communicated to the payer in an off-network manner, such as

by scanning a QR code. Once the payer receives the invoice, he executes the payment using the provided information.

In this process, the only communication before payment execution is from the payee to the payer through the invoice. However, many interesting use cases require bidirectional communication between the payer and payee before payment execution. Some examples of these use cases are refunds and reusable invoices. As LN evolves into a comprehensive financial platform, facilitating these use cases is becoming more essential.

LNURL (Lightning Network Uniform Resource Locator) [11] is a promising method that introduces bidirectional communication. LNURL integrates HTTP servers alongside LN nodes, allowing real-time bidirectional communication before initiating payments. This capability enhances LN's flexibility for various financial applications, including the ones mentioned above.

Despite its advantages, LNURL presents challenges. Users must maintain an HTTP server alongside their LN node, increasing resource demands, complexity, and potential security risks. Furthermore, the reliance on external communication methods can impact user privacy and anonymity.

An alternative, more recent, approach to facilitate bidirectional communication between nodes within LN is through Onion Messages (OMs) [12]. OMs are transmitted within LN directly between peers without the necessity of establishing channels between them. OMs, by design, leverage onion routing [13], a technique that ensures messages pass through multiple intermediate nodes without them being able to learn about the origin, destination, or the content of the message, enhancing users' anonymity and privacy.

While OMs offer significant benefits such as enhanced privacy and bidirectional communication within the network without prior channel establishment, they also introduce potential security challenges. One concern is the increased attack surface they create. Since OMs can be used to send anonymous messages that can traverse

numerous hops, there is a risk of adversaries exploiting this feature for Denial of Service (DoS) attacks [14, 15].

An adversarial node could exploit the network by flooding it with OMs, disrupting normal operations without the risk of detection due to the anonymity inherent in onion routing. The layered encryption of OMs obscures the origin of these spam messages, making it challenging to trace and block them. This complicates the prevention of DoS attacks, as opposed to conventional networks where the source of each message is identifiable and can be blocked if deemed malicious.

1.1 Contributions

This work proposes methods to enhance the OM service's robustness against DoS attacks. We examine two types of DoS attacks. The first type of attack involves an adversary flooding the network with OMs, aiming to degrade the OM service network-wide. This attack can overwhelm the network's capacity to handle legitimate messages effectively. The second type of attack targets a specific victim node by flooding it with OMs. This attack strategy intends to isolate the victim node from the rest of the network, disrupting its ability to properly communicate with other nodes through OMs.

We propose a solution consisting of two key components to mitigate these threats. The first component limits the distance OMs can travel within the network, reducing the potential impact of DoS attacks. The second component introduces a method to regulate the rate at which OMs are forwarded among peers, preventing excessive traffic that could overwhelm individual nodes.

Our theoretical analysis, supported by simulations on a network resembling the real-world LN, demonstrates that implementing these components effectively mitigates the identified attacks, provided the adversary does not control a substantial portion of the network's total funds. This approach enhances the resilience of the OM service against malicious activities, ensuring a more reliable and secure commu-

nication infrastructure for participants.

Chapter 2

Background and Related Works

This chapter aims to provide the necessary context for the security problem that our method addresses. We begin with an overview of LN and an explanation of the current payment process. Next, we discuss the limitations of the existing payment process in supporting the advanced applications expected from a widely used payment network. We briefly explore how LNURL [11] can enable these features by integrating HTTP web servers with LN nodes while also increasing complexity and expanding the attack surface regarding privacy and security. Finally, we explain what OMs are and how they can facilitate the implementation of these features within LN.

2.1 Lightning Network

Despite Bitcoin's numerous advantages, it still has some issues and limitations on becoming the global payment network people use daily. We mention some of these problems here.

- **Latency.** Typically, a transaction on the Bitcoin blockchain first spends some time in the transaction pool and then has to have six confirmation blocks on top of the block it's included in to be considered finalized. With a block time of around 10 minutes, a transaction takes approximately more than an hour to become final on Bitcoin.

- **Fees.** In the Bitcoin blockchain, miners invest significant computing power and

energy to mine blocks in exchange for a financial reward. Each block added to the blockchain comes with a block reward and the fees associated with the transactions included in the block. Consequently, the current Bitcoin blockchain network is unsuitable for small payments, as the transaction fees might exceed the value of the payment itself.

- **Throughput.** Even if users accept the latency and are willing to pay the high fees, the limited throughput of Bitcoin will not allow all transactions to go through the blockchain. The throughput of the Bitcoin blockchain is around 7 transactions per second [3, 4]. This limit is much lower than needed to facilitate users' daily transactions [7].

- **Privacy.** Privacy on the Bitcoin blockchain is a significant concern due to its transparency. All transactions are recorded on a public ledger, meaning anyone can view the transaction details, including the sender, receiver, and the amount. Although Bitcoin addresses are pseudonymous, sophisticated techniques can link transactions to real-world identities [16–19], de-anonymizing users and undermining user privacy. This becomes a more significant issue if we want Bitcoin to be used for daily transactions and payments and not just for investment.

Layer two solutions can address the latency, fees, throughput, and privacy issues on the Bitcoin blockchain by enabling off-chain transactions still secured by the underlying blockchain. This thesis focuses on the Lightning Network (LN) [7, 8], a second layer payment channel network (PCN) built on Bitcoin.

The concept behind LN is intuitive: recording every single transaction on the blockchain is unnecessary. In LN, two parties open a bidirectional channel with each other by one or both [20] of them locking up funds in an on-chain transaction that requires both channel partners to cooperate and prevents either channel partner from spending the funds unilaterally.

Two on-chain transactions are required for each channel: one to open and fund the channel and another to close it, as illustrated in Figure 2.1. However, once the channel is open, the participating parties can exchange their balances off-chain as frequently as they wish. The on-chain transaction used to open the channel is called the "funding transaction", and the total amount of funds deposited in this transaction is referred to as the "channel capacity".

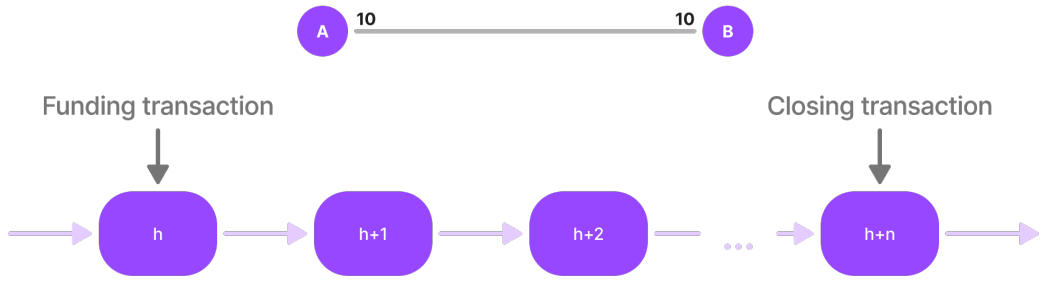

Figure 2.1: On-chain transactions for funding and closing a channel.

After opening a channel, both parties can send funds up to the channel's capacity back and forth with off-chain transactions by updating the commitment transactions that spend the funding transaction's output. Each transaction in the sequence uses Bitcoin's scripting language; thus, a Bitcoin smart contract manages the negotiation of funds between channel partners. The smart contract is set up to penalize a channel partner who tries to submit a previously revoked channel state. The off-chain transactions executed on the channel are instant and low-cost while leveraging the security of the underlying blockchain. When the channel partners decide to settle the final balance, they will publish the last commitment transaction on-chain, which sends the funds from the funding transaction to their chosen addresses.

Until now, we have described how two parties can leverage an LN channel to send low-cost instant payments back and forth off-chain. However, it is difficult to expect any two parties that want to send payments to each other to establish a channel since it requires an on-chain transaction, which is slow and costs high fees. To address this, once several participants have channels from one party to another, payments

can also be "forwarded" through intermediary parties by choosing a path across the network from the payer to the payee that goes through several payment channels. For instance, Alice can send a payment to Charlie without a direct channel with him by routing the payment through intermediary nodes like Bob and Eve. Alice would then pay small fees to Bob and Eve to incentivize them to forward the payment, as shown in Figures 2.2 and 2.3.

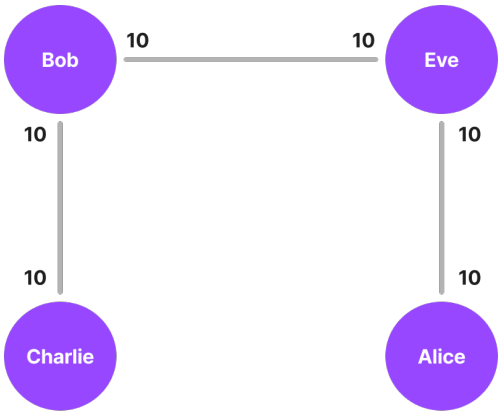

Figure 2.2: The initial state where Alice wants to send a payment of 2 coins to Charlie without having a direct channel with him.

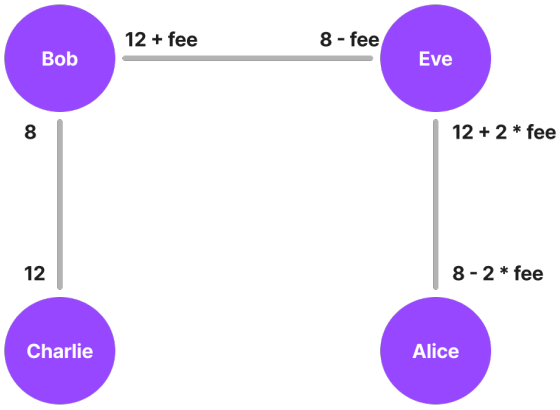

Figure 2.3: The final state of channel balances after Alice has sent a payment of two coins to Charlie through the intermediary nodes.

The LN uses onion routing [13] to ensure privacy and security while forwarding payments through intermediary nodes. This method, originally developed for secure communication, encrypts the payment information in layers similar to the layers of an onion. As the payment is forwarded through each intermediary node, only the information relevant to that specific node is decrypted, while the rest remains encrypted. This approach prevents any intermediary from knowing the full path, the payment amount, or the identities of the sender and receiver. By using onion routing, LN not only safeguards user privacy but also ensures that payments are routed efficiently and securely across the network.

This approach offers significant advantages. First, it is simple and secure, handling off-chain transactions while ensuring security through the underlying blockchain. Sec-

ond, transactions between peers in their channels are settled instantly, with negligible fees paid to the intermediary nodes to forward the payment in the network. Most importantly, network throughput can be significantly improved, making the system more efficient and scalable.

It is important to understand that the Lightning Network is nothing more than an application on top of Bitcoin, using Bitcoin transactions and Bitcoin Script. The LN protocol is a creative way to get more benefits from Bitcoin by allowing an arbitrary amount of instant payments with instant settlements without having to trust anyone else but the Bitcoin network. LN addresses all the previously mentioned concerns. Here's how:

- **Latency.** LN allows transactions to be conducted off-chain. Only the initial and final states are recorded on the underlying blockchain. This significantly reduces the time users wait for confirmations, as transactions can be settled instantly off-chain.

- **Fees.** By conducting transactions off-chain, LN reduces the need to pay high on-chain transaction fees. Since only the opening and closing of a channel are recorded on the blockchain, the intermediate transactions incur minimal fees, making micropayments feasible.

- **Throughput.** LN dramatically increases the transaction throughput by allowing numerous transactions to occur off-chain. LN can handle thousands of transactions per second compared to the Bitcoin blockchain's 7 [3, 4] transactions per second.

- **Privacy.** LN enhances user privacy through several key mechanisms. Keeping transactions off the public ledger ensures they are not permanently and publicly recorded. LN also employs onion routing for payment forwarding, which prevents intermediary nodes from learning the payment amount, source, or desti-

nation. Additionally, channel balances in LN are private to the two participants involved, so network nodes cannot determine payment flows by inspecting these balances. Consequently, transactions on LN remain visible only to the source and destination [7, 8].

2.1.1 Payment Flow

In LN, the payment process begins with the recipient generating an invoice. This invoice is a unique, encoded string containing information such as the payment amount, expiry time, the recipient's node ID, the amount, and the hash of a unique secret preimage. The payer scans a QR code or copies this invoice into their LN wallet to initiate the payment. Upon receiving the invoice, the payer's wallet first verifies the invoice details and then searches for a payment route across the network of LN nodes and channels using the publicly available LN topology data it has received by gossiping with other network nodes [21].

Once a suitable route is found, the source node generates an onion-encrypted payment packet and forwards it to the first intermediary node. The payment is locked using the hash of the preimage, a secret piece of data known only to the recipient. The onion-encrypted message then travels through the selected route, with each node forwarding the payment until it reaches its destination. Upon receiving the payment, the recipient reveals the preimage, allowing all nodes along the route to unlock and settle the transaction. This process ensures quick, atomic, and secure off-chain payment settlements without recording each transaction directly on the Bitcoin blockchain, thus achieving scalability and efficiency.

However, this approach cannot facilitate some functionalities required for a global payment network like LN, mostly because of the lack of bidirectional communication before payment execution. With the current process, there is only a unidirectional communication from the payee to the payer through the exchanged invoice.

2.1.2 Shortcomings and Solutions

In this section, we point out some core features that LN lacks, which prevent it from becoming the global payment system it aspires to be. Some of these features are:

- **Streamlined Refunds.** In the current payment flow, only the payee must send an invoice via an off-network method when initiating a transaction. However, if the payee needs to issue a refund, the original payer must follow the same process and provide a new invoice for the refund through off-network communication. This approach is impractical for merchants, as they cannot expect customers to generate invoices for refunds. Instead, merchants should be able to issue refunds automatically without requiring any action from the customer.

- **Reusable Invoices.** The current invoices can only be used once since they contain the preimage, which should be kept secret and revealed after payment completion. This prevents users from publishing an invoice once and receiving multiple payments.

- **Asynchronous payments.** With the current payment flow, both the payee and the payer must be online simultaneously for the payment to go through successfully. The requirement for the payee to be online at each payment can be inconvenient, especially for small, everyday transactions expected to go through instantly.

Facilitating the features mentioned above requires bidirectional communication before payment execution. However, the current invoices only facilitate one-way communication from the payee to the payer before payment execution.

One of the promising methods that was built to facilitate these use cases and more is LNURL [11]. With LNURL, users also run a web server alongside their LN node. The webservers will facilitate the bidirectional communication necessary for the mentioned features. Instead of sending the invoice, the payee will communicate

with the payer how to reach his web server and some possible query parameters to pass to the web server.

LNURL is very effective in addressing the requirements to facilitate the mentioned features. However, as mentioned, integrating LNURL requires running a web server alongside the LN node. This requirement makes it more difficult for users to run and maintain LN nodes. Running a separate web server can also introduce new privacy and security challenges.

Onion Messages (OMs) [12] were introduced to provide the required bidirectional communication more securely and privately inside the same network of LN nodes.

2.2 Onion Messages

OMs enable lightweight, private, and secure communication within the Lightning Network (LN) by utilizing onion routing [13], similar to the Tor network. While OMs [12] can facilitate the interesting features mentioned above, they can also be used by a malicious actor to execute a Denial of Service (DoS) [22] attack on the network.

DoS attacks using OMs are difficult to prevent because of their distributed and privacy-preserving nature. OMs use layered onion encryption to conceal the message content from intermediary nodes, revealing only the necessary routing information for each node to forward the message to the next. This layered encryption ensures that intermediary nodes cannot determine the message's origin, destination, or payload, preserving privacy at every step of the communication process.

Unlike the payment messages, which can only be 1366 bytes, OMs can also have a bigger size of 32834 bytes. This bigger size was introduced to enable nodes to forward bigger payloads containing more information. However, in Appendix A, we show that this bigger size can be used to create messages that travel for hundreds of hops; a malicious actor can leverage this, making the DoS attack using OMs more effective.

Moreover, an adversary can leverage Sybil attacks to further enhance the impact of

a DoS attack within the LN. A Sybil attack occurs when an attacker creates multiple nodes to gain more influence over the network. In the context of OMs, an adversary could deploy numerous Sybil nodes that participate in routing OMs. This increases the likelihood of successfully executing a DoS attack and makes it more challenging to detect and mitigate the attack, as the malicious traffic appears to be spread across many seemingly independent nodes. Combining Sybil attacks and exploiting larger OM sizes poses a significant threat to the LN's stability, allowing adversaries to execute highly effective and difficult-to-detect DoS attacks.

2.3 Related Works

2.3.1 Tor

Tor [23], a close counterpart to OM service, recently experienced a DoS attack [24, 25]. Tor's response involved integrating PoW as a defense mechanism [26, 27]. OM service, however, requires a tailored solution, as it differs from Tor in several aspects:

1. Tor messages have a default travel distance of three hops [28], whereas OMs are allowed to travel hundreds of hops as calculated in Appendix A.

2. Tor's nodes have distinct roles (entry, relay, exit), unlike the homogeneous nature of the OM service nodes.

3. While Tor focuses on anonymous internet access, the OM service facilitates lightweight communication with significantly lower transmission rates.

4. Implementing a rate limit proportional to invested funds, as proposed in this paper, is simpler in LN compared to the Tor network, as LN inherently integrates the concept of money.

2.3.2 The Probabilistic Method

A probabilistic algorithm proposed on the LN mailing list [29] suggests applying per-peer rate limits on incoming OMs to be relayed. The core idea is to track the neighbor that requested the forwarding of the last OM on each outgoing connection. When a message exceeds the rate limit, an `onion_message_drop` message is sent to the previous hop, indicating the breach and halving the rate limit for that peer. The previous hop then follows the same process, forwarding the message to its previous hop and similarly halving its rate limit with that peer. Although occasional misattributions may occur, this scheme aims to statistically penalize the correct source of the DoS attack.

For example, if a node, say A, receives a rate limit error from one of its peers, say B. It will refer to its memory to identify which peer had requested the last message forwarded to B. If C was the last peer who requested a message to be forwarded to B, the punishment for C is that A would respond by tightening its rate limit for C, essentially penalizing C for exceeding the rate limit. Subsequently, C will take the same measure for his peers, creating a chain of punishment. This cascade of adjustments will most likely propagate back to the original sender of excessive message requests.

The primary advantages of this method are its linear memory usage and simplicity. However, despite its simplicity, this method poses several challenges:

1. Originally designed for a scenario where OMs travel through peers with established channels, it does not match the current LN implementation that allows OMs to traverse peers without channels.

2. The method imposes strict rate limits on all links in a path after an adversary sends numerous messages, potentially affecting many peers given the maximum hop limit of 469.

3. Determining when and how to halt the rate limit chain propagation between peers requires careful consideration for balancing security and operational efficiency.

4. Reverting the network to normalcy from a strict rate limit is complex and crucial for maintaining resilience.

2.3.3 Our Work

In this work, we tailored a solution for the OM service, addressing the specific challenges outlined above and ensuring effective attack mitigation. We leveraged the nodes' sum of channel balances and assigned the rate limit proportional to it. This acts similarly to a proof-of-stake (PoS) mechanism and assigns the more invested nodes in the network a higher bandwidth.

Chapter 3

Mitigating DoS Attacks on Onion Messages

3.1 System Model

We model LN using a graph $G_L = (V, E_L)$, where V represents the set of nodes in the network, and E_L represents the set of channels. The communication graph is also represented as a graph $G_C = (V, E_C)$, where E_C denotes the communication links between nodes in LN. This work assumes that G_C is a complete graph. This implies that any two nodes in the network can establish a communication link (e.g., a TCP connection) to exchange OMs, and there's no need for two nodes to have an LN channel between them to forward a message. This aligns with what has been currently implemented in BOLTs [12], the standard reference outlining LN's implementation.

We assume that each node in LN maintains a list of all other nodes in the network, along with the amount of their funds, defined as the sum of the capacities of the channels they own. This assumption aligns with the LN, where channel capacities are publicly announced, even though channel balances remain private.

Node $A \in V$ may accept an OM from another node $B \in V$ and forward it to another node $C \in V$. However, A may limit the rate at which it forwards messages for B. We note that a specific rate-limiting algorithm is not prescribed in the OM specifications outlined in BOLTs [12]. However, nodes are strongly encouraged to implement a rate-limiting algorithm. In our model, we make the simplifying assumption

that all nodes within the network uniformly adopt the same rate-limiting algorithm. This simplification enables us to explore network behavior under a standardized rate-limiting framework.

Adversary. The adversary can create one or more nodes in the network by paying the fees for funding transactions and putting in the minimum channel capacity required and can send OMs using these nodes at any rate it wishes. However, we assume that the total funds locked in the channels owned by the adversary are small compared to the total funds in the whole network. We also assume that joining the Lightning Network does not present a major barrier to the adversary other than paying the funding transaction fee, but controlling a large portion of the total funds in the network does. Thus, our mitigation method leverages a Proof of Stake (PoS) like mechanism to prevent Sybil attacks.

Considering the above model, we explore two DoS attacks. We recognize that the adversary may harbor various motives for executing an attack within the Lightning Network. Two primary motives under consideration are as follows:

- **Degrading Network Availability:** In the first attack, the adversary attempts to flood the network with unnecessary OMs to degrade the OM service. This type of attack seeks to hinder the network's general functionality and its ability to efficiently transmit messages.

- **Disrupting a Specific Node:** In the second attack, the adversary targets a particular victim node with OMs to prevent other nodes in the network from reaching it. This targeted approach seeks to incapacitate a specific network participant, potentially for malicious purposes.

We've defined a straightforward success metric for our DoS prevention methods: the probability of successfully sending an OM between two honest nodes.

- In the network degradation attack scenario, we randomly select two honest

nodes as the source and destination. In this case, the success metric is the probability that the destination will successfully receive the message.

- We follow a similar approach for the attack aimed at a specific node, but the destination is fixed to the targeted node, while the source is chosen randomly. The success metric measures the probability of successfully delivering the message to the targeted node.

3.2 Proposed Solution

Our proposed solution comprises two straightforward components. The first component involves limiting the number of hops an OM can traverse. Drawing inspiration from the Tor network [23], which achieves anonymity with just three hops [28], we recognize that, since OMs move through peers rather than channels, nodes can efficiently reach any other node within a short hop count. While we may not necessarily restrict this number to three, we introduce methods to limit the maximum number of hops, as what is required is much lower than the existing 469-hop limit.

The second component of our solution involves implementing simple forwarding rules for nodes to follow. In the subsequent section, we analyze how the combined limitations on the maximum number of hops and the enforcement of these simple forwarding rules significantly mitigate the considered DoS attack scenarios.

3.2.1 Limiting the Travel Distance

While onion packets used for forwarding payments are limited to 1366 bytes, OMs can have a larger size of 32834 bytes [30]. The expanded size of OMs introduces a potential vulnerability that a malicious actor can exploit to send messages that maximize the available space to traverse numerous hops. A 1366-byte OM packet has a maximum travel distance of 19 hops, but a 32834-byte OM packet has a much larger maximum distance of 469 hops as calculated in Appendix A. To mitigate this,

we introduce two alternative methods to impose a cap on the maximum number of times a message can get forwarded in the network.

Soft leash.

The soft leash does not force a strict hard limit on the maximum number of hops for an OM. Rather, it makes it exponentially more difficult for a source node to create OMs that travel far. This effectively serves as a practical deterrent against excessively long paths while allowing for flexibility within reasonable limits. Additionally, it allows honest users the flexibility to choose a longer path to enhance their privacy when needed while making it more difficult for adversaries to generate large volumes of spam messages.

In this method, similar to the solution recently implemented in the Tor network [26], each hop requires the sender to provide proof of work (PoW). However, OMs within LN differ from packets in the Tor network in one key aspect: Tor uses a default path length of 3 and a maximum path length of 8, while LN does not impose such a strict limit. Given this fundamental difference between LN and the Tor network, we propose a PoW-based algorithm that scales exponentially rather than linearly with the number of hops. As illustrated in Figure 3.1, we achieve this by linking each hop's PoW to the preceding hop's, effectively creating a chain of PoWs. With this approach, each additional hop appended to this chain exponentially increases the computational challenge of calculating the PoW for the entire path. This limits the adversary's capacity to add hops far beyond the desired max number of hops set by adjusting the difficulty of PoW. Furthermore, this method naturally imposes a boundary on the adversary's potential fan-out factor.

Figure 3.1 illustrates the process undertaken by the source node A to establish shared secrets with nodes along the path (B, C, D) for creating an onion-encrypted message. For each node along the path, the hash resulting from the combination of the shared secret and the latest Bitcoin block hash must adhere to PoW conditions,

such as being below a certain value.

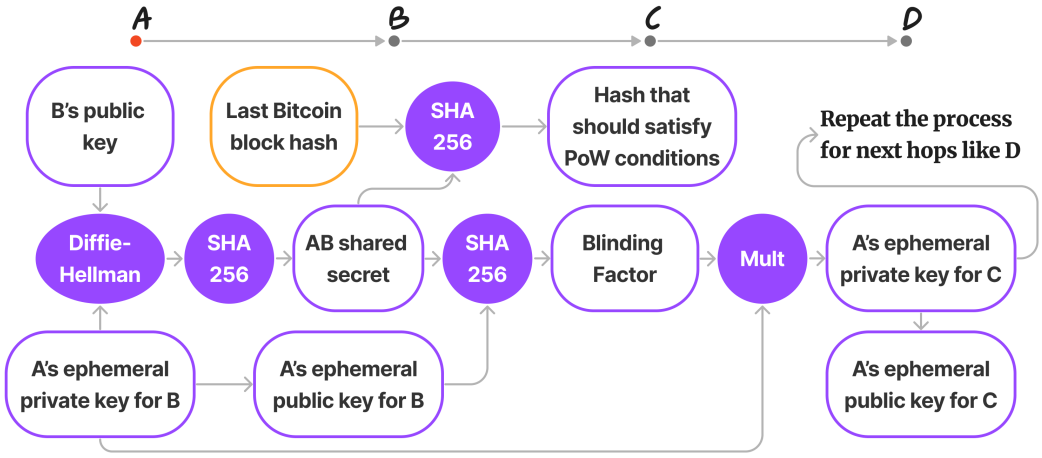

Figure 3.1: The source node PoW generation process.

The difficulty adjusts dynamically. Each network node chooses its acceptable target for the PoW difficulty and adjusts it based on the rate of messages it receives from its peers. It will then announce its PoW target in gossip messages.

For the technical implementation, we must understand that the onion encryption in LN is possible by the source node using the Diffie–Hellman algorithm to create shared secrets with each node in the path [30]. To make the messages untraceable between multiple hops, an additional measure involves blinding the key at each hop [30]. This deterministic blinding procedure enables the sender to compute corresponding blinded private keys during packet construction, effectively deriving all associated shared secrets. This enables the source node to create the OM and encrypt each layer using the relevant shared secret for the intended hop to decrypt. We demonstrate this algorithm for creating PoWs in Figure 3.1.

In the methodology we put forth, every node along the path expects the PoW provided for that hop to be less than a certain target. To meet this condition, the source node must repeatedly create temporary key pairs (ephemeral key pairs), which is the only changeable aspect of the entire process. The aim is to find a key pair that produces PoWs for each hop that is less than the target of that particular hop.

Subsequently, each node in the path verifies the PoW associated with that node.

Because PoWs are chained together, the computational demand to generate valid PoWs grows exponentially as the number of hops in the route increases. This limits the sender's capacity to add more hops. Furthermore, this method naturally imposes a boundary on the attacker's potential fan-out factor.

To prevent replay attacks on our PoW, we borrowed an idea from Tor. We use a hash table or a bloom filter for less capable nodes to keep track of previously used PoWs. However, unlike Tor's random seeds [27], we link our PoW to the latest Bitcoin block's hash as illustrated in Figure 3.1. Although Bitcoin block hashes are not a true source of randomness [31, 32], they suffice for our use case. This way, we deter precomputed PoWs, making the network more secure. We note that the hash table or bloom filter used for replay protection can reset with each new block since PoWs associated with previous blocks become invalid. This makes generated PoWs unique for each node at each block height, simplifying the process and enhancing network efficiency by omitting the need to gossip randomly chosen seeds. This approach is feasible because every LN node is connected to a Bitcoin node and can access on-chain data.

The only adjustable input to the process of creating the shared secrets with all the hops along the path is the ephemeral key pair that the source node chooses for the payment.

We want the difficulty to increase exponentially rather than linear because it's exponentially less justified for the source node to include more hops in the path. For example, it is justified if the source node wants four nodes on the path instead of three for more privacy. But it's significantly less justified if a node wants eleven nodes on the path instead of ten.

This method does not introduce any new attack surfaces because the protocol looks exactly the same from the outside. Each shared secret is only known to the source and the exact node with which it was supposed to be used.

To get a better sense of the PoW difficulty, we simulated the process of choosing ephemeral public-private key pairs, calculating shared secrets, and checking the PoW condition for different difficulties and path lengths. The result of this experiment can be observed in table 3.1. This table contains the average number of ephemeral key pairs tried by the source node to generate a valid PoW for all hops in a path based on different difficulties and path lengths. We can see that the difficulty increases exponentially with increasing the path length.

Path length	**3 leading zeros**	**4 leading zeros**	**5 leading zeros**
1	6	7	30
2	63	330	1194
3	433	4854	14415
4	2826	30382	4020887

Table 3.1: Average number of ephemeral key pairs tried to generate a valid PoW.

From table 3.1, we can see that by choosing the right difficulty for each hop, honest nodes can easily create a few messages that can travel for many hops, providing them with great privacy. At the same time, it will be very difficult for an adversary to create many messages that travel for many hops.

Hard leash.

This method enforces a strict upper limit on the hops an OM can traverse. As illustrated in Fig. 3.2, this is achieved by dividing the message into two segments, one for routing information and another for the actual payload. For example, consider a 32,834-byte message. The first 66 bytes of the message are dedicated to the message headers, leaving us with 32768 bytes. Let's divide the remaining 32768 bytes into two segments, a first portion of 280 bytes for routing information and a second segment of 32,488 bytes allocated for the payload. In this configuration, with the calculations in Appendix A, the limited size of 280 bytes dedicated to routing data restricts the

packet to a maximum of 4 hops. It is worth noting that these specific numbers are adjustable but need consensus among all nodes within the network.

The encryption and decryption processes for the message remain consistent, regardless of the division into sections. Each message segment, both routing and payload parts, will be encrypted for each hop by the source and decrypted at its corresponding hop. Although only the last hop will see the plain payload, this approach ensures that the message appears distinct at each hop, hence preserving its untraceable nature as required.

Figure 3.2 illustrates how an OM is structured into two parts while still fitting within a single packet. One part contains the routing information, while the other contains the payload, which is exclusively intended for the final hop. The message length remains constant regardless of the number of hops in the path.

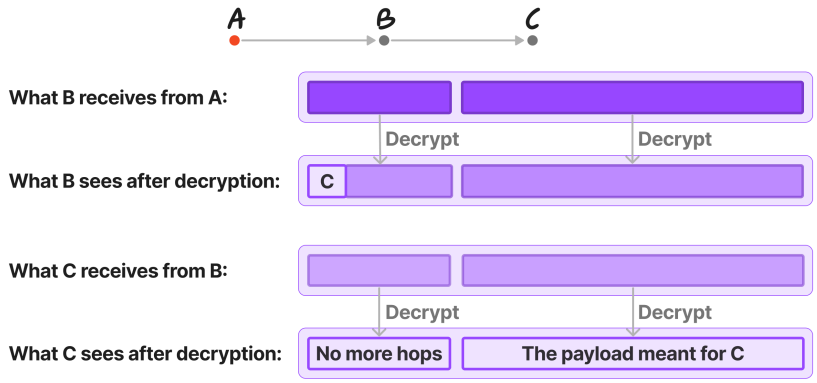

Figure 3.2: Hard leashed onion messages.

Importantly, both parts are encrypted by the source for each hop and get decrypted separately by each node on the path. At each hop, the path information part is also padded with bits to maintain consistent length throughout the route. Notably, intermediary hops only gain access to encrypted data in the payload section, with meaningful content revealed solely at the final hop after decryption. The gradual shift in color opacity signifies the successive removal of encryption layers from the message.

3.2.2 Forwarding and Routing

As the second component of our solution, we propose simple rules for nodes to follow when forwarding OMs and routing their messages. Importantly, the memory requirement to implement these rules scales linearly with the number of peers, which makes this method feasible.

Forwarding

In our approach, nodes do not limit OMs destined for themselves and always prioritize sending their own messages over forward requests of their neighbors. Because of this, the first and last link in the path will always be able to forward the message. Additionally, nodes do not limit the rate of OMs they send on any outgoing link. However, they enforce a rate limit on forwarding OMs from each of their peers. This rate limit is set to $\alpha_A \cdot F_B$ for peer B by node A, where α_A is an adjustable parameter for each node, and F_B is the sum of the capacities of channels owned by B. We refer to this sum as the *funds* of node B.

Setting the rate limit proportional to F_B ensures that nodes with greater investment in the network receive a higher rate limit. This strategy strengthens the system's resilience against Sybil attacks and significantly raises the financial barrier for potential attackers to impact the network. The coefficient α is included to ensure that each node only accepts forwarding requests up to its capacity or willingness.

Imposing this rate limit can be as straightforward as a node, say A, storing a `last_forward_time` variable for each of its peers. If A receives a forwarding request from, say B, it compares the time of this request with the `last_forward_time` of B. If the difference is large enough (specifically, greater than $\frac{1}{\alpha_A \cdot F_B}$), A will forward B's message and update its `last_forward_time`; otherwise, A discards B's message. However, nodes can also implement slightly more sophisticated techniques, such as sliding windows or token bucketing [33], to impose the rate limit more effectively.

Routing

Since our forwarding rules prioritize nodes with greater investment by allocating higher rate limits, source nodes should choose random paths weighted by the funds of the nodes in the network. This approach routes messages through higher-bandwidth links and reduces the likelihood of messages being routed through adversarial nodes, assuming the adversary controls only a small portion of the network's funds.

In essence, nodes with greater investment in the network receive more bandwidth, making it harder for adversaries to saturate these high-bandwidth links. When routing honest messages, we prioritize these high-bandwidth links to be included in the path.

Algorithm 1 outlines the proposed path-finding mechanism. This algorithm selects a path of a desired length from a graph by probabilistically choosing nodes using their funds as weights. It first computes the cumulative distribution of node funds, normalizing them to create cumulative probabilities. During path selection, for each hop, a random number between zero and one is generated, and the corresponding node is selected based on the cumulative probabilities. This process is repeated until the path reaches the desired length, ensuring that nodes with higher funds have a higher chance of being selected, resulting in a path that reflects the underlying distribution of node funds.

Algorithm 1: The proposed path selection algorithm.

```
Input: V ;     // A list of graph nodes V along with nodes' funds
Input: l ∈ ℕ ;                    // Desired path length where l < L
Output: path ;                        // A list of relay nodes from
                                    // the source to the destination
Function SelectNextNode(cumulativeProbabilities):
    r ← Random number between 0 and 1;
    foreach (node, cumulativeProbability) ∈ cumulativeProbabilities do
        if r ≤ cumulativeProbability then
            return node;
        end if
    end foreach
cumulativeProbabilities ← [];
totalFunds ← Σ_{node∈V} node.funds;
cumulative ← 0;
foreach node ∈ V do
    cumulative ← cumulative + node.funds;
    cumulativeProbabilities ←
     cumulativeProbabilities ∪ {(node, cumulative/totalFunds)};
end foreach
path ← [];
while |path| < l do
    next ← SelectNextNode(cumulativeProbabilities);
    path ← path ∪ {next};
end while
return path;
```

3.3 Resistance of the Proposed Solution

In Section 3.1, we outlined two potential attack scenarios that pose significant risks to the system. In this section, we conduct a detailed analysis to evaluate the effectiveness of the proposed solution in mitigating the impact of these scenarios.

3.3.1 Flooding the Network

In this attack scenario, the adversary attempts to degrade the availability of the OM service by flooding the network with a large number of OMs. The objective is to overwhelm the system by saturating as many links as possible and exploiting the bandwidth its neighboring nodes provide. This attack is designed to strain network

resources and disrupt normal operations, diminishing the overall service quality.

As discussed in Section 3.1, we assume the communication graph is complete, given that OMs traverse through peer connections rather than LN channels. Additionally, because a link can be saturated from one direction while still available from the opposite direction, we model the network as a complete directed graph without self-loops. This assumption provides a more accurate representation of the network's communication dynamics and potential vulnerabilities.

Theorem 1 *Let L denote the maximum number of hops an OM can travel, and $l \leq L$ denote the OM path-length parameter used in the path-selection algorithm (Algorithm 1). Let F_A be the total funds of the adversary, and $F_H = \sum_{i=1}^{n} F_i$ be the total fund of honest nodes, where F_i denotes the fund of honest node U_i, and n is the total number of honest nodes in the network.*
Then, for every pair of distinct honest nodes U_s and U_d, the probability that an OM message, sent on a random path chosen by Algorithm 1, from U_s to U_d fails is at most

$$\begin{aligned} 1 - (1-\gamma)^{l-1} + (L-1) \cdot (l-2) \cdot (1-\gamma) \cdot \gamma \cdot \delta \\ \leq (L-1) \cdot (l-1) \cdot \delta \cdot \gamma \end{aligned}$$

where

$$\gamma = \frac{F_A}{F_A + F_H}$$

is the fraction of funds in the entire network that belongs to the adversary, and

$$\delta = \frac{F_{max}}{F_{avg}} = \frac{\max_i(F_i)}{\left(\sum_i F_i\right)/n} = \frac{\max_i(F_i)}{F_H/n}$$

is the ratio of the maximum fund held by any single honest node to the average fund across all honest nodes in the network.

Proof. Let P be the random path selected by Algorithm 1. The OM sent on P fails if and only if at least one of the following events occurs: 1) event $\mathcal{E}_{drp}$: the path P contains an adversarial node, in which case the adversarial node will drop the

message; 2) event $\mathcal{E}_{sat}$: the path P contains an honest link saturated by the adversary[1], in which case the OM is dropped by the receiver of the link. In the following, we find an upper bound on the probability for each of the above two events. We then simply add these probabilities to get an upper bound on the probability of OM failure.

To select the random path P, Algorithm 1 chooses $(l-1)$ nodes at random in $(l-1)$ rounds, where in each round, the probability that a given node is selected is proportional to its fund. Therefore, $Pr(\mathcal{E}_{drp})$, the probability that Algorithm 1 selects a node that belongs to the adversary is simply equal to

$$Pr(\mathcal{E}_{drp}) = 1 - (1-\gamma)^{l-1}.$$

Next, we establish an upper bound on $Pr(\mathcal{E}_{sat})$.

For any two nodes $U_i, U_j \in V$, we assume that $\alpha_i = \alpha_j = \alpha$, indicating that all nodes in the network dedicate an equal computing capacity for forwarding OMs. The adversary can establish up to n connections to all the n honest nodes, where each connection can push a rate of at most $\alpha \cdot F_A$. Moreover, each connection can engage at most $(L-1)$ honest links (the first link is not an honest link as it is incident to an adversarial node, hence $(L-1)$ instead of L). Therefore, the sum of all the rates the adversary can impose on honest links in the network denoted $\mathfrak{b}$, is at most

$$\mathfrak{b} \leq \alpha \cdot F_A \cdot (L-1) \cdot n \tag{3.1}$$

Assuming they are honest links, the first and last links on path P do not drop the OM, even when saturated. This is because the source node prioritizes forwarding its own OMs, and the destination node always accepts OMs destined for itself. Therefore, in analyzing $Pr(\mathcal{E}_{sat})$, we only consider the middle $l-2$ links on the path P.

The probability that the kth link, $1 < k < l$, on the path P is a saturated honest link is

$$\sum_{i,j} f_i \cdot f_j \cdot I_{i,j}, \tag{3.2}$$

[1]A link other than the first and the last links as those links belong to, respectively, the source and destination and do not drop the OM message.

where $f_i = \frac{F_i}{F_A+F_H}$, $f_j = \frac{F_j}{F_A+F_H}$, $f_i \cdot f_j$ is the probability that the kth link, $1 < k < l$, is (U_i, U_j), and $I_{i,j}$ is a variable equal to one if the adversary has saturated (U_i, U_j), the link from U_i to U_j; and equal to zero otherwise. To set $I_{i,j}$ to one, the adversary needs to push OMs at a rate of at least αF_i from U_i to U_j. Since the adversary's maximum budget is $\mathfrak{b}$, we must then have

$$\sum_{i,j} (\alpha \cdot F_i) \cdot I_{i,j} \leq \mathfrak{b} \tag{3.3}$$

Thus, we get

$$
\begin{aligned}
Pr(\mathcal{E}_{sat}) &\overset{by(3.2)\text{ and union bound}}{\leq} (l-2)\cdot\left(\sum_{i,j} f_i\cdot f_j\cdot I_{i,j}\right)\\
&\overset{f_{max}=\max_i(f_i)}{\leq} (l-2)\cdot\left(\sum_{i,j} f_i\cdot f_{max}\cdot I_{i,j}\right)\\
&= (l-2)\cdot f_{max}\cdot\left(\sum_{i,j} f_i\cdot I_{i,j}\right)\\
&= (l-2)\cdot f_{max}\cdot\left(\sum_{i,j} \frac{F_i}{F_A+F_H}\cdot I_{i,j}\right)\\
&= (l-2)\cdot \frac{f_{max}}{F_A+F_H}\cdot\left(\sum_{i,j} F_i\cdot I_{i,j}\right)\\
&= (l-2)\cdot \frac{f_{max}}{F_A+F_H}\cdot\frac{1}{\alpha}\cdot\left(\sum_{i,j}(\alpha\cdot F_i)\cdot I_{i,j}\right)\\
&\overset{(3.3)}{\leq} (l-2)\cdot \frac{f_{max}}{F_A+F_H}\cdot\frac{1}{\alpha}\cdot\mathfrak{b}\\
&\overset{(3.1)}{\leq} (l-2)\cdot \frac{f_{max}}{F_A+F_H}\cdot\frac{1}{\alpha}\cdot(\alpha\cdot F_A\cdot(L-1)\cdot n)\\
&= (l-2)\cdot \frac{f_{max}}{F_A+F_H}\cdot(F_A\cdot(L-1)\cdot n)\\
&= (L-1)\cdot(l-2)\cdot\frac{F_A}{F_A+F_H}\cdot(f_{max}\cdot n)\\
&= (L-1)\cdot(l-2)\cdot\gamma\cdot(f_{max}\cdot n)\\
&= (L-1)\cdot(l-2)\cdot\gamma\cdot\left(\frac{F_{max}}{F_A+F_H}\cdot n\right)\\
&= (L-1)\cdot(l-2)\cdot\gamma\cdot\left(\frac{F_H}{F_H}\cdot\frac{F_{max}}{F_A+F_H}\cdot n\right)\\
&= (L-1)\cdot(l-2)\cdot\gamma\cdot\left(\frac{F_H}{F_A+F_H}\cdot\frac{F_{max}}{F_H/n}\right)\\
&= (L-1)\cdot(l-2)\cdot\gamma\cdot(1-\gamma)\cdot\delta
\end{aligned}
\tag{3.4}
$$

Finally, by a union bound, we get that the probability that the OM fails (over the choice of random path selection by Algorithm 1) is at most equal to

$$
Pr(\mathcal{E}_{drp})+Pr(\mathcal{E}_{sat})\leq\left[1-(1-\gamma)^{l-1}\right]+\left[(L-1)\cdot(l-2)\cdot\gamma\cdot(1-\gamma)\cdot\delta\right] \tag{3.5}
$$

thus

$$
\begin{aligned}
Pr(\mathcal{E}_{drp}) + Pr(\mathcal{E}_{sat}) &\leq \left[1 - (1-\gamma)^{l-1}\right] + [(L-1)\cdot(l-2)\cdot\gamma\cdot(1-\gamma)\cdot\delta] \\
&\leq [(l-1)\cdot\gamma] + [(L-1)\cdot(l-2)\cdot\gamma\cdot(1-\gamma)\cdot\delta] \\
&< [(L-1)\cdot\gamma] + [(L-1)\cdot(l-2)\cdot\gamma\cdot(1-\gamma)\cdot\delta] \\
&< [(L-1)\cdot\gamma] + [(L-1)\cdot(l-2)\cdot\gamma\cdot\delta] \\
&= (L-1)\cdot\gamma\cdot[1 + (l-2)\cdot\delta] \\
&= (L-1)\cdot\gamma\cdot[1 - \delta + (l-1)\cdot\delta] \\
&\leq (L-1)\cdot(l-1)\cdot\gamma\cdot\delta
\end{aligned}
$$

where the last inequality is because $\delta \geq 1$ as the maximum value of a set of numbers is not less than their average, hence $\delta = \frac{\max_{i=1}^{n}(F_i)}{\sum_{i=1}^{n} F_i/n} \geq 1$. ■

Sybil resistance. One of the main objectives of the proposed solution is to prevent the adversary from gaining an advantage in failing OMs by simply adding nodes with no or small funds in the network. Theorem 1 shows the proposed solution achieves this goal. As proven in the theorem, the power of adversary (in failing OMs) is at most equal to

$$(L-1)\cdot(l-1)\cdot\gamma\cdot\delta. \tag{3.6}$$

This equation is a function of the fraction of total funds in the network owned by the adversary (γ) rather than the number of nodes controlled by the adversary. The other parameters in the equation, i.e. L and l and δ, are all system parameters and are out of the adversary's control.

Impact of the travel distance. An OM can travel up to L hops, which means the parameter l in the path selection algorithm (algorithm 1) can be as large as L. By setting $l = L$ in equation (3.6), we find that the probability of OM failure increases quadratically with L. This underscores the importance of minimizing L to limit the adversary's power.

Impact of distribution of funds. Consider two networks, each with an identical number of honest nodes and the same total amount of funds. Also, assume that

the adversary possesses the same amount of funds in both networks. However, the distribution of funds among the honest nodes differs between these networks. This raises an interesting question: which network is more resilient against the maximum number of OM failures imposed by the adversary? Theorem 1 suggests that a uniform distribution of funds among honest nodes offers the highest resilience.

According to (3.6), the bound established in Theorem 3.6, both networks have identical values of γ, as the funds of the adversary and the total funds of the honest nodes are the same in each. However, the network with a uniform distribution of funds among honest nodes has $\delta = 1$, the minimum possible value by definition. In contrast, in a network with a non-uniform distribution, $\delta > 1$ because the maximum fund held by any honest node is strictly greater than the average funds of honest nodes.

To further highlight the effects of fund distribution on network resilience, the following example evaluates the adversary's power in two distinct scenarios. In the first scenario, the total funds are uniformly distributed among the honest nodes. In the second scenario, the same amount of total funds is allocated non-uniformly among the nodes.

Example 1 *Consider a network with n honest nodes and a fixed total fund F_H. Assume that the adversary's total fund equals the average fund of honest nodes, that is, $F_A = F_{avg} = F_H/n$. Let $L = l = 3$. In the first scenario, the total fund of F_H is uniformly distributed among the honest nodes, such that each node receives $F_H/n = F_{avg}$. For this scenario, we have $\gamma_1 = \frac{F_A}{F_A+F_H} = \frac{F_H/n}{F_H/n+F_H} = \frac{1}{n+1}$ and $\delta_1 = \frac{F_{max}}{F_{avg}} = 1$. Thus, by Theorem 1, the probability of failure is given by:*

$$\begin{aligned} Pr(fail_1) &= \mathcal{O}(L \cdot l \cdot \gamma_1 \cdot \delta_1) \\ &= \mathcal{O}\left(\frac{1}{n}\right). \end{aligned} \tag{3.7}$$

In the second scenario, suppose the total fund of F_H is distributed non-uniformly among honest nodes. Specifically, the first $n-1$ honest nodes receive identical funds

of f, but the n'th node receives a fund of $m \cdot f$, where $1 < m < n$ is a number. For this scenario, we get

$$\gamma_2 = \frac{F_A}{F_A + F_H} = \frac{F_H/n}{F_H/n + F_H} = \frac{1}{n+1} = \gamma_1,$$

and

$$\delta_2 = \frac{F_{max}}{F_{avg}} = \frac{m \cdot f}{\frac{m \cdot f + (n-1)f}{n}} = \frac{m \cdot n}{m + n - 1}$$

Therefore, by Theorem 1, we have:

$$\begin{aligned} Pr(fail_2) &= \mathcal{O}(L \cdot l \cdot \gamma_2 \cdot \delta_2) \\ &= \mathcal{O}(\gamma_2 \cdot \delta_2) \\ &= \mathcal{O}\left(\frac{1}{n+1} \cdot \frac{m \cdot n}{m + n + 1}\right) \\ &= \mathcal{O}\left(\frac{m}{n}\right). \end{aligned} \tag{3.8}$$

Note that $\gamma_2 = \gamma_1$, but $\delta_2 = \theta(m) \cdot \delta_1$. Thus, according to Theorem 1, the probability of failure in the second scenario is expected to be a factor of $\theta(m)$ higher than in the first. The adversary can exploit this heightened risk to significantly increase the probability of failure. To demonstrate this, consider the adversary sends OMs at a rate of $\alpha \cdot F_A = \alpha \cdot F_{avg}$ to each honest node U_i, where $1 \leq i < n$, and directs them to forward these OMs to node U_n, effectively saturating all links to U_n.

Assume two honest nodes, U_s and U_d, with U_s using the routing algorithm (Algorithm 1) to select a random path $P = (U_s, U_i, U_j, U_d)$ of length $l = 3$ to send an OM to U_d through random nodes U_i and U_j. The probability that the link (U_i, U_j) is saturated equals the probability that $i < n$ and $j = n$, which is equal to

$$\frac{(n-1)F_{avg}}{n \cdot F_{avg}} \cdot \frac{F_{avg}}{n \cdot F_{avg}} = \frac{n-1}{n^2} = \theta\left(\frac{1}{n}\right),$$

in the first scenario, while the equivalent probability in the second scenario is

$$\frac{(n-1) \cdot f}{(n-1) \cdot f + m \cdot f} \cdot \frac{m \cdot f}{(n-1) \cdot f + m \cdot f} = \frac{(n-1) \cdot m}{(n+m-1)^2} = \theta\left(\frac{m}{n}\right).$$

These probabilities are asymptotically optimal as they match the upper bound established by Theorem 1, with the latter being a factor of $\theta(m)$ larger than the former, illustrating the significant impact of a non-uniform fund distribution.

A tighter bound. We can further tighten the bound established in Theorem 1. The idea to do this is by making the second inequality in (3.4) tighter. By doing so, Theorem 2 replaces $\delta = \frac{F_{max}}{F_{avg}}$ in Example 1 with $\check{\delta} = \frac{F_{max}^{(k)}}{F_{avg}}$, where $F_{max}^{(k)}$ is the average funds of the top k, $k = 1 + \lfloor \frac{(L-1)F_A}{F_{avg}} \rfloor$ honest nodes. Since clearly $F_{max}^{(k)} \leq F_{max}$, we get that $\check{\delta} \leq \delta$, hence we get a tighter upper bound in Theorem 2.

Theorem 2 *Let L denote the maximum number of hops an OM can travel, and $l \leq L$ denote the OM path-length parameter used in the path-selection algorithm (algorithm 1). Let F_A be the total funds of the adversary, and $F_H = \sum_{i=1}^{n} F_i$ be the total fund of honest nodes, where F_i denotes the fund of honest node U_i, and n is the total number of honest nodes in the network.*

Then, for every pair of distinct honest nodes U_s and U_d, the probability that an OM message, sent on a random path chosen by algorithm 1, from U_s to U_d fails is at most

$$\begin{aligned} 1 - (1-\gamma)^{l-1} + (L-1) \cdot (l-2) \cdot \gamma \cdot \check{\delta} \\ \leq (L-1) \cdot (l-1) \cdot \check{\delta} \cdot \gamma \end{aligned}$$

where

$$\gamma = \frac{F_A}{F_A + F_H}$$

is the fraction of funds in the entire network that belongs to the adversary, and

$$\check{\delta} = \frac{F_{max}^{(k)}}{F_{avg}}$$

is the ratio of the average fund held by the top k honest nodes to the average fund across all honest nodes in the network, and $k = 1 + \lfloor \frac{(L-1)F_A}{F_{avg}} \rfloor$.

Proof. Replacing $\mathfrak{b}$ in (3.3) with the right-hand side term in 3.1 we get

$$\sum_{i,j} (\alpha \cdot F_i) \cdot I_{i,j} \leq \alpha \cdot F_A \cdot (L-1) \cdot n,$$

thus

$$\sum_{i,j} F_i \cdot I_{i,j} \leq F_A \cdot (L-1) \cdot n$$

Without loss of generality, we assume $f_1 \geq f_2 \geq \ldots \geq f_n$. Similarly to the proof of Theorem 1, we have

$$Pr(\mathcal{E}_{sat}) \leq (l-2) \cdot \left(\sum_{i,j} f_i \cdot f_j \cdot I_{i,j} \right).$$

Starting with this inequality, we get

$$\begin{aligned}
Pr(\mathcal{E}_{sat}) &\leq (l-2) \cdot \left(\sum_{i,j} f_i \cdot f_j \cdot I_{i,j} \right) \\
&\leq (l-2) \cdot \left(\frac{\mathfrak{b}}{\alpha \cdot F_H} \cdot f_1 + \frac{\mathfrak{b}}{\alpha \cdot F_H} \cdot f_2 + \ldots + \frac{\mathfrak{b}}{\alpha \cdot F_H} \cdot f_k \right) \\
&\leq (l-2) \cdot \frac{\mathfrak{b}}{\alpha \cdot F_H} \cdot (f_1 + f_2 + \ldots + f_k) \\
&= (l-2) \cdot \frac{\mathfrak{b}}{\alpha \cdot F_H} \cdot \frac{\sum_{i=1}^{k} F_i}{F_A + F_H} \\
&= (l-2) \cdot \frac{\mathfrak{b}}{\alpha \cdot (F_A + F_H)} \cdot \frac{\sum_{i=1}^{k} F_i}{F_H} \\
&\leq (l-2) \cdot \frac{\alpha \cdot F_A \cdot (L-1) \cdot n}{\alpha \cdot (F_A + F_H)} \cdot \frac{\sum_{i=1}^{k} F_i}{F_H} \\
&= (L-1) \cdot (l-2) \cdot \frac{F_A}{F_A + F_H} \cdot \frac{\sum_{i=1}^{k} F_i}{F_H/n} \\
&= (L-1) \cdot (l-2) \cdot \gamma \cdot \check{\delta}
\end{aligned} \tag{3.9}$$

Next, similar to the proof of Theorem 1, we get

$$\begin{aligned}
Pr(\mathcal{E}_{drp}) + Pr(\mathcal{E}_{sat}) &\leq \left[1 - (1-\gamma)^{l-1}\right] + \left[(L-1) \cdot (l-2) \cdot \gamma \cdot \check{\delta}\right] \\
&\leq \left[(l-1) \cdot \gamma\right] + \left[(L-1) \cdot (l-2) \cdot \gamma \cdot \check{\delta}\right] \\
&< \left[(L-1) \cdot \gamma\right] + \left[(L-1) \cdot (l-2) \cdot \gamma \cdot \check{\delta}\right] \\
&= (L-1) \cdot \gamma \cdot \left[1 + (l-2) \cdot \check{\delta}\right] \\
&= (L-1) \cdot \gamma \cdot \left[1 - \check{\delta} + (l-1) \cdot \check{\delta}\right] \\
&\leq (L-1) \cdot (l-1) \cdot \gamma \cdot \check{\delta}
\end{aligned}$$

where the last inequality is because $\check{\delta} \geq 1$ as the average funds of top honest nodes is not less than the average fund of all honest nodes. ■

The adversary's optimal strategy. Suppose the adversary's strategy is to maximize the probability that the h-th link of the path between the source and the destination, where $1 < h < l$, is saturated. The probability that the h-th link of the path is between nodes U_i and U_j is $f_i \cdot f_j$. The adversary's cost to saturate this link is $\alpha \cdot F_i = c \cdot f_i$, where $c = \alpha \cdot (F_A + F_H)$ is a constant. Therefore, informally speaking, to gain a probability $f_i \cdot f_j$, the adversary needs to spend a cost of $c \cdot f_i$ from its budget $\mathfrak{b}$. This suggests that the adversary should target U_j with the maximum value of f_j (i.e., the node with the maximum fund) to maximize the gain $f_i \cdot f_j$. If the adversary has already saturated all the links to the node with the maximum fund and still has a remaining budget, it should aim to saturate the links to the next node with the highest fund, and so on. The above suggests that to maximize the probability of OM failure, the adversary should aim at saturating the links to the honest nodes with top funds. This intuition is used in Theorem 2 to improve the bound in Theorem 1.

Partial communication graph. In line with the BOLT standards, we have assumed that the communication graph G_C is a complete graph, meaning that any two nodes in the network can establish a communication link to exchange OMs. An interesting question is whether the bound of Theorem 1 on the network's resilience can be improved if nodes restrict themselves to an incomplete/partial communication graph. We answer this positively for networks where funds are distributed non-uniformly between nodes, that is, in networks with $\delta > 1$. This is positive news, as in practice, δ can be quite large. For instance, as of May 20, 2024, $\delta = 446$ in LN.

Next, we demonstrate a method based on partial communication graphs, which can significantly improve the resilience of networks such as LN where δ is large. In this method, the source node restricts the proposed path selection algorithm (Algorithm 1) to a subset S of nodes when constructing a path to the destination. The fund of each node in the subset, however, remains equal to the node's funds in the entire network, not just within the subset S. The following theorem establishes a bound on the probability of OM failure in this method, expressed as a function of the total and

maximum funds within the subset S.

Theorem 3 *Suppose we restrict the selection of relay nodes in the path-selection algorithm (Algorithm 1) to a subset S of nodes within the network. Then, for every pair of distinct honest nodes U_s and U_d, the probability that an OM message, sent along a randomly chosen path by Algorithm 1 from U_s to U_d, fails is at most*

$$(l-1)\cdot F_A\cdot\left(\frac{1}{F_{tot}^{(s)}}+(L-1)\cdot\frac{|S|\cdot F_{max}^{(s)}}{\left(F_{tot}^{(s)}\right)^2}\right) \tag{3.10}$$

where F_A represents the total funds of the adversary in the entire network, $F_{max}^{(s)}$ is the maximum fund held by any single node in the subset S, and $F_{tot}^{(s)}$ is the total fund of all nodes within S.

Proof. Let P be the random path selected by Algorithm 1 when relay nodes are restricted to the subset S. The OM message sent on P fails if and only if at least one of the following events occurs: 1) Event $\mathcal{E}_{drp}$: the path P includes an adversarial node, which consequently drops the message. 2) Event $\mathcal{E}_{sat}$: the path P features an honest link that is saturated by the adversary, leading to the OM message being dropped by the receiver of that link.

To determine the random path P, Algorithm 1 selects $(l-1)$ nodes at random across $(l-1)$ rounds. In each round, the probability of selecting a specific node is proportional to its fund. Consequently, the probability $Pr(\mathcal{E}_{drp})$, that Algorithm 1 selects an adversarial node, is given by

$$\begin{aligned} Pr(\mathcal{E}_{drp}) &= 1-\left(1-\frac{F_A'}{F_{tot}^{(s)}}\right)^{l-1} \\ &\leq (l-1)\cdot\frac{F_A}{F_{tot}^{(s)}}, \end{aligned} \tag{3.11}$$

where $F_A'\leq F_A$ represents the total fund of the adversary's nodes within the subset S.

Let $F_1, F_2, \ldots, F_h$ denote the funds of honest nodes $U_1, U_2, \ldots, U_h$ in S, where h denotes the number of honest nodes in S. Note that only the nodes in S forward OMs. The adversary is capable of establishing up to h connections with these honest nodes, each connection pushing a rate of at most $\alpha \cdot F_A$. Furthermore, each connection can engage up to $(L-1)$ honest links. Consequently, the total rate that the adversary can impose on links between honest nodes in S, denoted by $\mathfrak{b}^{(s)}$, is bounded by

$$\begin{aligned} \mathfrak{b}^{(s)} &\leq \alpha \cdot F_A \cdot (L-1) \cdot h \\ &\leq \alpha \cdot F_A \cdot (L-1) \cdot |S| \end{aligned} \tag{3.12}$$

The probability that the kth link, $1 < k < l$, on the path P is a saturated honest link is

$$\sum_{1 \leq i,j \leq h} f_i \cdot f_j \cdot I_{i,j}, \tag{3.13}$$

where $f_i = \frac{F_i}{F_{tot}^{(s)}}$, $f_j = \frac{F_j}{F_{tot}^{(s)}}$, $f_i \cdot f_j$ is the probability that the kth link, $1 < k < l$, is (U_i, U_j), and $I_{i,j}$ is a binary variable equal to one if the adversary has saturated (U_i, U_j), the link from U_i to U_j; and equal to zero otherwise. To set $I_{i,j}$ to one, the adversary needs to push OMs at a rate of at least αF_i from U_i to U_j. Given the adversary's maximum budget is $\mathfrak{b}^{(s)}$, we must then have

$$\sum_{1 \leq i,j \leq h} (\alpha \cdot F_i) \cdot I_{i,j} \leq \mathfrak{b}^{(s)}. \tag{3.14}$$

Thus, we get

$$
\begin{aligned}
Pr(\mathcal{E}_{sat}) &\overset{by(3.13)\text{ and union bound}}{\leq} (l-2)\cdot\left(\sum_{1\leq i,j\leq h} f_i\cdot f_j\cdot I_{i,j}\right)\\
&\overset{f^{(s)}_{max}=\max_{U_i\in S}(f_i)}{\leq} (l-2)\cdot\left(\sum_{1\leq i,j\leq h} f_i\cdot f^{(s)}_{max}\cdot I_{i,j}\right)\\
&= (l-2)\cdot f^{(s)}_{max}\cdot\left(\sum_{1\leq i,j\leq h} f_i\cdot I_{i,j}\right)\\
&= (l-2)\cdot f^{(s)}_{max}\cdot\left(\sum_{1\leq i,j\leq h} \frac{F_i}{F^{(s)}_{tot}}\cdot I_{i,j}\right)\\
&= (l-2)\cdot \frac{f^{(s)}_{max}}{F^{(s)}_{tot}}\cdot\left(\sum_{1\leq i,j\leq h} F_i\cdot I_{i,j}\right)\\
&= (l-2)\cdot \frac{f^{(s)}_{max}}{F^{(s)}_{tot}}\cdot\frac{1}{\alpha}\cdot\left(\sum_{1\leq i,j\leq h} (\alpha\cdot F_i)\cdot I_{i,j}\right) \qquad (3.15)\\
&\overset{(3.14)}{\leq} (l-2)\cdot \frac{f^{(s)}_{max}}{F^{(s)}_{tot}}\cdot\frac{1}{\alpha}\cdot\mathfrak{b}^{(s)}\\
&\overset{(3.12)}{\leq} (l-2)\cdot \frac{f^{(s)}_{max}}{F^{(s)}_{tot}}\cdot\frac{1}{\alpha}\cdot(\alpha\cdot F_A\cdot(L-1)\cdot|S|)\\
&= (l-2)\cdot \frac{f^{(s)}_{max}}{F^{(s)}_{tot}}\cdot(F_A\cdot(L-1)\cdot|S|)\\
&= (L-1)\cdot(l-2)\cdot\frac{F_A}{F^{(s)}_{tot}}\cdot\left(\frac{F^{(s)}_{max}\cdot|S|}{F^{(s)}_{tot}}\right)\\
&= (L-1)\cdot(l-2)\cdot\frac{|S|\cdot F_A\cdot F^{(s)}_{max}}{\left(F^{(s)}_{tot}\right)^2}.
\end{aligned}
$$

Therefore, by (3.15), (3.11) and a union bound, we get that the probability that the OM fails (over the choice of random path selection by Algorithm 1) is at most

equal to

$$
\begin{aligned}
Pr(\mathcal{E}_{drp}) + Pr(\mathcal{E}_{sat}) &\leq \left((l-1) \cdot \frac{F_A}{F_{tot}^{(s)}} \right) + \left((L-1) \cdot (l-2) \cdot \frac{|S| \cdot F_A \cdot F_{max}^{(s)}}{\left(F_{tot}^{(s)}\right)^2} \right) \\
&\leq \left((l-1) \cdot \frac{F_A}{F_{tot}^{(s)}} \right) + \left((L-1) \cdot (l-1) \cdot \frac{|S| \cdot F_A \cdot F_{max}^{(s)}}{\left(F_{tot}^{(s)}\right)^2} \right) \\
&\leq (l-1) \cdot F_A \cdot \left(\frac{1}{F_{tot}^{(s)}} + (L-1) \cdot \frac{|S| \cdot F_{max}^{(s)}}{\left(F_{tot}^{(s)}\right)^2} \right)
\end{aligned}
\tag{3.16}
$$

■

We define a subset $S \subseteq V$ as optimum if it minimizes the upper bound (3.10) of Theorem 3. Note that the leading coefficients $(l-1)$ and F_A in (3.10) are independent of the choice of the set S. Thus, a set S is optimum if it is a member of

$$
\mathcal{S} = \underset{S \subseteq V}{\arg\min} \left(\frac{1}{F_{tot}^{(s)}} + (L-1) \cdot \frac{|S| \cdot F_{max}^{(s)}}{\left(F_{tot}^{(s)}\right)^2} \right) \tag{3.17}
$$

Although the set V has an exponential number of subsets S, Theorem 5 demonstrates that a set in $\mathcal{S}$ can be found in quadratic time. We will use this set in the simulation setting to show that LN can improve its resilience against the adversary by using a partial communication graph.

Lemma 4 *Consider a network with m nodes $u_1, u_2, \ldots, u_m$. Let F_i denote the funds of node U_i in the network. Suppose $F_1 \leq F_2 \leq \ldots \leq F_m$.*
Then there exists a set $S \in \mathcal{S}$ such that $S = \{U_k | k \in [i, j]\}$ for some integers $1 \leq i \leq j \leq m$.

Proof. Let us define the *spread* of a set as the greatest difference between the subscript indices of any two nodes in the set. For instance, the spread of the set $\{U_2, U_5, U_7, U_8\}$ is $8 - 2 = 6$. Let S be a set in $\mathcal{S}$ with the minimum spread. We show that $S = \{U_k | k \in [i, j]\}$ for some integers $1 \leq i \leq j \leq m$. Let U_i and U_j

be the nodes in S with the maximum difference between their indices. Towards a contradiction, suppose that there is an integer $i < k < j$ such that $U_k \notin S$. Let $S^\dagger = (S \cup \{U_k\}) \setminus \{U_i\}$, that is $S^\dagger$ is formed from S by adding U_k to and removing U_i from S. We have $|S^\dagger| = |S|$, $F_{max}^{(s^\dagger)} = F_{max}^{(s)}$ and $F_{tot}^{(s)} \leq F_{tot}^{(s^\dagger)}$ as $F_i \leq F_k$. Therefore, by (3.17), we get

$$S \in \mathcal{S} \Longrightarrow S^\dagger \in \mathcal{S}.$$

This is a contradiction since $S^\dagger$ has a strictly smaller spread than S, but S was assumed to have the smallest spread among the sets in $\mathcal{S}$.

■

Theorem 5 *There exists a quadratic-time algorithm (algorithm 2) for finding a set in $\mathcal{S}$.*

Proof. By Lemma 4, there is a set $S \in \mathcal{S}$ such that $S = \{U_k | k \in [i, j]\}$ for some integers $1 \leq i \leq j \leq n$. There are quadratically many sets of this form. Using dynamic programming as described in Algorithm 2, one that belongs to $\mathcal{S}$ can be identified in quadratic time. ■

Algorithm 2: Finds subset S to minimize Equation 3.17.

Input: $funds$ `// List of funds of honest nodes`
Output: S `// Subset of honest nodes that minimizes` $\delta \cdot \gamma$

Sort $funds$ in non-decreasing order;
$L \leftarrow$ maximum OM travel distance;
$n \leftarrow$ length of $funds$;
$minValue \leftarrow \infty$;
$bestSubset \leftarrow \{\}$;
for $start \leftarrow 0$ **to** $n - 1$ **do**
    $sMax \leftarrow funds[start]$;
    $sSum \leftarrow funds[start]$;
    $sLen \leftarrow 1$;
    for $end \leftarrow start + 1$ **to** n **do**
        $sSum \leftarrow sSum + funds[end]$;
        $sLen \leftarrow sLen + 1$;
        if $sMax < funds[end]$ **then**
            $sMax \leftarrow funds[end]$;
        end if
        $currentValue \leftarrow 1/sSum + (L - 1) * sLen * sMax/(sSum)^2$;
        if $currentValue < minValue$ **then**
            $minValue \leftarrow currentValue$;
            $bestSubset \leftarrow funds[start : end]$;
        end if
    end for
end for
return $bestSubset$;

3.3.2 Targeting a Victim Node

In this attack scenario, the adversary aims to prevent nodes from reaching a victim node W. We demonstrate that the probability of any honest node U's message to W being dropped before reaching W is bounded by the limits specified in Theorem 1 and Theorem 2.

Let P denote the random path selected by Algorithm 1. As in the previous analysis, the OM sent on P fails if at least one of the following events occurs: 1) Event $\mathcal{E}_{drp}$: The path P contains an adversarial node, which will drop the message; 2) Event $\mathcal{E}_{sat}$: The path P contains an honest link saturated by the adversary, leading to the OM being dropped by the link's receiver.

The probability that algorithm 1 selects a path containing an adversarial node is given by

$$Pr(\mathcal{E}_{drp}) = 1 - (1 - \gamma)^{l-1},$$

consistent with previous results.

To analyze $Pr(\mathcal{E}_{sat})$, note that according to the forwarding rules, if the last node on P is honest, then the link from this node to W will never drop a message destined for W, even if the link is saturated. Moreover, all nodes on path P are selected by Algorithm 1 independently of the destination node W. Therefore, $Pr(\mathcal{E}_{sat})$ is independent of the destination node choice. This implies that in analyzing $Pr(\mathcal{E}_{sat})$, we can consider W to be any node in the network, and hence, the analysis of $Pr(\mathcal{E}_{sat})$ mirrors those presented in Theorem 1 and Theorem 2.

3.4 Privacy

The content of OMs is encrypted, ensuring that only the intended destination can decrypt the message. An interesting question is whether a node on path P selected by Algorithm 1 may gain information about the source or the designation node.

We prove that the middle nodes on path P, i.e., nodes other than the first and last nodes on P, gain no information about the source or the destination. The intuition is simple: Algorithm 1 selects path P independent of the source and the destination. After selecting P, the source sends the message to the first node on P, and the last node on P is set to forward the message to the destination. Therefore, the path P reveals no information about the source or the destination. Consequently, the middle nodes on P, which in the best case know the entire P, gain no information about the source and the destination. The next theorem states this fact formally.

Theorem 6 *Consider a network and let X be the random variable representing the node that is the source of an OM sent over the network. Let P be the random variable*

representing the random path chosen by Algorithm 1. Then

$$I(X;P) = 0,$$

i.e., knowing the random path P provides no information about X.

Proof. algorithm 1 is designed to choose the path P independently of the source and the destination nodes. Specifically, algorithm 1 does not use the source or the destination nodes as inputs in its decision-making process. This design ensures that the choice of P is not influenced by any information about the identity of the source node X.

Given this independence, the conditional probability of P given X is the same as the unconditional probability of P:

$$\Pr(P \mid X) = \Pr(P).$$

Consequently, the conditional entropy $H(P \mid X)$ equals the entropy $H(P)$, implying that:

$$I(X;P) = H(P) - H(P \mid X) = 0,$$

which signifies that X and P are independent. Therefore, knowing the path P provides no information about the source node X. ■

The probability that the first node on P belongs to the adversary is

$$\frac{F_A}{F_A + F_H}.$$

Similarly, the last node on P belongs to the adversary with the same probability. Since Algorithm 1 selects the nodes on P independently at random (with replacement), the probability that both the first and the last nodes on path P are adversarial is

$$\left(\frac{F_A}{F_A + F_H}\right)^2 \tag{3.18}$$

By Theorem 6, the middle nodes on path P gain no information about the source or the destination, hence the probability that the adversary discovers both the source and the destination is bounded by (3.18).

3.5 Simulations

We leverage simulations to understand the extent of mitigation our solution provides. The simulations were executed on a graph constructed from an exported list of public nodes and channels from the real-world LN graph on May 20, 2024. At that time, the network comprised approximately 15,120 nodes and 51,579 public channels, containing about 3,268.82 bitcoins. With the bitcoin price of around 69,000 USD per bitcoin, the total value in the network was approximately 225,548,580 USD.

Forwarding onion messages does not require nodes to have a channel between them, and they can be sent just like gossip messages, as currently implemented. We assume a complete graph of nodes in our simulation because, unlike channel creation, creating a peer connection is cheap. We should also keep in mind that links are bidirectional, and although a link might get saturated in one direction, it might still be able to forward OMs in the other direction.

Our simulation imports all the nodes and channels, along with their capacities, from the exported file into a NetworkX graph. We then insert an adversary node with a certain amount of total funds (F_A) into the network. We assume the adversary does not have a computation limitation for creating messages, so it will send as many messages as its neighbor's rate limits allow it to. It then forwards those messages to the neighbors of its neighbors, asking them to further forward the messages as far as L hops, which is the number of maximum hops that our distance-limiting algorithm allows it to. In our simulations, we assume $L = 3$, meaning that the maximum distance an OM can travel is three.

To get an upper bound of the adversary's effectiveness, we assume the adversary to be a **strong adversary**. We define a strong adversary as one that can saturate any given set of links in the network up to a budget $\mathfrak{b}$. It is important to note that a feasible adversary may not be strong, as the set of links it can saturate must form a set of paths originating from itself (the paths the adversary uses to flood the network).

However, in our simulations, we use a strong adversary instead of a feasible one to evaluate the extent of the damage (OM failures) that any feasible adversary could potentially cause.

We then proceed with executing the adversary's channel saturation process. We experiment with two saturation algorithms that an adversary can possibly choose to apply. In the first one, which we call the **test adversary**, the adversary starts using its budget $\mathfrak{b}$, saturating channels at random. In the second one, we consider the **optimal adversary** described in Section 3.3.1, which starts by saturating the links to the highest fund node and then progresses to the next highest fund node.

After the saturation phase, many of the links in the network would become saturated by the adversary, and the forwarding requests of honest users on these links would be unable to be successfully executed. Then, we start the second phase of the simulation, which is sending honest onion messages and measuring their failure rate.

In the second phase, since we know that the first link from the source and the last link that goes to the destination are both always available, we only choose the intermediary hops, and then if any of the links in the chosen path are saturated, we consider it a failed attempt otherwise we count it as a successful attempt.

The simulation results are plotted in Figure 3.3. This plot demonstrates how the optimal adversary performs much better than the test adversary. We can also see that the upper bound presented in Theorem 2 follows the optimal adversary much closer than Theorem 1's upper bound.

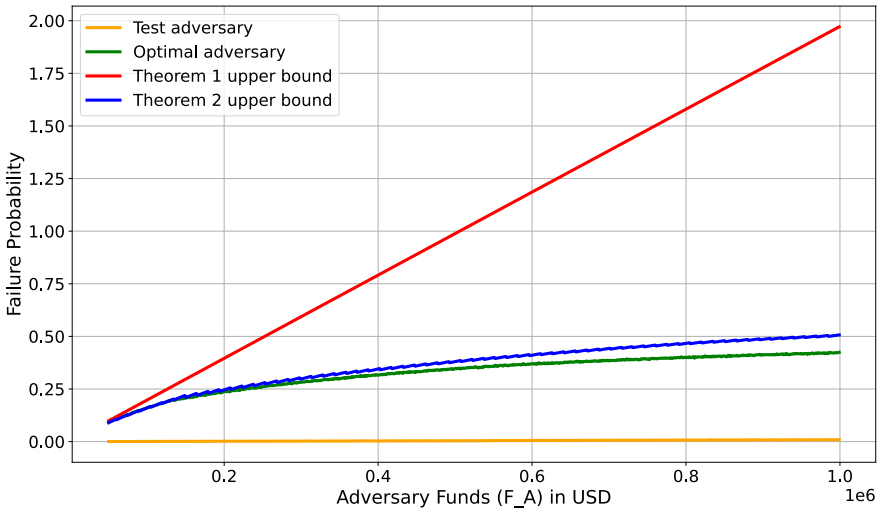

Figure 3.3: Optimal and test adversaries failure rates compared with the upper bounds in Theorems 1 and 2.

To better be able to see the details, Figure 3.4 contains the same data without the Theorem 1's upper bound. This chart shows how closely the upper bound in Theorem 2 follows the failure rate that an optimal adversary can cause. We can also see that the failure rate for the test adversary is not zero but rather very small. Most of the test adversary's failures are because of the adversary node being on the path of transactions rather than failure because of a saturated link.

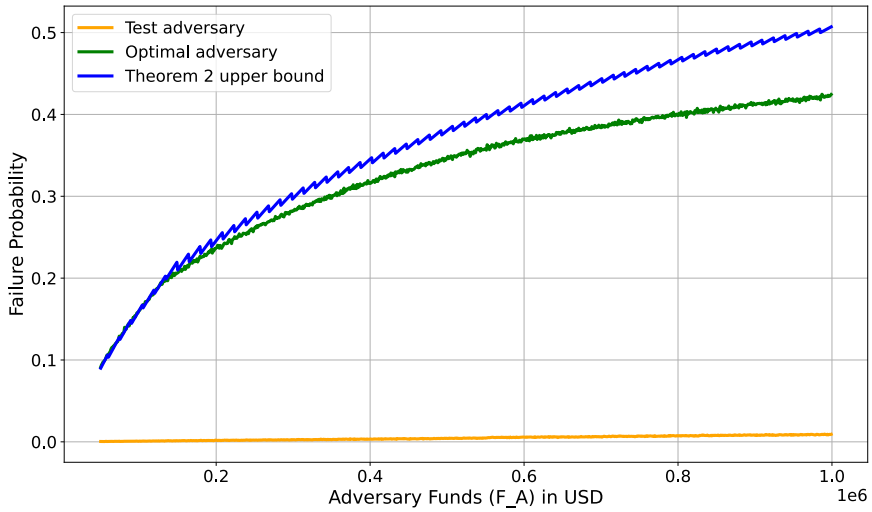

Figure 3.4: Optimal and test adversaries failure rates compared with the upper bound in Theorem 2.

To further analyze the plot in Figure 3.4, consider an adversary with a fund of 100,000 USD. If the adversary begins to saturate links at random, it will generate a 0.001 probability of honest OM failure across the network, which does not significantly impact the experience of honest users. However, if the adversary employs an optimal strategy, they can increase the failure probability to 0.15 for honest OMs. Importantly, no adversary with a fund of 100,000 USD can exceed the failure probability of approximately 0.15, as established by the upper bound derived in Theorem 2. Thus, the optimal strategy of the adversary almost approaches this theoretical upper bound.

Now, consider an adversary with a larger fund, such as 1 million USD. If the adversary randomly saturates links, it would induce a 0.009 failure probability on honest OMs, a relatively insignificant impact. However, by employing an optimal strategy, the adversary can escalate the failure probability to approximately 0.42, which is notably more effective and is closer to the theoretical upper bound of 0.5, as established by Theorem 2.

We conducted the same experiments on the partial communication graph to com-

pare the results with the upper bound calculated in Theorem 3. Algorithm 2 was applied to LN to identify the optimal subset that minimizes the upper bound specified in Theorem 3. This subset contains $|S| = 26$ nodes, with a total fund of $F_{tot}^{(S)} = 153{,}660{,}300$ USD and a maximum individual fund of $F_{max}^{(S)} = 13{,}305{,}211$ USD. Since honest nodes derive this subset from publicly available information, the adversary can also determine the subset. In both Theorem 3 and the simulation, we assume the adversary is aware of the subset that honest nodes will select and will create nodes that integrate into this chosen subset. The adversary node is added to this subset, and the simulation is then executed. The results are shown in Figure 3.5.

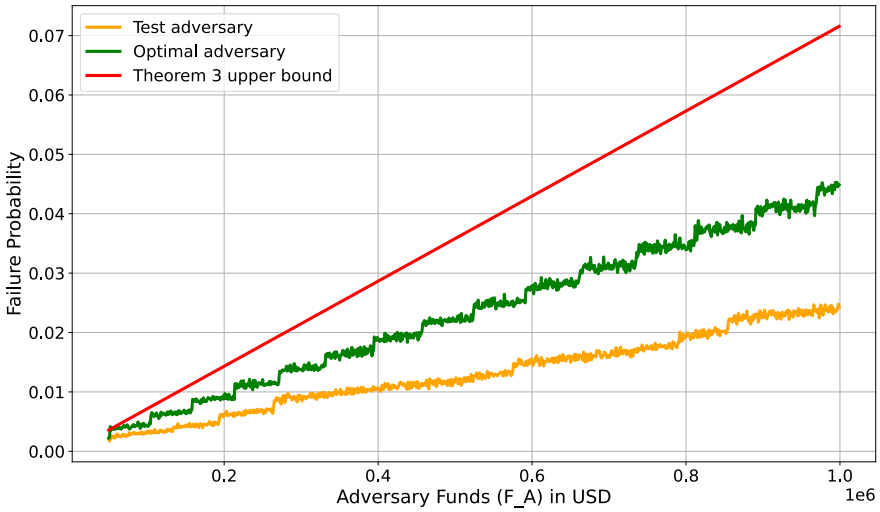

Figure 3.5: Optimal and test adversaries failure rates in the subset compared with the upper bound in Theorem 3.

We observe that the difference between the failure rates caused by the optimal adversary and the test adversary is much smaller in this subset (Figure 3.5) compared to the whole network (Figure 3.4). This smaller difference is attributed to the fact that the value of δ for this subset is approximately 2.25, which is significantly lower than the 446 calculated for the entire network. A smaller δ indicates a more uniform distribution of node funds within this subset, reducing the impact of the saturation algorithm. As a result, whether links are saturated randomly or with an optimal

strategy makes little difference.

Comparing Figure 3.5 with Figure 3.4, we observe that restricting the communication graph to the selected subset has notably reduced the optimal adversary's ability to disrupt the OM service. The results indicate that even an optimal adversary with a budget of one million dollars can only achieve a 4.5% failure rate in the OM service, with the upper bound, as calculated by Theorem 3, being approximately 7.1%. This upper bound of 7.1% for the subset is significantly lower than the 50% failure rate derived from Theorem 2, as illustrated in Figure 3.4.

When interpreting these results, it's important to remember that honest OMs can be reattempted. For instance, in the presence of an optimal adversary with a fund of 1 million USD, a single attempt to send an honest OM has a success probability of approximately 0.58 on the whole network as shown in Figure 3.4. Adding a second attempt increases the success probability to $1 - (0.42^2) = 0.8236$. Similarly, with three, four, and five attempts, the success probability rises to 0.9259, 0.9689, and 0.9869, respectively. This demonstrates that with our proposed mitigation method, even in the presence of an optimal adversary with 1 million USD in funds, honest nodes can achieve a success probability of over 92% with just three attempts. This is particularly significant given that onion messages are designed as an unreliable, lightweight communication method [12].

Chapter 4

Conclusion and Future Work

This research investigated potential DoS threats to LN leveraging the newly introduced OM service. We presented two possible attack scenarios: one aimed at degrading the overall OM service availability and another targeting the reachability of a specific node.

To mitigate these attacks, we proposed a method composed of two mechanisms. The first involves reducing the OM travel distance by implementing a hard or soft leash. The second enforces a rate limit on nodes, where this limit is suggested to be proportional to the funds invested by each node to increase the cost of the DoS attacks. We demonstrated the efficacy of our proposed solution in mitigating the attacks. Implementing these mechanisms leads to fewer messages traveling shorter distances, establishing an effective prevention/mitigation strategy against DoS threats. Finally, we demonstrated the effectiveness of our proposed solution in mitigating the attacks.

While our proposed solution provides a robust defense against DoS threats in Lightning Network's onion messaging service, testing the solution in real-world scenarios, including various adversarial models and network conditions, would be valuable. Such experiments would provide insights into potential edge cases and help refine the proposed mechanisms for optimal performance and security.

Bibliography

[1] A. Bashiri and M. Khabbazian, "Short paper: Onion messages on leash," *Financial Cryptography and Data Security - 28th International Conference*, 2024.

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized business review*, p. 21 260, 2008.

[3] G. Karame, "On the security and scalability of bitcoin's blockchain," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 1861–1862.

[4] A. Chauhan, O. P. Malviya, M. Verma, and T. S. Mor, "Blockchain and scalability," in *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, IEEE, 2018, pp. 122–128.

[5] A. Hafid, A. S. Hafid, and M. Samih, "Scaling blockchains: A comprehensive survey," *IEEE access*, vol. 8, pp. 125 244–125 262, 2020.

[6] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, and Y. Liu, "A survey on the scalability of blockchain systems," *IEEE network*, vol. 33, no. 5, pp. 166–173, 2019.

[7] J. Poon and T. Dryja, *The bitcoin lightning network: Scalable off-chain instant payments*, 2016.

[8] C. Decker and R. Wattenhofer, "A fast and scalable payment network with bitcoin duplex micropayment channels," in *Stabilization, Safety, and Security of Distributed Systems: 17th International Symposium, SSS 2015, Edmonton, AB, Canada, August 18-21, 2015, Proceedings 17*, Springer, 2015, pp. 3–18.

[9] N. Papadis and L. Tassiulas, "Blockchain-based payment channel networks: Challenges and recent advances," *IEEE Access*, vol. 8, pp. 227 596–227 609, 2020.

[10] G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, and S. Ravi, "Concurrency and privacy with payment-channel networks," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 455–471.

[11] "Lnurl documents." (), [Online]. Available: https://github.com/lnurl/luds (visited on 08/15/2024).

[12] "Bolt 4: Onion messages." (), [Online]. Available: https://github.com/lightning/bolts/blob/master/04-onion-routing.md#onion-messages (visited on 08/15/2024).

[13] D. Goldschlag, M. Reed, and P. Syverson, "Onion routing," *Communications of the ACM*, vol. 42, no. 2, pp. 39–41, 1999.

[14] N. Long and R. Thomas, "Trends in denial of service attack technology," *CERT Coordination Center*, vol. 648, no. 651, p. 569, 2001.

[15] B. B. Gupta, R. C. Joshi, and M. Misra, "Distributed denial of service prevention techniques," *arXiv preprint arXiv:1208.3557*, 2012.

[16] P. Koshy, D. Koshy, and P. McDaniel, "An analysis of anonymity in bitcoin using p2p network traffic," in *Financial Cryptography and Data Security: 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers 18*, Springer, 2014, pp. 469–485.

[17] J. Herrera-Joancomartí, "Research and challenges on bitcoin anonymity," in *International Workshop on Data Privacy Management*, Springer, 2014, pp. 3–16.

[18] M. C. K. Khalilov and A. Levi, "A survey on anonymity and privacy in bitcoin-like digital cash systems," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2543–2585, 2018.

[19] N. Tovanich and R. Cazabet, "Fingerprinting bitcoin entities using money flow representation learning," *Applied Network Science*, vol. 8, no. 1, p. 63, 2023.

[20] "[lightning-dev] dual funding proposal." (), [Online]. Available: https://lists.linuxfoundation.org/pipermail/lightning-dev/2018-November/001682.html (visited on 08/15/2024).

[21] "Bolt 7: P2p node and channel discovery." (), [Online]. Available: https://github.com/lightning/bolts/blob/master/07-routing-gossip.md (visited on 08/15/2024).

[22] F. Lau, S. H. Rubin, M. H. Smith, and L. Trajkovic, "Distributed denial of service attacks," in *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0*, IEEE, vol. 3, 2000, pp. 2275–2280.

[23] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation onion router," in *13th USENIX Security Symposium (USENIX Security 04)*, San Diego, CA: USENIX Association, Aug. 2004. [Online]. Available: https://www.usenix.org/conference/13th-usenix-security-symposium/tor-second-generation-onion-router.

[24] "Tor blog: Tor is slow right now. here is what is happening." (), [Online]. Available: https://blog.torproject.org/tor-network-ddos-attack/ (visited on 08/15/2024).

[25] R. Jansen, T. Vaidya, and M. Sherr, "Point break: A study of bandwidth Denial-of-Service attacks against tor," in *28th USENIX Security Symposium (USENIX Security 19)*, Santa Clara, CA: USENIX Association, Aug. 2019, pp. 1823–1840, ISBN: 978-1-939133-06-9. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/jansen.

[26] "Tor blog: Introducing proof-of-work defense for onion services." (), [Online]. Available: https://blog.torproject.org/introducing-proof-of-work-defense-for-onion-services/ (visited on 08/15/2024).

[27] "Tor-dev mailing list: A first take at pow over introduction circuits." (), [Online]. Available: https://lists.torproject.org/pipermail/tor-dev/2020-June/014381.html (visited on 08/15/2024).

[28] F. Chen and J. Pasquale, "Toward improving path selection in tor," in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, IEEE, 2010, pp. 1–6.

[29] "Lightning-dev mailing list: Onion messages rate-limiting." (), [Online]. Available: https://lists.linuxfoundation.org/pipermail/lightning-dev/2022-June/003623.html (visited on 08/15/2024).

[30] "Bolt 4: Onion routing protocol." (), [Online]. Available: https://github.com/lightning/bolts/blob/master/04-onion-routing.md (visited on 08/15/2024).

[31] K. Chatterjee, A. K. Goharshady, and A. Pourdamghani, "Probabilistic smart contracts: Secure randomness on the blockchain," in *2019 IEEE international conference on blockchain and cryptocurrency (ICBC)*, IEEE, 2019, pp. 403–412.

[32] P. Christodoulou, G. C. Sirakoulis, S. A. Chatzichristofis, and K. Christodoulou, "Randomblocks: A transparent, verifiable blockchain-based system for random numbers," 2019.

[33] J. Kidambi, D. Ghosal, and B. Mukherjee, "Dynamic token bucket (dtb): A fair bandwidth allocation algorithm for high-speed networks," *Journal of High Speed Networks*, vol. 9, no. 2, pp. 67–87, 2000.

Appendix A: Travel Distance

Here we calculate the number of hops that the 1366-byte and 32834-byte messages can travel through the network. The HTLC messages can only use the 1366-byte size. The first 66 bytes are allocated to the packet's headers, leaving 1300 bytes for the payload. The TLV (Type-Length-Value) payloads require a minimum of 47 bytes for intermediate hops (including 1 byte for `tlv_length`, 2 for `amt_to_forward`, 2 for `outgoing_cltv_value`, 10 for `short_channel_id`, and 32 for `hmac`) and 37 bytes for the exit hop (comprising 1 byte for `tlv_length`, 2 for `amt_to_forward`, 2 for `outgoing_cltv_value`, and 32 for `hmac`). Consequently, the maximum number of intermediate hops is $\lfloor (1300 - 37)/47 \rfloor = 26$. Including the exit hop, the longest possible path length is 27.

The calculation above is for HTLC messages that forward payments between network nodes. However, the payload needed for each hop is different for onion messages. The calculation for 1366-byte onion messages is as follows.

Similar to HTLC messages, OM headers consist of 66 bytes: 1 byte for the `version`, 33 bytes for the `public_key`, and 32 bytes for the `hmac` [12]. Therefore, the payload capacity is $1366 - 66 = 1300$ bytes. Additionally, payloads at each OM intermediary hop require a minimum of 70 bytes, comprising 1 byte for the `length`, 32 bytes for the `hmac`, 1 byte each for the `type` and `subtype`, 33 bytes for the `blinded_node_id`, and 2 bytes for the `enclen` [12]. Consequently, the maximum number of hops a message can traverse is $\lfloor 1300/70 \rfloor = 18$ hops. Including the last hop, the longest possible path length is 19.

For the larger 32834-byte OMs similar to above, we calculate the maximum travel

distance to be $\lfloor (32834 - 66)/70 \rfloor = 468$ hops. Including the last hop, the longest possible path length is 469.