Short Paper: Onion Messages on Leash

Amin Bashiri and Majid Khabbazian

University of Alberta
{abashiri,mkhabbaz}@ualberta.ca

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 static invoices, refunds, and asynchronous payments, onion messages may also be used for unintended applications such as streaming or spam. To mitigate this, LN nodes can impose a rate limit on forwarding onion messages. 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 through hundreds of nodes, affecting all the nodes on its way. In addition, the adversary can hide their true identity thanks to the privacy-preserving feature of onion routing. To address this threat, we propose a simple solution with two main components. The first component limits the distance over which OMs can travel. For this purpose, we propose two methods: a hard leash and a soft leash. The hard leash imposes a strict limit on 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 degrade the onion messaging service, assuming that the adversary does not control a significant fraction of funds in the network.

Keywords: Lightning Network · Onion Messages · Denial of Service

1 Introduction

The Lightning Network (LN) [19,12] was conceived as a second-layer solution to address Bitcoin's scalability challenges. In this network, nodes establish channels by jointly committing their funds in a multi-signature transaction on the Bitcoin blockchain. For each transaction, the participants within the channel determine the allocation of the locked funds by mutually signing new output transactions that reference the output of the initial funding transaction. This approach enables off-chain, rapid, and cost-effective transactions.

For LN to evolve into a comprehensive financial platform, it must expand its support to cover a broader spectrum of use cases. Many of these use cases require bidirectional communications between the payer and the payee before invoice generation and payment execution. Several methods have been proposed to incorporate this communication functionality into LN. One such method is LNURL (Lightning Network Uniform Resource Locator) [5]. LNURL leverages HTTP servers alongside LN nodes to establish bidirectional communication between the payer and the payee. A notable drawback of LNURL is that it necessitates users to operate an HTTP server beside their LN node. This external interaction elevates the node's resource requirements, introduces complexity, expands the attack surface, and can potentially compromise user privacy and anonymity.

An alternative solution to enable communication between nodes within LN is using Onion Messages (OMs). OMs [2] are transmitted within LN directly between peers, eliminating the necessity of establishing channels between them. OMs also take advantage of onion routing [14], which offers robust anonymity and privacy for users. On the negative side, OMs introduce a potential attack surface. With the capability to send anonymous and privacy-preserving messages that can traverse many hops, concerns arise regarding the possibility of exploiting this feature for a Denial of Service (DoS) [18,15] attack. More specifically, an adversary potentially controlling many LN nodes can flood the network with OMs.

In this work, we explore two distinct types of DoS attacks that can potentially be executed within the network. In the first attack, the adversary attempts to flood the network with unnecessary OMs to degrade the OM service in the whole network. As for the second attack, the adversary floods a victim node with OMs to prevent other nodes from reaching the victim node. To mitigate these attacks, we propose a simple solution composed of two components. The first component puts a leash on OMs to limit the distance they can travel in the network. The second component puts a leash on the forwarding rate. We show that these two components together can effectively mitigate the aforementioned attacks, provided that the adversary does not control a significant fraction of the total funds in the network.

2 System Model

We model LN using a graph $G_{LN} = (V, E_{LN})$, where V represents the set of nodes in the network, and E_{LN} 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.

We assume that each node in LN has a list of all other nodes in the network, together with their funds, defined as the sum of the capacities of channels they

own. A 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 in the specifications outlined in BOLT[1] (Basis of Lightning Technology) [1] for OMs, a specific rate-limiting algorithm is not prescribed. However, nodes are strongly encouraged to implement their own rate-limiting algorithms. 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.

The adversary controls one or more nodes in LN and can send OMs using these nodes at any rate it wishes without being detected. However, we assume that the total funds locked in the channels owned by the adversary are a small fraction of the total funds in the whole network. We assume that joining the Lightning Network does not present a major barrier to the adversary, but controlling a large portion of the total funds in the network does.

Considering the above model, we explore two DoS attacks. In the first attack, the adversary attempts to flood the network with unnecessary OMs to degrade the OM service. In the second attack, the adversary floods a victim node with OMs to prevent other nodes in the network from reaching the victim node.

3 Proposed Solution

In this section, we present our solution to mitigate the DoS attacks mentioned earlier. We will also provide an analysis of the effectiveness of the 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 [13], which achieves anonymity with just three hops [10], 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 needed is evidently much lower than the existing 504-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 contribute significantly to mitigating the considered DoS attack scenarios.

3.1 Limiting the Travel Distance

While HTLC onion packets are limited to 1366 bytes, OMs can have a larger size of 32834 bytes [3]. The expanded size of OMs introduces a potential vulnerability

[1] BOLTs are a set of specifications that detail the inner workings of the Lightning Network. It provides comprehensive guidelines regarding the expected behavior and interactions of nodes within the network.

a malicious actor can exploit to send messages that maximize the available space to traverse numerous hops. A 1366-byte message has a maximum travel distance of 20 hops, but the 32834-byte message has a much larger maximum distance of 504 hops. To mitigate this, we introduce two methods to impose a cap on the maximum number of times a message can travel in the network.

Hard leash. The first method we propose enforces a strict upper limit on the number of hops an OM can traverse. As illustrated in Fig. 1, 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 divided into two segments: a first portion of 326 bytes for routing information and a second segment of 32,508 bytes allocated for the payload. In this configuration, the 326 bytes dedicated to routing data restrict the message 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.

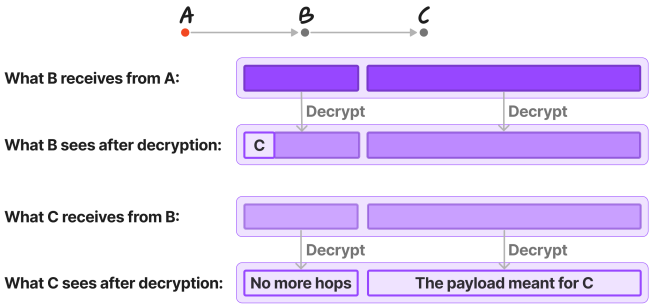

Fig. 1. An illustration of how an OM is structured into two parts while still fitting within a single packet. One part contains the information on hops of the path, while the other contains the payload, which is exclusively intended for the final hop. The message length remains constant regardless of the number of hops in the path.

Soft leash. Unlike the hard leash, this method does not force a strict hard limit on the maximum number of hops for an OM. Rather, it makes it exponentially difficult for an OM to travel far. This effectively serves as a practical deterrent against excessively long paths while allowing for flexibility within reasonable limits. In addition, it gives honest users the flexibility to opt for a lengthier path to enhance their privacy when necessary.

In this method, similar to the solution recently implemented in the Tor network [6], 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 en-

forces a maximum hops limit of 3, 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 2, 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.

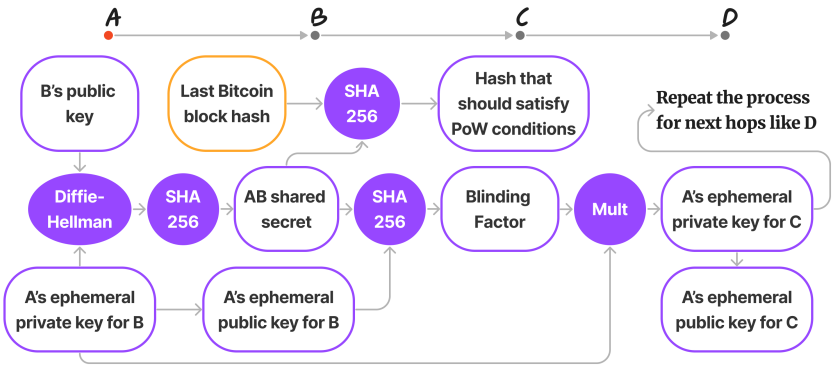

Fig. 2. Illustration of the process undertaken by the source node A to establish shared secrets with nodes along the path (B, C, D) for creating an onion-encrypted message. For each node along the path, the hash resulting from the combination of the shared secret and the latest Bitcoin block hash must adhere to PoW conditions, such as being below a certain value.

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 [8], we link our PoW to the latest Bitcoin block's hash as illustrated in Figure 2. Although Bitcoin block hashes are not a true source of randomness [9,11], they suffice for our use case. 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.

3.2 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. Additionally, they 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 $\alpha_A \cdot C_B$ for peer B by node A, where α_A is an adjustable parameter, and C_B is the sum of the capacities of channels owned by B. Setting the rate limit proportional to C_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_A 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 every other node in the network. 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 $\frac{1}{\alpha_A \cdot C_B}$), A will forward B's message and update its `last_forward_time`; otherwise, A discards B's message. The node can implement slightly more sophisticated techniques, such as sliding windows or token bucketing [17], to impose the rate limit more effectively.

Routing: In this study, we simplify the analysis by assuming that each node has an equal total channel capacity (i.e., $\forall A, B \in V$, $C_A = C_B$). In this setting, we select paths uniformly at random. In general settings, however, the routing algorithm should consider the sum of the channel capacities of the nodes when choosing a random path. Since our forwarding rules prioritize nodes with greater investment by allocating a higher rate limit, the source should choose random paths weighted based on the sum of capacities of each node's channels. This also decreases the probability of the message being routed through an adversary, assuming the adversary does not possess a significant portion of the network's funds.

3.3 Resistance of the Proposed Solution

We mentioned two attack scenarios in Section 2. In this section, we present a high-level analysis of the effectiveness of the proposed solution in these scenarios.

Degradation attack. In this attack, the adversary floods the network with OMs in an attempt to degrade the availability of the OM service. To analyze this attack, suppose the network consists of n honest nodes and f dishonest nodes controlled by the adversary (hence $|V| = n + f$). We simplify the analysis by assuming that each node (honest or dishonest) has an equal sum of channel capacities. We note that this simple assumption can be relaxed. Under this assumption, and assuming that $\alpha_A = \alpha_B$ for every two nodes A and B, the forwarding rate limit is the same for every node.

Let k be the maximum number of links a single OM can visit (i.e., how far an OM can travel). Thus, any dishonest node can saturate up to $(k-1) \cdot n$

honest links, where an honest link is defined as a link between two honest nodes. This is because a dishonest node can send a forwarding request to all n honest nodes, where each forwarding request can visit up to $k-1$ honest links on its way. Therefore, all dishonest nodes can saturate up to $f \cdot (k-1) \cdot n$ honest links. Also, notice that there are $f \cdot n + \binom{f}{2}$ dishonest links, where a dishonest link is defined as a link incident to a dishonest node. Therefore, there are up to

$$m = \overbrace{f \cdot (k-1) \cdot n}^{\text{saturated honest links}} + \overbrace{f \cdot n + \binom{f}{2}}^{\text{dishonest links}}$$

links that are either dishonest or saturated.

Now consider two honest nodes A and B, and suppose that A sends an OM to B through a random path of length $l \leq k$. The probability that a given link of this path is dishonest or saturated is $\frac{m}{\binom{n+f}{2}}$. Therefore, by the union bound, the probability that no link on the selected path is dishonest or saturated is at least

$$1 - l \cdot \frac{m}{\binom{n+f}{2}} \geq 1 - 2 \cdot \frac{f \cdot k \cdot l}{n}, \tag{1}$$

which is close to one if $f \ll n$, and k is small.

To have an idea of the effectiveness of the proposed solution, let us consider an example. Let us set $n = 16,000$ (at the time of writing, LN has about 16,000 nodes). Suppose we limit the maximum travel distance to 4 hops ($k = 4$) and assume that the adversary controls ten dishonest nodes ($f = 10$). Additionally, assume that the path between nodes A and B has a length of three hops ($l = 3$). Then, by (1), we get that A's message is successfully received by B with a probability of at least 98.5%. Note that A can significantly improve this success probability by sending its message through multiple disjoint paths.

Targeting a victim node. In this attack scenario, the adversary attempts to saturate all the honest links to a victim node, say B, so that another honest node, say A, cannot send a message to it. Although A can always send an OM directly to B, we will imagine that it would prefer not to do so to preserve its privacy. The adversary can execute the attack using even a single dishonest node: the dishonest node sends OMs to all the honest nodes at the maximum allowable rate, asking each node to forward the message to B. This alone, however, cannot prevent an honest node, say A, from sending an OM to B because even though all the links to B are saturated, B will not drop A's message since this message is destined to B itself. Therefore, as long as A's message reaches the node before B in the path, it will successfully reach B.

To prevent A's message from reaching B, the adversary needs to saturate all (or a large fraction of) the links at distance two of B. The number of such links is $\theta(n^2)$, i.e., the same order as the total number of honest links in the network. Using a similar argument as in the previous attack scenario, we can show that A's message has a high chance of reaching B, particularly if A sends its message to B using multiple disjoint paths.

4 Related Work

Tor [13], a close counterpart to OM service, recently experienced a DoS attack [7,16]. Tor's response involved integrating PoW as a defense mechanism [6,8]. 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 [10], whereas OMs are allowed to travel up to 504 hops; **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 more straightforward in LN than in the Tor network.

A probabilistic algorithm proposed in LN's mailing list [4] applies per-peer rate limits on incoming OMs to be relayed. The key idea is to track the source of each OM per outgoing connection. When a message hits the rate limit, an `onion_message_drop` is sent to the sender, indicating the rate limit breach. The sender then relays this drop message to the last sender, halving their rate limits with that peer. While occasional misattribution may exist, the scheme aims to statistically penalize the right incoming peer.

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 504; **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.

5 Conclusion

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.

6 Acknowledgment

We are grateful to the reviewers for their valuable feedback. Majid Khabbazian acknowledges the support of Alberta Innovates.

References

1. Basis of lightning technology (bolt). https://github.com/lightning/bolts. Accessed: Jan 10th, 2024.
2. Bolt 4: Onion messages. https://github.com/lightning/bolts/blob/master/04-onion-routing.md#onion-messages. Accessed: Jan 10th, 2024.
3. Bolt 4: Onion routing protocol. https://github.com/lightning/bolts/blob/master/04-onion-routing.md. Accessed: Jan 10th, 2024.
4. Lightning-dev mailing list: Onion messages rate-limiting. https://lists.linuxfoundation.org/pipermail/lightning-dev/2022-June/003623.html. Accessed: Jan 10th, 2024.
5. Lnurl documents. https://github.com/lnurl/luds. Accessed: Jan 10th, 2024.
6. Tor blog: Introducing proof-of-work defense for onion services. https://blog.torproject.org/introducing-proof-of-work-defense-for-onion-services/. Accessed: Jan 10th, 2024.
7. Tor blog: Tor is slow right now. here is what is happening. https://blog.torproject.org/tor-network-ddos-attack/. Accessed: Jan 10th, 2024.
8. Tor-dev mailing list: A first take at pow over introduction circuits. https://lists.torproject.org/pipermail/tor-dev/2020-June/014381.html. Accessed: Jan 10th, 2024.
9. Krishnendu Chatterjee, Amir Kafshdar Goharshady, and Arash Pourdamghani. Probabilistic smart contracts: Secure randomness on the blockchain. In *2019 IEEE international conference on blockchain and cryptocurrency (ICBC)*, pages 403–412. IEEE, 2019.
10. Fallon Chen and Joseph Pasquale. Toward improving path selection in tor. In *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pages 1–6. IEEE, 2010.
11. Panayiotis Christodoulou, Georgios Ch Sirakoulis, Savvas A Chatzichristofis, and Klitos Christodoulou. Randomblocks: A transparent, verifiable blockchain-based system for random numbers. 2019.
12. Christian Decker and Roger 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*, pages 3–18. Springer, 2015.
13. Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-Generation onion router. In *13th USENIX Security Symposium (USENIX Security 04)*, San Diego, CA, August 2004. USENIX Association. URL: https://www.usenix.org/conference/13th-usenix-security-symposium/tor-second-generation-onion-router.
14. David Goldschlag, Michael Reed, and Paul Syverson. Onion routing. *Communications of the ACM*, 42(2):39–41, 1999.
15. Brij B Gupta, Ramesh Chandra Joshi, and Manoj Misra. Distributed denial of service prevention techniques. *arXiv preprint arXiv:1208.3557*, 2012.
16. Rob Jansen, Tavish Vaidya, and Micah Sherr. Point break: A study of bandwidth Denial-of-Service attacks against tor. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1823–1840, Santa Clara, CA, August 2019. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity19/presentation/jansen.
17. Jayakrishna Kidambi, Dipak Ghosal, and Biswanath Mukherjee. Dynamic token bucket (dtb): a fair bandwidth allocation algorithm for high-speed networks. *Journal of High Speed Networks*, 9(2):67–87, 2000.

18. Neil Long and Rob Thomas. Trends in denial of service attack technology. *CERT Coordination Center*, 648(651):569, 2001.
19. Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments, 2016.