Payment Censorship in the Lightning Network Despite Encrypted Communication

Charmaine Ndolo
Dresden University of Technology, Germany

Florian Tschorsch
Dresden University of Technology, Germany

Abstract

The Lightning network (LN) offers a solution to Bitcoin's scalability limitations by providing fast and private off-chain payments. In addition to the LN's long known application-level centralisation, recent work has highlighted its centralisation at the network-level which makes it vulnerable to attacks on privacy by malicious actors. In this work, we explore the LN's susceptibility to censorship by a network-level actor such as a malicious autonomous system. We show that a network-level actor can identify and censor all payments routed via their network by just examining the packet headers. Our results indicate that it is viable to accurately identify LN messages despite the fact that all inter-peer communication is end-to-end encrypted. Additionally, we describe how a network-level observer can determine a node's role in a payment path based on timing, direction of flow and message type, and demonstrate the approach's feasibility using experiments in a live instance of the network. Simulations of the attack on a snapshot of the Lightning mainnet suggest that the impact of the attack varies from mild to potentially dramatic depending on the adversary and type of payments that are censored. We analyse countermeasures the network can implement and come to the conclusion that an adequate solution comprises constant message sizes as well as dummy traffic.

2012 ACM Subject Classification Networks → Network privacy and anonymity; Security and privacy → Software and application security; Security and privacy → Distributed systems security

Keywords and phrases Lightning network, payment channel networks, censorship resistance

Digital Object Identifier 10.4230/LIPIcs.AFT.2024.12

Supplementary Material
Software: `https://github.com/tud-dud/lightning-censorship-simulator`

1 Introduction

Bitcoin [25] and similar blockchain-based payment systems continue to enjoy significant popularity. While Bitcoin is to date the most popular based on its market capitalisation, it suffers from grave constraints with respect to scalability, which limit its ability to compete with traditional (centralised) payment systems. Layer 2 solutions are gaining traction as a feasible solution to the scalability challenges by enabling off-chain transactions. One such solution is the Lightning network (LN) [30] – a peer-to-peer (P2P) payment channel network (PCN) enabling fast, low-cost and private Bitcoin payments. It is a network of off-chain bilateral channels in which funds can move in either direction between the two channel partners. LN also implements multi-hop payments such that payments can be routed over multiple intermediate channels in cases where the sender and beneficiary of a payment do not have a direct channel. In order to offer a degree of payment privacy, all P2P communication subsequent to connection establishment is encrypted using the Noise [29] protocol framework. Furthermore, LN uses the Sphinx mix format [10] to implement onion routing of payments across the network. This means that, among others, routing nodes only know their predecessor and successor when forwarding a payment, but do not know if either is the source or destination of the payment.

A compelling selling point of decentralised P2P systems is their censorship-resistant nature due to their fundamental design, i.e., there is no single point of failure or owner. Nonetheless, blockchain-based cryptocurrencies such as Bitcoin have been subject to state-wide bans enforced via legal frameworks and/or technical means such as aggressive protocol blocking.

While the broader topic of Internet censorship is by no means a new one, it remains highly relevant due to some of the censorship currently imposed across the globe, e.g., in China [15], India [46], Iran [4] or Turkmenistan [28]. Blockchain networks such as Bitcoin and Ethereum have also been shown to be vulnerable to censorship despite their design [22,44]. State censors employ different techniques such as high prices, notoriously low broadband speeds [17,35] or various network-level techniques [15,46] to restrict access to services. Powerful tools such as Geneva [8] and OONI [13] are able to detect or even evade censorship based on a censor's identified strategies. If a censor using network-level tactics intends on bypassing common censorship-detection tools, the need for subtle, difficult to reproduce yet effective measures arises. Additionally – and aside from ethical concerns – such restrictions are detrimental to their national and international image and have the potential to spark unrest. Thus, a censor may instead look to discretely implement such a ban such that it either goes unnoticed or initial blame is put on other actors, e.g., application issues. Assuming that a certain level of operation (within the censor's area of jurisdiction) can be maintained, the censor may even be able to plausibly deny the fact that they are indeed tampering with network traffic.

In this work, we explore the Lightning network's susceptibility to censorship by a network-level actor such as a malicious autonomous system (AS). For the previously detailed reasons, we assume that the censor's goal is not to disrupt the entire network but to control participation in the LN within their domain. In doing so, the censor seeks to limit their impact on the day-to-day operations of the greater LN and avoid collateral damage. The censor strives to remain undetected as much as possible such that from an observer's perspective, e.g., a user issuing payments or network explorer, it is difficult to recognise that a given node is under attack but aims to maximise their impact on the censored nodes.

Our work expands on previous work by von Arx et al. [43] in which they showed that application messages can be identified based on traffic analysis. We first confirm that a network-level adversary is able to accurately classify LN traffic using the header data and flow direction of transmitted packets by implementing a rule-based classification program for live LN traffic. Our results indicate that it is possible to accurately identify LN messages in real time despite the fact that all P2P communication in the network is end-to-end encrypted. Based on this, we show how a network-level actor can censor all payments routed via their network using a simple state machine to determine if a packet should be dropped. All other LN operations, e.g., channel management, remain unaffected. Due to the atomic nature of the payment process in the LN, dropping select messages eventually results in payment failure without attempting alternative routing options. This result may not be adequate for a censor who does not want to tamper with third-party activity that just so happens to be traversing their network. Thus, we then show that it is possible for a network-level observer to determine a node's role in a payment path based on the timing and direction of flow as well as the message type. We use the information to enhance the attack such that a censor can selectively block payments, e.g., block intra-AS payments but permit inter-AS payments.

We implemented the attack as an efficient `netfilter` program and validated the attack's feasibility and performance in a private network as well as in the public testnet. Our experiments show that for rates of up to 1 payment per second, we are able to correctly determine a node's role in a payment path. While this rate may sound underwhelming, it exceeds the currently estimated payment rates in Lightning by five orders of magnitude.

Furthermore, simulations of the attack on the mainnet's channel graph show that the impact on the broader network is almost non-existent in the case of selective censorship. Based on reviewing current state-of-the-art traffic fingerprinting protection measures, we discuss and verify possible mitigation strategies for the LN. We come to the conclusion that an adequate solution entails implementing some form of cover traffic and constant-size messages in the network similar to what is implemented in the Tor network.

To summarise, the following are our main contributions:

1. We show that network packets can accurately be mapped to the corresponding LN message types using the payload length and sequence of messages in real time;
2. We exploit timing information and type of message to identify an on-path node's role in a payment path;
3. Based on the preceding contributions, we present a censorship attack on the LN that is founded on selectively dropping network packets identified to be related to payments;
4. We implement the attack, deploy it in a private Lightning network and report on our findings. We evaluate the attack using our implementation and simulations; and
5. We analytically discuss possible countermeasures the LN can implement and derive recommendations for the network.

The remainder of this paper is structured as follows. We provide a pertinent introduction to the LN and present our system model in Section 2. The core of this work is Section 3 in which we describe a censorship attack on the LN and evaluate it comprehensively in Section 4. We discuss countermeasures for this attack vector in Section 5 and provide an overview of related work in Section 6. We conclude this work and discuss avenues for future work in Section 7.

2 Background and system model

We provide the reader with a pertinent introduction to the Lightning network in Section 2.1; we refer the reader to [1, 30, 31] for a comprehensive introduction beyond the scope of this work. We briefly analyse the network topology in Section 2.2 and describe our threat model in Section 2.3.

2.1 The Lightning network

The Lightning network (LN) is a peer-to-peer (P2P) network of bilateral off-chain payment channels, i.e., a payment channel network (PCN). A payment channel signifies a financial relationship between a pair of nodes in which a set number of funds (the channel's *capacity*) is committed via a transaction on the Bitcoin blockchain. Lightning payments alter the distribution of the channel's capacity (*balances*) between the two endpoints. Payments in the LN can be routed via multiple hops for a fee that is independently set by each node. In order to route payments securely over multiple hops, payments are secured by Hashed Timelock Contracts (HTLCs), which guarantee that payments are made atomically, i.e., a payment either succeeds at all hops or fails at all hops. An HTLC is basically a conditional payment that can be claimed by producing a preimage that is revealed by the payment's beneficiary. During channel establishment, each node defines and announces how long they are willing to wait for an HTLC to be resolved – the *time lock*. If the *time lock* expires before the HTLC is resolved, the HTLC expires and is settled on-chain which requires a forceful closure of the affected channel.

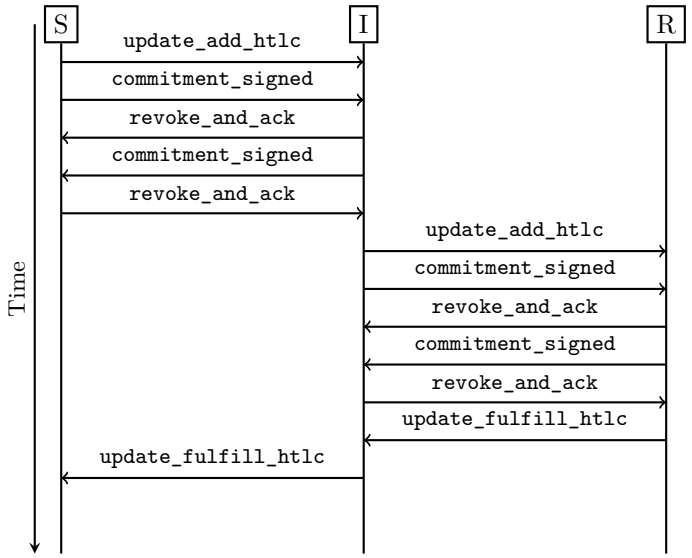

Figure 1 The sequence of messages exchanged for a payment between a sender (S) and recipient (R) routed via an intermediate hop (I). The `update_fulfill_htlc` message is only sent in the case of a successful payment and is replaced by an `update_fail_htlc` message otherwise.

Once a payment channel has been established, an arbitrary number of payments can be made over the channel. Finding a suitable path for a payment is an essential part of LN and is delegated to the sender of a payment. Based on the public channel graph, the sender tries to find a path connecting them to the recipient of the payment. For the sake of illustration, the following assumes a payment from node S to node R made via an intermediate node I. S begins by encoding the calculated path in an onion packet using the Sphinx message format [10], i.e., a packet with multiple layers of encryption that each identify the next hop on the path. Forwarding nodes along the path hence only know their predecessor and successor on the path, but do not know if either is the payment's source or destination. S initiates the payment by constructing an HTLC and sending it in an `update_add_htlc` message with the onion packet to I. Upon receipt, I decrypts the topmost layer to receive its payload and prepares to forward the `update_add_htlc` message to the next hop. However, I will only forward it to the next hop after the new conditional payment is reflected in the $S - I$ channel's state. The state update must also be irrevocably committed by both nodes using a handshake of `commitment_signed` and `revoke_and_ack` messages as shown in Figure 1.

Once the state updates have been successfully completed, I forwards the remaining onion packet to R in a new `update_add_htlc` message. I and R then negotiate the new state in the same way S and I did (cf. Figure 1). As soon as I and R conclude the handshake necessary for the channel update, R sends an `update_fulfill_htlc` message to I. The message contains the preimage needed by each hop to settle the HTLC with its channel partner. The `update_fulfill_htlc` is propagated to all hops along the path in reverse order such that each hop can redeem the conditional payment. In the event of an error along the way, e.g., due to insufficient balances or time lock, a node will immediately send an `update_fail_htlc` to its predecessor instead, which will be propagated to all preceding nodes as well.

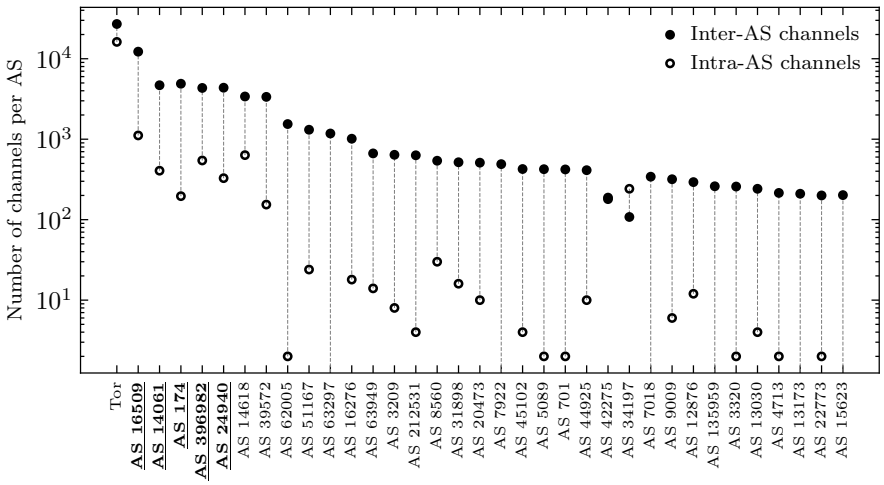

Figure 2 The sum of channels where both endpoints belong to different ASs and the sum of channels where both belong to the same AS for the 35 ASs with the highest number of channels. The top five ASs in the network w.r.t. to the number of channels are underlined and printed in bold.

2.2 Topology

It is well-known that the application level of the LN is highly centralised [21, 34, 41]. That is, while the network is considerably large in terms of the number of nodes and channels, most payments are routed via a small subset of available channels. This has been shown to be detrimental to the network's privacy goals and overall resilience [19, 26]. Recent work [43] revealed that the network layer is similarly centralised, with just a handful of ASs technically being able to compromise payment privacy. For instance, 80% of all Lightning channels are hosted at just five ASs.

As gaining a deeper understanding of the topological structure may prove to be useful to discover potential censorship strategies, we examined the distribution of channels to ASs based on a snapshot of the mainnet's channel graph on 12 January 2024. The network comprised 12,781 nodes and 112,958 channels after reducing it to its largest strongly connected component. We pruned all nodes (and their channels) that had not announced at least one public network address from the obtained channel graph, which leaves approximately 22% of the nodes in the channel graph. We then mapped each node's announced address to the corresponding AS using the GeoLite2 database.[1] We examined the distribution of node degrees across AS and find that all high-degree (> 500 channels) nodes belong to different ASs. Further, we analysed the share of channels in which both endpoints belong to the same AS and depict the results in Figure 2. The figure shows the total number of channels that are shared by two different ASs and the total number of channels that belong to the AS alone. Except for nodes connecting to the network over Tor and a handful of ASs, e.g., AS 34197 or AS 42275, most channels in Lightning are between a pair of ASs and not within the same AS.

[1] Available at `https://www.maxmind.com` (accessed on 12 January 2024).

2.3 Adversary Model

We explore the feasibility of imposing censorship of the LN in this work. As previously mentioned, we presume that the censor aims to impose (from their perspective) effective censorship within their area of jurisdiction. In other words, the censor is not interested in disrupting the greater LN, but only controlling Lightning's operation in their sphere of influence. Additionally, the censor wants to maintain plausible deniability and hence looks to implement the ban such that a certain level of operation is upheld within their domain despite the ongoing censorship. This is why applying less sophisticated methods such as port and IP blocking are out of the question for the censor.

Similar to multiple related works [27, 43], we assume a powerful yet malicious network-level adversary such as an AS or a party cooperating with an AS. While the attack can be executed by any adversary with access to network-level traffic, e.g., an operator of a Lightning node, the impact and significance of the attack is directly related to the adversary's scope of influence. The adversary's foremost goal is to control activity in and access to the LN within their area of influence. For instance, this may be to enforce a controversial ban on cryptocurrencies.

The adversary expects that all inter-peer communication is end-to-end encrypted as per the Lightning specifications [1]. The adversary is only interested in LN nodes using a clearnet address because of the fixed-size cells transmitted by the Tor network. Furthermore, we assume that all nodes are operating on the default port: Transmission Control Protocol (TCP) port 9735 [1]. In case a node is using a non-default port, the adversary may use publicly-available data to trivially identify the port in use. Similarly, the adversary can refer to such data to learn which client implementation a node is running or infer the client [24]. Knowledge of the client implementation in use is, however, strictly not necessary.

We focus on an adversary that fully controls at least one AS network. The network-level adversary can observe and inspect all communication sent over their network; it is however encrypted by the application layer. As the adversary wants to minimise the risk of detection, blocking all traffic on port 9735 would be self-defeating. Instead, and in order to maintain a level of operation and plausible deniability, the adversary is capable of executing refined filtering techniques such as selective packet dropping.

2.4 Ethical Considerations

We would like to emphasise that the primary goal of this work is to contribute to further developing and improving the network for all Lightning users. Uncovering, presenting, and fixing potential issues in the network is a core part of that process. We do not see this work as an instruction manual for adversaries and strongly disapprove of any misappropriation of our work. It is for this reason that we have decided to not make our proof-of-concept implementation of the attack available to the public. We believe that this paper contains enough information and details for the reader to reproduce with their own implementation. We made the code available during the peer review process and will consider doing the same to researchers upon request.

As far as the practical evaluation of the presented attack is concerned, we followed the guidelines of the Menlo report [5] and general security research best practices. In particular, with the exception of obtaining a network snapshot from our own node, we did not interact with the public mainnet in any way. We deployed a modified version of our proof-of-concept implementation to the testnet in order to validate the feasibility of the attack's preliminary phase. However, at no point did we actually mount the attack in the testnet. All adverse

Table 1 Comparison of expected message sizes (in bytes) as specified in [1] and actual captured message sizes. The sizes refer to messages containing exactly one HTLC. The node running LND sent all messages in two packets – an $18B$ payload followed by the remainder.

	Message size (bytes)		
Message	Specifications	LND `v0.17.2-beta`	CLN `v23.11.2`
`update_add_htlc`	1450	18 + 1468	1486
`commitment_signed`	162	18 + 116	134
`revoke_and_ack`	97	18 + 115	133
`update_fulfill_htlc`	72	18 + 90	108

experiments were conducted in our private network comprising only nodes we set up for the precise purpose. In order to evaluate the potential impact of our work on the main network, we followed a simulation-based approach using the obtained snapshot. The simulation mocks payment routing in the network by reconstructing the topology locally.

3 Censorship Attack

In the following, we present a novel censorship attack on a set of nodes in the LN. The attack leverages the fixed message sizes defined in the Lightning specifications [1] as well as its overall protocol design. This allows an adversary to accurately classify encrypted application traffic based on network-layer data without much effort in real time. Subsequently, we show how a network-level attacker can censor payments and enhance the attack with knowledge on a payment source and destination.

3.1 Message Classification

A recent work [43] presented an attack on privacy in the LN based on monitoring network-layer traffic. The first step of the attack is to map TCP packets to application messages based on the payload lengths in combination with the sequence of observed packets. The censorship attack we demonstrate in Section 3.2 makes use of the same shortcoming. To that end, we take a closer look at identifying LN messages based on network-level observations. In what follows, we use HTLCs to exemplify the procedure. It, however, applies to other message types analogously.

Figure 1 illustrates the type and sequence of messages exchanged between two channel partners during the payment process. By generating and capturing LN packets in a private network in order to validate the feasibility of matching network packets to application messages, we established that none of the captured TCP payload lengths corresponds to the sizes defined in the BOLTs. Table 1 shows the actual message sizes for the two most popular clients [24, 47] – Lightning Network Daemon (LND) and Core Lightning (CLN). We observed that nodes running on LND sent each application message in two TCP packets, the first of which was always $18B$. While the sizes of the messages sent by these clients differ from what is expected, they remains constant and hence allow us to identify the application messages based on the size, order of arrival and direction. The direction is not actually strictly necessary but it provides additional insights on the packet origin that we make use of to refine the adversary's strategy. Additionally, the adversary should know which client software is running due to the slight differences in payload size. Inferring the implementation, however, can be done with reasonable effort by, e.g., analysing the transmitted payloads

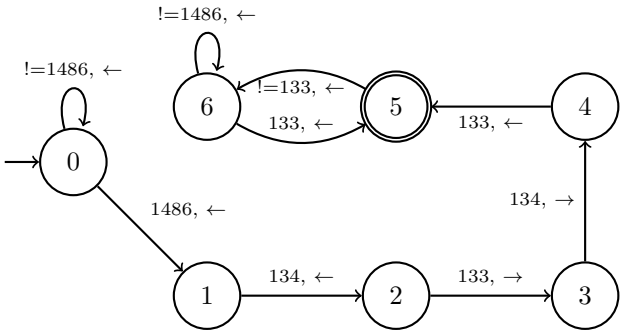

Figure 3 A state machine implementing the censorship rules for incoming payments. State transitions are defined by the (sum of the) TCP payload lengths and direction of flow (→ for egress traffic, ← for ingress traffic). All omitted transitions reset the state machine to its initial state.

(cf. Table 1) or using the public channel graph [24, 47]. For example, a packet with a $134B$ TCP payload from A to B that is preceded by a $1486B$ payload in the same direction and succeeded by $18B$ and $115B$ payloads in the reverse direction, is bound to have been a `commitment_signed` message. Furthermore, A is likely running CLN whereas B is almost certainly running LND.

3.2 Payment censorship in the LN

As per Section 2.3, the adversary wants the attack to go largely unnoticed and is indifferent towards third parties. This is why simply blocking or interfering with all LN traffic is not a viable strategy. However, given that an adversary is capable of identifying LN application messages by monitoring the network traffic, they can selectively interfere with the traffic passing their network.

In the following, we show how an adversary such as an AS can censor all payments involving nodes in their network while maintaining a degree of plausible deniability by preserving LN functionality in their network. Consequently, the adversary does not interfere with any messages pertaining to node management and channel management, e.g., open and close channel messages. By allowing nodes to operate Lightning channels, neither the affected nodes nor other observers have credible reason to put blame on the AS when issues with payments start to surface. For instance, a (suspicious) user inspecting the LN topology using a network explorer will not recognise that a malicious AS is suppressing its nodes' participation in the network.

However, the adversary pays close attention to all TCP traffic on port 9735 that is assumed to be payment-related using the method described in Section 3.1. The adversary must then interrupt the payment process in order to provoke application failures. The adversary prompts such failures by dropping select packets following the state machine in Figure 3 for each pair of source and destination. State 0 is the initial state in which the adversary waits for an `update_add_htlc` message which means that a payment is underway. The state machine's transitions are defined by the payload lengths of the series of messages exchanged between two nodes when a payment is being made. We choose to have the adversary drop the first `revoke_and_ack` message that is sent from the source to the recipient (cf. Figure 1). This is identified by arriving at the accepting state, state 5, after a series of messages. Although the adversary could drop the other payment messages, we opted for the `revoke_and_ack` message due its terminal position in the series of exchanged messages. We thus expect that

the adversary will make "correct" decisions. Note that Figure 3 depicts the rules applied by an adversary for incoming payments only. Outgoing payments can be censored analogously by reversing the direction of flow in the transition rules.

As discussed in Section 2.1, the `revoke_and_ack` message is sent in response to a state update; it revokes the previous state and acknowledges the new one. The payment process can therefore not proceed until either the recipient receives the `revoke_and_ack` message or the payment's timelock expires. At this point, a node will no longer attempt to route a payment via an alternative path until the payment conclusively fails. Lightning clients thus initiate retransmissions of the unacknowledged `revoke_and_ack` message for as long as the payment is valid. This is why the adversary needs state 6 in Figure 3, i.e., to block all subsequent `revoke_and_ack` messages from getting to the recipient. Note that the effect is similar when the first `commitment_signed` from the sender to recipient is not acknowledged.

3.3 Selective censorship

So far, the adversary is able to monitor network traffic and block all payments routed via their network by dropping all `revoke_and_ack` messages. This is not yet quite satisfactory because the adversary's goal is to remain largely unnoticed and minimise the collateral damage. The current strategy, however, defeats this objective. We thus refine the adversary's packet dropping criteria by showing how to determine a node's role in a payment based on network-level observations. The adversary can then selectively drop packets depending on the censored node's position in the path. Besides contributing to the adversary remaining undetected, the ability to selectively drop LN messages using knowledge of a node's position allows them to block payments based on origin and/or destination. For example, a malicious AS could let all payments pass that neither originate from nor are destined for their network, or allow all incoming payments but block outgoing payments.

An on-path node in the LN can occupy one of three roles for a given payment: *sender*, *intermediary* or *recipient*. When forwarding a payment in the network, intermediate nodes are not aware of other nodes' or even their own positions in the path. While determining a node's role has been subject of previous work [19], we are, to the best of our knowledge, the first to do so based on live network traffic. The adversary is hence able to use the node's role for their decision on whether or not to block a packet. Based on the combination of packet direction, message type and position in the sequence of transmitted messages, it is possible to determine a node's role as follows:

1. *sender*: a node is the initiator of a payment if it sends an outgoing `update_add_htlc` message "out of the blue". In other words, if a sufficient amount of time t has passed since the last incoming `revoke_and_ack` message, we conclude that the current `update_add_htlc` message belongs to a separate payment. Due to the symmetric exchange of messages during payment routing (cf. Figure 1), an intermediate hop will always receive a `revoke_and_ack` message before offering an HTLC to the next hop in the path. If there is no such `revoke_and_ack` message, the purpose of the `update_add_htlc` message must be to initiate a new payment.
2. *intermediary*: if that less than t time has passed since receipt of a `revoke_and_ack` message when an `update_add_htlc` message is sent, i.e., an incoming `revoke_and_ack` was observed within time t before the outgoing `update_add_htlc`, the node is an intermediary.
3. *recipient*: when a node sends an `update_fulfill_htlc` message, it is the final destination of the payment if the previous (incoming) message was a `revoke_and_ack` message. We can conclude this because an intermediate hop will always send a new `update_add_htlc` message after receiving a `revoke_and_ack` so as to offer an HTLC to the next hop (cf.

```
iptables -I INPUT -p tcp --dport 9735 -j NFQUEUE --queue-num 1
iptables -I OUTPUT -p tcp --dport 9735 -j NFQUEUE --queue-num 1
```

Figure 4 The iptables rule set to direct all ingress and egress TCP traffic destined for port 9735 to queue 1.

Figure 1). Note that we can only know with certainty that a node is the destination during the settlement of the HTLC. This means that all HTLC offers need to be delivered in order to determine if a node is the recipient.

The `update_fulfill_htlc` message only applies to successful payments but similar logic can be applied for failed payments when an `update_fail_htlc` message is sent. Using the aforementioned rules, the adversary can augment their attack and apply selective censorship.

3.4 Implementation

There are generally several feasible options to implement the attack. However, bearing the following properties in mind, we chose to implement the attack using the `netfilter`[2] framework.

1. *performance*: as the adversary only wants to interfere with relevant traffic, an efficient implementation is crucial. The filter thus needs to be capable of making in-flight decisions in a very efficient manner;
2. *scalability*: due to the LN's network-level centralisation, it is safe to assume that such a malicious AS observes up to thousands of channels concurrently. Furthermore, increasing the complexity of the state machine, e.g., to accommodate other message types, should not come at the cost of performance; and
3. *generalisability*: an implementation that does not rely on the specifics of an adversary's infrastructure.

While a hardware-level firewall, i.e., on the network interface card (NIC), may be attractive from a performance and scalability standpoint, the functionality generally depends on the specific NIC. In contrast, the `netfilter` project has been readily available in the Linux kernel since version 2.4 and provides, among others, the iptables module.

We implemented Figure 3 as a user space program using the `nfq-rs`[3] library in Rust. In order for the program to receive packets, we must first define an iptables rule set that is responsible for directing all TCP packets on port 9735 to a `netfilter` queue. The relevant rule set is shown in Figure 4. Note that we implemented the state machine in Figure 3 without regard for the source address as correlating independent TCP streams is out of the scope of this work. In other words, our program does not maintain state for different source addresses, and assumes that multiple unrelated payments are not received concurrently. If a packet is determined to be an incoming `revoke_and_ack` message, the program returns an `NF_DROP` verdict, i.e., the packet is discarded. All other packets are allowed to traverse the network stack by issuing the `NF_ACCEPT` verdict.

A notable alternative to the `netfilter` project is eXpress Data Path (XDP) [2, 16] – a framework that enables packet processing within extended Berkeley Packet Filter (eBPF) programs. XDP has been available in the Linux kernel as of Linux 4.8 and requires neither specialised hardware nor kernel bypass. It is an integrated fast path in the kernel stack

[2] `https://www.netfilter.org`
[3] `https://github.com/nbdd0121/nfq-rs`

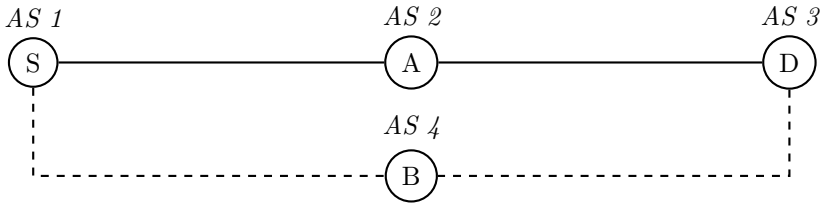

Figure 5 The private network environment used to validate the attack's practicability. We assume that each node belongs to a different AS and A's AS is malicious. The solid path represents the preferred path between S and D w.r.t. the path selection parameters whereas the dashed path represents an alternative path.

and works together with the TCP/IP stack. Packet processing happens before meta-data structures are allocated by the kernel leading to high processing speeds [33]. As the majority of LN traffic is not dropped by the adversary, we do not expect to gain a significant improvement in performance from XDP. Nonetheless, and for the sake of comparison, we also implemented the attack as an eBPF program that makes use of XDP to process the incoming packets following Figure 3. However, as XDP inspects just ingress traffic, this implementation only features the "base" attack described in Section 3.2.

4 Analysis

We evaluated the attack using the proof-of-concept implementation on deployed Lightning nodes. We first describe our evaluation setup then analyse the attack in various scenarios as well as its impact on the greater network based on conducted simulations.

4.1 Evaluation Setup

Due to the potentially destructive nature of the attack on the public network, we did not perform any measurements on the mainnet. All experiments were conducted either in the testnet or in a private network depending on the potential for harm and interference with other nodes. Both setups are described in the following.

4.1.1 Regtest

We set up a private Bitcoin network in regression test (regtest) mode which allowed us to deploy the complete attack code without interacting or interfering with other nodes in the public networks. Furthermore, regtest mode allows users to create a private blockchain and mine blocks instantaneously as the mining difficulty is set to zero. We configured four different Lightning nodes in the network as shown in Figure 5 running on different machines within the same network. S, A, B were all running LND `v0.17.2-beta` while D was running CLN `v23.11.2`. We assume that all four nodes are in different ASs and that AS 2 is adversarial. The attack code is therefore attached to A's network interface.

4.1.2 Testnet

In order to validate the attack in a realistic environment, we set up a node, A, running LND `v0.17.2-beta` in the public testnet. We strategically opened six balanced channels with moderate capacities between $300k$ sat and $500k$ sat to six nodes nodes, $I_1, I_2, I_3, I_4, I_5, I_6$,

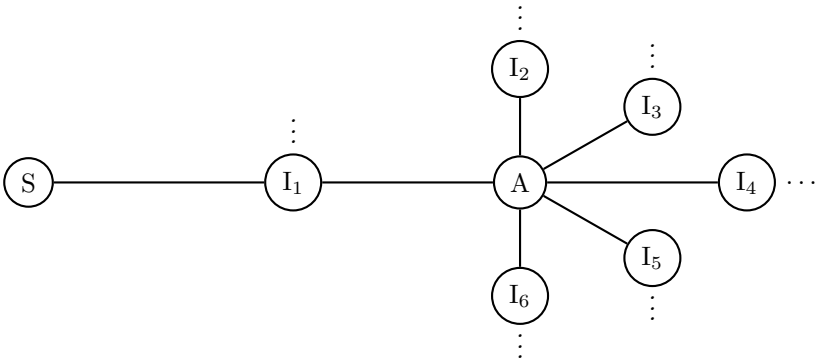

Figure 6 Our two nodes, the sender S and the adversary A, in the testnet with public channels to other nodes I_i in the network. Three dots symbolise the intermediate nodes' channels with other unrelated nodes. The attack is deployed on A's machine.

in the network and configured zero-fee routing policies. The channels were positioned to connect previously unconnected differently-sized hubs to each other in the hope of receiving routing requests.[4]

We deployed a modified version of the `netfilter` program on A's machine and attached it to A's only network interface. The program was modified such that instead of dropping a packet when the relevant state is reached, an entry is written to a log file notifying us that the packet would have been dropped. We also modified A's LND source code to log whenever a `revoke_and_ack` message is received – no other changes were made to the client software. We reiterate that no harm was caused to other nodes or the network in general.

We also set up a second Lightning node, S, with a channel to one of the nodes A was connected to as illustrated in Figure 6. We abstain from a direct channel between our two nodes in order to route payments over the Internet. We then generated random payments worth 1100 sat from S to the nodes $I_1, ..., I_6$ using the `sim-ln` tool.[5] We chose this amount as its the lowest amount satisfying the minimum payment amount all involved nodes were willing to forward. Due to the topology, all of the payments coming from S could only be routed via A. This resulted in a total of 71 payments in the span of 24 hours that were all delivered successfully.

4.2 Feasibility

In the following, we look at the message classification efficacy of the approach described in Section 3.1. Hereafter, we discuss the practicability of the attack described in Section 3.2.

4.2.1 Accuracy

In order to evaluate how well message identification works when packets are sent via the Internet, we used our testnet setup and deployed the code in the public Lightning testnet.

We compared the ground-truth LN message and our program's output using the generated logs, and calculated commonly used classification metrics for the `revoke_and_ack` message type: precision and recall. We recorded a precision of 1.0 and recall of 1.0. This means

[4] At the time of writing, approximately 2 months since joining the testnet, we are yet to receive any routing requests.

[5] `https://github.com/bitcoin-dev-project/sim-ln`

that the program returns neither false positives nor false negatives. These results are not unexpected and emphasise the exact problem brought by the highly deterministic nature of communication in the LN. We discuss the accuracy for higher payment rates in Section 4.3.

4.2.2 Practicability

Given the confidence that we can correctly identify encrypted LN messages, we sought to verify that the adversary can actually censor nodes in its network. We thus performed all of the following tests in our private regtest network.

In the first experiment, S tried to send $10k$ sat to D. As a result of the fees and timelock advertised by A and B, the most attractive path for payments from S to D was via A. After receiving the HTLC offer from S, A offered an HTLC to D by sending an `update_add_htlc` message immediately. The attack code thus correctly identified that the payment is not destined for A and does not drop any packets.

In the second experiment, S attempted to send $10k$ sat to A via their shared channel. We run the "base" attack on A's interface as we know it is the recipient and can only otherwise determine the recipient as the HTLC is being settled (cf. Section 3.3). We evaluate identifying a node's role in a payment path in an ensuing analysis. Once the program got to state 5 of Figure 3, the `revoke_and_ack` message was dropped. S retransmitted the packet as it is not acknowledged by A before closing the TCP connection. This left the channel in a temporarily inactive state and the payment in a pending state. After an exponential backoff period, S reestablished the connection and sent the `revoke_and_ack` message again. Note that S can still open a P2P connection as the code only drops `revoke_and_ack` messages. All subsequent `revoke_and_ack` received on A's interface were dropped which triggers the connection close, reestablishment and retransmission loop. We advanced the blockchain manually by mining blocks until the time lock elapsed. At that point, the payment attempt failed permanently, and S forcefully closed the channel as well as the TCP connection. We reversed the direction of payment and observed similar behaviour on the CLN node with a few minor differences mainly with respect to retransmissions.

In summary, we confirmed that it is possible to execute the attack and block payments based on network-level observations. Furthermore, we verified that the adversary is able to selectively censor payments and thus leave third-party payments intact.

4.3 Performance

Subsequent to the feasibility analysis, we studied the implementations' performance in regard to the induced delays, throughput and accuracy at different payment rates. The sole fact that the attack can be executed is not sufficient if such is not possible efficiently.

4.3.1 Latency

In the first of the three performance-related measurements, we examined the delay added to each TCP packet received on or directed to port 9735 by both the `netfilter` and XDP implementations. Figure 7 shows the time required to process TCP packets in microseconds by both implementations, i.e., the duration from the program first accessing a packet to a decision being made on the packet. The data was collected during the 24-hour time frame in which the measurements in Section 4.2.1 were performed and is depicted as a violin plot.

The results indicate that both implementations are quite efficient and issue a verdict on packets within the same median time of $\approx 11\mu s$ per packet. The mean processing time is 13µs per packet and 14µs per packet for the `netfilter` and XDP programs respectively. That

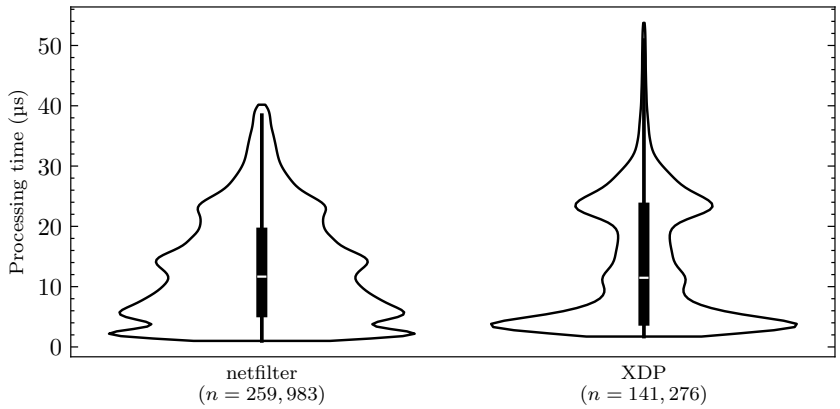

Figure 7 The packet processing times in microseconds for the `netfilter` and XDP programs based on packets received in the testnet over a period of 24h. The difference in the observed number of packets is due to the fact that XDP only receives ingress traffic.

equates to a mean throughput of $\approx 76,923 pps$ and $\approx 71,4428 pps$ respectively regardless of packet size. It may be surprising that the XDP program does not outperform the `netfilter` implementation despite XDP's superiority to iptables in respect to speed [7,8]. However, these measurements were performed on the user-space code attached to either of the subsystems and do not reflect the underlying technologies' throughput capabilities. In summary, we conclude that the delays induced by the additional filtering layer are negligible and do not hamper the feasibility of the attack. Such delays in the range of tens of microseconds are likely to go unnoticed by LN users or even routing nodes.

4.3.2 Throughput

We studied the maximum rate at which packets may be received by the `netfilter` program before they start being dropped due to congestion in the queue. The maximum queue length defaults to 1024 packets; all packets will be dropped as long as the target queue is full.

As per the previous measurements, the `netfilter` program achieves a mean throughput of $\approx 76,923 pps$. Hence, in order to have 1024 queued packets, the program must receive packets at a rate roughly 1000 times higher than $76,923$, i.e., $76,923 \cdot 10^3 pps$. Based on the traffic we observed in the testnet, we strongly believe that it is highly unlikely for a single Lightning node to generate and/or receive packets at speeds remotely close to that.

The largest AS (with respect to the number of nodes) in the mainnet is AS 14618 (Amazon.com) with 298 nodes as of 12 January 2024. Let us assume that, hypothetically, AS 14618 wants to execute the attack using a single instance of the program, i.e., all LN packets to/from the 298 nodes are processed sequentially by the same instance of the program. Further, we assume that the number of packets at each node is even.[6] This means that each node must pass approximately $3 \cdot 10^5 pps$ to the program in order to achieve a combined rate of $76,923 \cdot 10^3 pps$. Similarly, we do not consider such rates to be feasible in the LN. While we have made simplifying assumptions, these results indicate that the censorship attack can be executed in a large-scale manner using `netfilter`.

[6] This is a reasonable assumption to make as the amount of traffic at central nodes and less central nodes probably balance each other out.

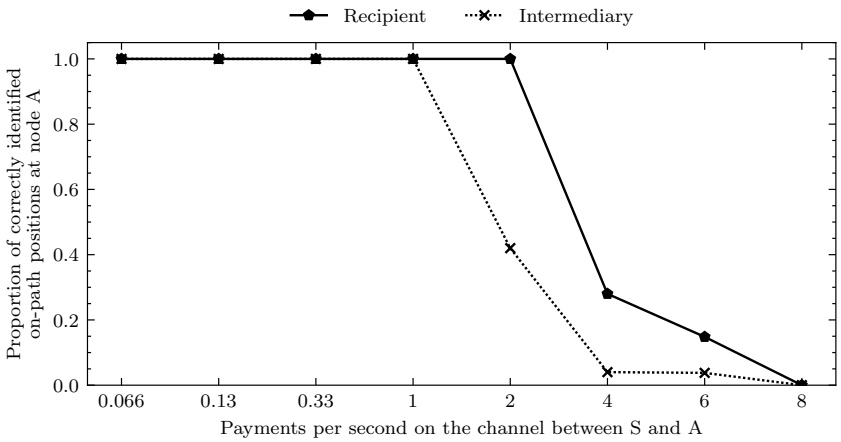

Figure 8 The success rate of identifying the monitored node's role in a payment, i.e., whether the node is the recipient or an intermediate routing hop.

4.3.3 Position identification

The concluding performance-related measurements studied the outcome of identifying a node's position in a payment path for different transaction rates. While there are no studies on the network's throughput, the channel-wise transaction rate is estimated to be rather low, e.g., 0.000019 payments per second based on reports by a central routing node [43]. To that end, we sent payments at different rates that we believe to be realistically achievable in the LN. The payments were issued in the private network due to the high volume.

For each of the different transaction rates under study, S issued 50 payments to A and 50 payments to D. We recorded A's true position in each payment's path as well as the `netfilter` program's verdict on the its position. We omit the classification of the different messages as it remains possible even at higher transaction rates without significant effort.

The proportion of routing positions correctly identified using the methodology described in Section 3.3 is depicted in Figure 8 for a varying number of payments per second between nodes S and A. At a rate of up to 1 payment per second the program correctly identified A's position in a payment path in all cases. However, as the rate increases beyond 1 payment per second, the accuracy gradually declines and ultimately falls to zero at 8 payments per second. This is because of the shorter intervals between messages which make it harder to distinguish whether messages are related or not. It is worth noting that correctly uncovering a node's position when it is the recipient is slightly more robust to higher payment rates. A transaction rate of 1 payment per second is indeed very low, however, we remark that it is still significantly higher than current estimates of LN's throughput.

These results show that, as long as the network's throughput does not increase drastically, the attack can be executed accurately.

4.4 Global impact

Naturally, we did not perform any measurements on the public network. Instead, and similar to multiple previous works [6, 26, 43], we simulated the attack using a snapshot of the channel graph obtained from a fully-synced LND node on 12 January 2024. We extended the LN simulator from [26] with some networking logic in order to map nodes to their corresponding

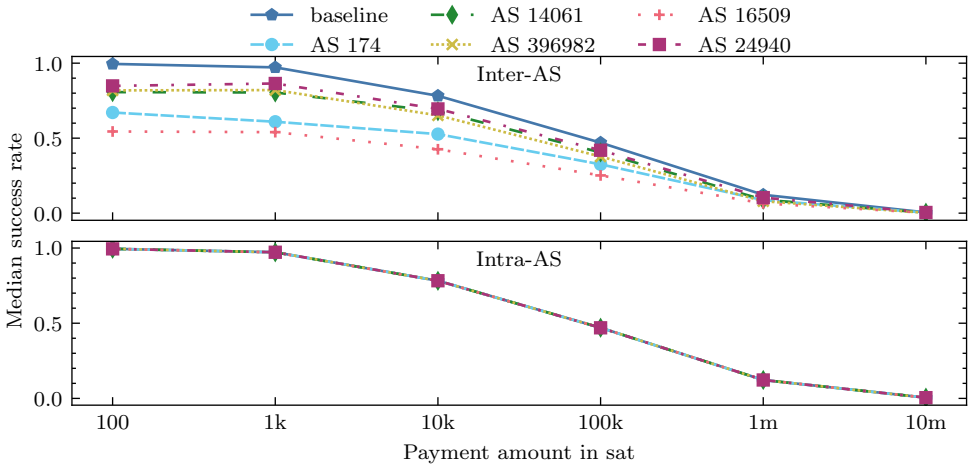

Figure 9 The median success rate when each of the top five ASs forbid either all inter-AS payments or all intra-AS payments.

AS[7], as well as the ability to simulate node failures. The code is publicly available on GitHub.[8] As actual payment volumes in LN are unknown, we simulated various payment volumes following the categorisation in [12] ranging from 100 sat to $10m$ sat. We simulated a set of $1,000$ payments between random sender-receiver pairs for each of the selected amounts. In order to measure the impact of censorship by a malicious AS on the LN, we simulated sending the payments in two different adversarial scenarios: when an AS allows either only local payments, i.e., intra-AS payments, or only payments involving at least one other AS, i.e., inter-AS payments. As shown in Section 4.2.1, a network-level adversary is able to determine a node's role in payment path, and we can thus simulate selective censorship. The ASs were ranked based on the number of channels and the top five were selected. We repeated each simulation scenario ten times with different seeds for the random number generator, i.e., for each set of $1,000$ sender-receiver pairs and for each AS, the channel graph was reinitialised before simulating payment delivery for each of the selected amounts.

The median success rate, i.e., the ratio of successful payments and the total number of payments, for all conducted simulations is shown in Figure 9. Besides observing what is already known in regard to the inverse relation between the success rate and payment amount [6, 26], the results clearly suggest that most payments in the LN are made between different ASs. Bearing the low proportion of intra-AS channels in mind (cf. Figure 2), it is not surprising that a significant amount of payments are affected by inter-AS censorship. The impact of the attack varies depending on the choice of malicious AS, e.g., AS 16509 causes a decrease of up to 45% while AS 24940 results in a drop of "only" up to 18% in the success rate. In contrast, when an AS only blocks payments within their network, the difference in the success rates is minimal suggesting that the impact on the greater network is negligible.

These results indicate that an adversary can block payments within their area of jurisdiction without causing significant harm to the wider network. On the other hand, the effects of a malicious adversary blocking payments being routed via their network would

[7] We used the GeoLite2 data from MaxMind, available at `https://www.maxmind.com`.
[8] `https://github.com/tud-dud/lightning-censorship-simulator`

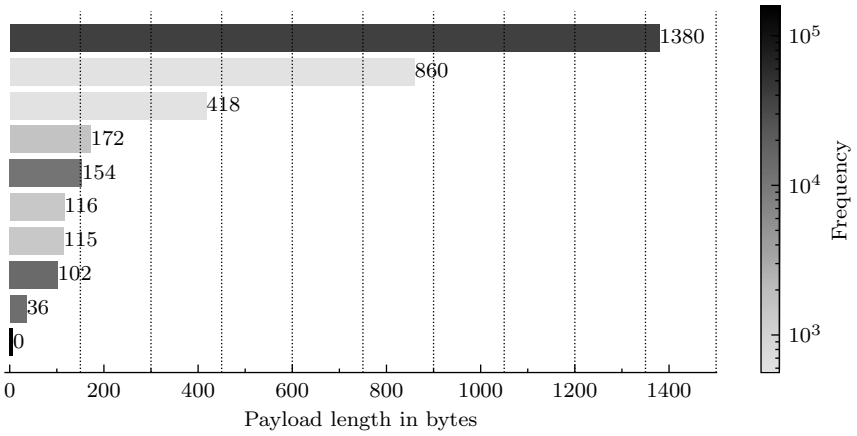

Figure 10 The ten most frequent Lightning payload lengths transmitted to/from an LND node in the testnet over 24 hours. The $18B$ packets LND nodes send are not included. The vertical lines illustrate the resulting number of packets if the payloads are padded and chunked to $150B$.

be very adverse for the network. The results of the other studied metrics such as fees and path lengths generally show little to no variation to the baseline simulation regardless of the applied dropping strategy. The charts have thus been omitted due to space constraints.

5 Countermeasures

In what follows, we discuss different measures the LN can implement in order to impede and/or mitigate network-level monitoring attacks. The authors of [43] propose a shift from a default port in LN to a pairwise-negotiated port in order to thwart port-based traffic filtering. Deviating from port 9735 is a stopgap which does not provide a suitable mitigation, but adds a layer of complexity to the attack that must be overcome. We argue that this alone is not adequate as, assuming the adversary is able to determine the new port, the attack can still be executed without any change. The new port can, for instance, be discovered using public crawl data, or by simply operating an LN node as each node stores the current topology locally. They also propose to "avoid adversarial ASs" by using third-party network services, e.g., Tor and VPNs, and implementing AS-aware routing. We argue that a VPN does not offer sufficient protection as it simply transfers the risk from one AS to another.

Briefly recapped, the core of the attack presented in this work exploits two side channels – payload size and timing information – to allow a network-level adversary to identify the different LN messages despite encryption. Arx et al. suggest hiding the lengths of the application data but do not provide specifics on a plausible padding strategy [43]. The reasoning behind employing a length-hiding scheme is that the network-level classification attacks rely on the TCP payload lengths to identify messages. It is, however, not clear which strategy is best suited for the LN.

5.1 Weighing the options

We recorded the lengths of all the TCP packets on port 9735 during the 24-hour time period in which the measurements in Section 4 were performed and depict the observations in Figure 10. As evident in Figure 10, the payload sizes of LN messages differ wildly. Consequently, finding a common length is not trivial. Simply padding all payloads to the

maximal length would result in a significant waste of bandwidth. Instead, we could chunk the data following a block-length padding strategy [14, 23], i.e., padding to the closest multiple of x bytes. As a result, a network-level adversary would only observe constant-length LN payloads. Nevertheless, as LN messages have a specified length, the adversary can still make use of the other side channel – the timing information – to classify the encrypted TCP payloads. All the attacker needs to do is map the messages sizes in Table 1 to multiples of x. Observing the direction of flow, number of packets and sequence still gives clear indications of the underlying messages in transit. For the sake of argumentation, let us assume that x is somewhat arbitrarily set to $150B$, i.e., all LN messages are sent in $150B$ chunks. As visualised in Figure 10, all but the first message exchanged during the HTLC commitment phase would be identical on the network layer (cf. Table 1). Identifying the application messages is then no longer possible by simply inspecting the observed packet sizes. However, as we know both the type and order of messages involved in the process, we know how many packets correspond to each of the messages. An adversary must therefore additionally keep a count of packets which adds a minimal layer of complexity to the attack. The perhaps most obvious telltale sign is the `update_add_htlc` message ($1450B$) that would be sent in ten packets followed by two packets for the `commitment_signed` ($162B$) in the same direction. Regardless of this weakness, other message types not discussed in this work would also need to be taken into consideration in order to define a meaningful chunk size.

If we turn our attention to the timing information, we realise that it is even more delicate. For instance, we cannot simply reorder messages while conforming to the protocol specifications. Techniques such as adaptive padding [36] which inject dummy packets into the packet flow thus become relevant. This destroys timing fingerprints without any additional latency. However, adaptive padding on its own is not an adequate countermeasure for the LN as the other side channel – message size – remains unaddressed. For similar reasons, transmitting packets at a constant rate [11] is not sufficient on its own either. Currently, Tor implements a variation of adaptive padding as a defence against website fingerprinting (WF) attacks derived from the Website Traffic Fingerprinting Protection with Adaptive Defence (WTF-PAD) [18] mechanism. In summary, WTF-PAD sends dummy data such that an attacker cannot tell real data apart from fake data based on expected packet inter-arrival times. Furthermore, since all traffic in Tor is padded to $514B$ cells, WTF-PAD impedes the effectiveness of WF attacks in Tor by obfuscating timing patterns.

5.2 Towards a solution

We examined whether the variant of padding that is implemented in Tor would provide sufficient protection in the LN against the attack at hand. We did so by configuring two LND nodes to connect to each other over Tor and opening a channel between them. The purpose of doing so is to utilise Tor's implementation of WTF-PAD and not for Tor's privacy properties. We issued payments in both directions, closed the channel and finally the TCP connection. Not only did all packets have the same packet length (as is expected when using Tor), but the flow of transmitted packets included packets that did not originate from the application. Consequently, we were not able to detect which packets belonged to which Lightning message by manually inspecting the capture. The rule-based state machine is therefore no longer capable of distinguishing application messages based on the network traces alone. In fact, we conjecture that this approach offers a high degree of protection for the LN against more sophisticated fingerprinting techniques by network-level adversaries as basically all size and timing features are destroyed.

Although we have established that the mechanisms implemented in Tor offer sufficient protection, the question of how much this protection costs remains unanswered. In order to get an approximation of the cost of using Tor, we captured all packets while executing the above operations in a thirty-minute time frame. In addition to the aforementioned deliberate activity, the time frame also includes periods in which only control messages were sent by the nodes, e.g., when the blockchain advances or health checks. Specifically, we concurrently captured the packets sent locally between the LND node and the Tor SOCKS5 proxy, as well as the packets sent between the Tor process and Tor network. The former provides data on the packets that actually come from the application while the latter provides data on what a network-level attacker would observe. The captures show a total of 14,824 bytes transmitted in 379 TCP packets to/from LND and 929,596 bytes in 3191 TCP packets to/from the Tor network. This equates to an increase of $\approx 6170\%$ in bandwidth when using Tor. The captures also show a peak rate of 0.116 Mbit/s when using Tor, which clearly should not cause any problems for LN nodes while maintaining their current hardware configurations. However, we note that these are overestimations of the actual overhead to expect in the LN as they include traffic in Tor that is not actually relevant to mitigating network-level message identification in the LN, e.g., circuit management. We therefore do not consider the universal usage of Tor in the LN to be the solution; the overhead of a standalone implementation of WTF-PAD in the LN is expected to be much lower. Besides, Tor nodes are susceptible to other potential threats [20,27,39] and using Tor implies higher latency in order to provide features that may not be required by all nodes in the LN.

An effective mitigation strategy for the LN must omit both the timing and size information. Obfuscating either properties is further complicated by the fact that crucial LN operations, e.g., channel opening or HTLC commitment, must follow an order defined in the protocol. This means that message flows between two Lightning nodes often follow deterministic patterns. In view of the preceding discussion, we recommend that the LN adopts a form of adaptive padding similar to Tor as a defence against network-layer monitoring attacks. That is, not only must we conceal all packet sizes on the network layer, we must also obfuscate the timing patterns in the P2P communication. Our assessments of the attack's feasibility over Tor demonstrate that fixed-length packets in conjunction with cover traffic effectively hamper the attack. While this solution will necessarily introduce a degree of overhead, the LN may be facing a technical version of pick your poison.

6 Related Work

6.1 Censorship

Internet censorship has been the subject of multiple works, e.g., [4,45,46], due to some of the extensive censorship currently imposed in various parts of the world. It is thus a highly relevant topic. P2P networks are generally considered to be more resistant to censorship than classic server-client networks as a result of their fundamental architectural differences. Nonetheless, there have been reports on the feasibility of imposing censorship in blockchain-based P2P networks such as Bitcoin [22] and Ethereum [44] by, for instance, exploiting application-level protocol designs. A prominent example of a state-imposed censorship in the realm of digital payment networks is the complete trading and mining ban in China. In a recent work by Sridhar et al., the authors present a censorship attack in the InterPlanetary File System (IPFS) [38] – a popular P2P content delivery network.

6.2 Network-level Attacks in the Lightning network

To the best of our knowledge, the LN's network layer has not received significant attention so far. In one of the few works, Casas et al. [9] analysed the P2P network and found that a significant number of nodes connects to the LN through Tor. An analysis conducted in [47] established a degree of geographic clustering among the nodes. The authors of [43] study attacks on the LN's network layer and show that it is possible to decipher encrypted LN messages via traffic analysis. Besides pursuing a different goal, our work not only confirms their findings but also refines the information an adversary can gain from traffic analysis. This additional information is what enables an adversary to impose selective censorship based on the payment's source and/or destination. Furthermore, our presented attack is based on real-time traffic monitoring and execution in contrast to [43]. There has also been research on AS-level side channel attacks on privacy and routing in the broad spectrum of cryptocurrency networks [3, 32, 37, 42] and anonymisation networks such as Tor [27, 39].

6.3 Lightning network Topology and Simulations

Numerous works have studied the structural properties of the Lightning channel graph and demonstrated that it is highly centralised [6, 34, 41] at the application level, e.g., a small number of nodes function as essential routing nodes due to their high centrality in the graph. As a result, it is susceptible to a variety of attacks on privacy and security [19, 31, 40]. The analysis of the channel graph's network level in [43] revealed that it is equally centralised and vulnerable to attacks on payment privacy. Our topological analysis of the channel graph complements existing ones and provides new insights on its network-level structure, e.g., most channels in the network are between distinct pairs of ASs.

A broad range of research on LN takes a simulation-based approach, e.g., [6, 19, 31], to analyse their studies' significance for the public mainnet. Simulations are often necessary in order to not interact with third-party nodes in the public network. Due to the availability of multiple open-source LN simulators, we did not develop a new simulator but instead extended an existing one [26] with the relevant functionality for this work.

7 Conclusion and Future Work

We studied potential censorship attacks in the Lightning network founded on monitoring network-level traffic. Furthermore, we demonstrated that it is feasible to determine a node's position in a payment path based on the observed traffic. In doing so, our work highlights the threat powerful adversaries such as autonomous systems pose to the Lightning network which is further heightened by the network-level centralisation. Based on our analysis of potential countermeasures, we conclude that an effective mitigation strategy in the LN inevitably implies some bandwidth overhead.

The attack presented in this work exploits two side channels at the network layer – payload size and timing patterns. We think that studying effective and efficient mitigation strategies is an interesting and relevant research question. Complementary to mitigation strategies, developing mechanisms to detect censorship is a similarly relevant question for future work. Additionally, and like in multiple other previous works, estimates used in this work with respect to the network's throughput relied on the occasional reports provided by node operators. Acknowledging that measuring throughput in a public P2P network is not straightforward, we believe that future research on Lightning would benefit from well-founded assessments of the network's throughput.

References

1 Bolt: Basis of lightning technology (lightning network specifications). URL: https://github.com/lightning/bolts.

2 XDP - IO Visor Project. https://www.iovisor.org/technology/xdp, 2016. [Accessed 01/02/2024].

3 Maria Apostolaki, Cedric Maire, and Laurent Vanbever. Perimeter: A network-layer attack on the anonymity of cryptocurrencies. In *Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part I*, volume 12674 of *Lecture Notes in Computer Science*, pages 147–166. Springer, 2021. doi:10.1007/978-3-662-64322-8_7.

4 Simurgh Aryan, Homa Aryan, and J. Alex Halderman. Internet censorship in iran: A first look. In *3rd USENIX Workshop on Free and Open Communications on the Internet (FOCI 13)*, Washington, D.C., 2013. USENIX Association. URL: https://www.usenix.org/conference/foci13/workshop-program/presentation/aryan.

5 Michael D. Bailey, David Dittrich, Erin Kenneally, and Douglas Maughan. The menlo report. *IEEE Secur. Priv.*, 10(2):71–75, 2012. doi:10.1109/MSP.2012.52.

6 Ferenc Béres, István András Seres, and András A. Benczúr. A cryptoeconomic traffic analysis of bitcoins lightning network. *CoRR*, 2019. arXiv:1911.09432.

7 Gilberto Bertin. Xdp in practice: integrating xdp into our ddos mitigation pipeline. In *Technical Conference on Linux Networking, Netdev*, volume 2, pages 1–5. The NetDev Society, 2017.

8 Kevin Bock, George Hughey, Xiao Qiang, and Dave Levin. Geneva: Evolving censorship evasion strategies. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2199–2214. Association for Computing Machinery, 2019. doi:10.1145/3319535.3363189.

9 Pedro Casas, Matteo Romiti, Peter Holzer, Sami Ben Mariem, Benoit Donnet, and Bernhard Haslhofer. Where is the light(ning) in the taproot dawn? unveiling the bitcoin lightning (IP) network. In *10th IEEE International Conference on Cloud Networking, CloudNet 2021, Cookeville, TN, USA, November 8-10, 2021*, pages 87–90. IEEE, 2021. doi:10.1109/CLOUDNET53349.2021.9657121.

10 George Danezis and Ian Goldberg. Sphinx: A compact and provably secure mix format. In *30th IEEE Symposium on Security and Privacy (SP 2009), 17-20 May 2009, Oakland, California, USA*, pages 269–282. IEEE Computer Society, 2009. doi:10.1109/SP.2009.15.

11 Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail. In *2012 IEEE Symposium on Security and Privacy*, pages 332–346, 2012. doi:10.1109/SP.2012.28.

12 Oguzhan Ersoy, Stefanie Roos, and Zekeriya Erkin. How to profit from payments channels. In *Financial Cryptography and Data Security - 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10-14, 2020 Revised Selected Papers*, volume 12059 of *Lecture Notes in Computer Science*, pages 284–303. Springer, 2020. doi:10.1007/978-3-030-51280-4_16.

13 Arturo Filastò and Jacob Appelbaum. OONI: Open observatory of network interference. In *2nd USENIX Workshop on Free and Open Communications on the Internet (FOCI 12)*. USENIX Association, 2012. URL: https://www.usenix.org/system/files/conference/foci12/foci12-final12.pdf.

14 Daniel Kahn Gillmor. Empirical dns padding policy. https://dns.cmrg.net/ndss2017-dprive-empirical-DNS-traffic-size.pdf, 2017. [Accessed 03/04/2024].

15 Nguyen Phong Hoang, Arian Akhavan Niaki, Jakub Dalek, Jeffrey Knockel, Pellaeon Lin, Bill Marczak, Masashi Crete-Nishihata, Phillipa Gill, and Michalis Polychronakis. How great is the great firewall? measuring china's DNS censorship. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3381–3398. USENIX Association, 2021. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/hoang.

16 Toke Høiland-Jørgensen, Jesper Dangaard Brouer, Daniel Borkmann, John Fastabend, Tom Herbert, David Ahern, and David Miller. The express data path: fast programmable packet processing in the operating system kernel. In *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies, CoNEXT 2018, Heraklion, Greece, December 04-07, 2018*, pages 54–66. ACM, 2018. doi:10.1145/3281411.3281443.

17 OpenNet Initiative. Turkmenistan, December 2010. URL: https://opennet.net/research/profiles/turkmenistan.

18 Marc Juarez, Mohsen Imani, Mike Perry, Claudia Díaz, and Matthew Wright. Toward an efficient website fingerprinting defense. In *Computer Security - ESORICS 2016 - 21st European Symposium on Research in Computer Security, Heraklion, Greece, September 26-30, 2016, Proceedings, Part I*, volume 9878 of *Lecture Notes in Computer Science*, pages 27–46. Springer, 2016. doi:10.1007/978-3-319-45744-4_2.

19 George Kappos, Haaroon Yousaf, Ania M. Piotrowska, Sanket Kanjalkar, Sergi Delgado-Segura, Andrew Miller, and Sarah Meiklejohn. An empirical analysis of privacy in the lightning network. In *Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part I*, volume 12674 of *Lecture Notes in Computer Science*, pages 167–186. Springer, 2021. doi:10.1007/978-3-662-64322-8_8.

20 Albert Kwon, Mashael AlSabah, David Lazar, Marc Dacier, and Srinivas Devadas. Circuit fingerprinting attacks: Passive deanonymization of tor hidden services. In *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015*, pages 287–302. USENIX Association, 2015. URL: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/kwon.

21 Jian-Hong Lin, Kevin Primicerio, Tiziano Squartini, Christian Decker, and Claudio J. Tessone. Lightning network: a second path towards centralisation of the bitcoin economy. *CoRR*, 2020. arXiv:2002.02819.

22 Angelique Faye Loe and Elizabeth Anne Quaglia. You shall not join: A measurement study of cryptocurrency peer-to-peer bootstrapping techniques. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 2231–2247. ACM, 2019. doi:10.1145/3319535.3345649.

23 A. Mayrhofer. Padding policies for extension mechanisms for dns (edns(0)). https://datatracker.ietf.org/doc/html/rfc8467, 2018. [Accessed 03/04/2024].

24 Ayelet Mizrahi and Aviv Zohar. Congestion attacks in payment channel networks. In *Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part II*, volume 12675 of *Lecture Notes in Computer Science*, pages 170–188. Springer, 2021. doi:10.1007/978-3-662-64331-0_9.

25 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. URL: https://nakamotoinstitute.org/bitcoin/.

26 Charmaine Ndolo and Florian Tschorsch. On the (not so) surprising impact of multi-path payments on performance and privacy in the lightning network. In *Computer Security. ESORICS 2023 International Workshops - CyberICS, DPM, CBT, and SECPRE, The Hague, The Netherlands, September 25-29, 2023, Revised Selected Papers, Part I*, volume 14398 of *Lecture Notes in Computer Science*, pages 411–427. Springer, 2023. doi:10.1007/978-3-031-54204-6_25.

27 Rishab Nithyanand, Oleksii Starov, Phillipa Gill, Adva Zair, and Michael Schapira. Measuring and mitigating as-level adversaries against tor. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. The Internet Society, 2016. URL: http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2017/09/measuring-mitigating-as-level-adversaries-against-tor.pdf.

28 Sadia Nourin, Van Tran, Xi Jiang, Kevin Bock, Nick Feamster, Nguyen Phong Hoang, and Dave Levin. Measuring and evading turkmenistan's internet censorship: A case study in large-scale measurements of a low-penetration country. In *Proceedings of the ACM Web Conference 2023*, WWW '23, pages 1969–1979. Association for Computing Machinery, 2023. doi:10.1145/3543507.3583189.

29 Trevor Perrin. The noise protocol framework. https://noiseprotocol.org/noise.pdf, 2018. [Accessed 05/04/2024].

30 Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments, January 2016. URL: https://lightning.network/lightning-network-paper.pdf.

31 Elias Rohrer and Florian Tschorsch. Counting down thunder: Timing attacks on privacy in payment channel networks. In *AFT '20: 2nd ACM Conference on Advances in Financial Technologies, New York, NY, USA, October 21-23, 2020*, pages 214–227. ACM, 2020. doi:10.1145/3419614.3423262.

32 Muhammad Saad and David Mohaisen. Three birds with one stone: Efficient partitioning attacks on interdependent cryptocurrency networks. In *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*, pages 111–125. IEEE, 2023. doi:10.1109/SP46215.2023.10179456.

33 Dominik Scholz, Daniel Raumer, Paul Emmerich, Alexander Kurtz, Krzysztof Lesiak, and Georg Carle. Performance implications of packet filtering with linux ebpf. In *30th International Teletraffic Congress, ITC 2018, Vienna, Austria, September 3-7, 2018 - Volume 1*, pages 209–217. IEEE, 2018. doi:10.1109/ITC30.2018.00039.

34 István András Seres, László Gulyás, Dániel A. Nagy, and Péter Burcsi. Topological analysis of bitcoin's lightning network. In *Mathematical Research for Blockchain Economy, 1st International Conference, MARBLE 2019, Santorini, Greece, May 6-9, 2019*, pages 1–12. Springer, 2019. doi:10.1007/978-3-030-37110-4_1.

35 RFE/RL's Turkmen Service. Internet in turkmenistan, already the world's slowest, faces further restrictions, January 2022. URL: https://www.rferl.org/a/turkmenistan-internet-slowest-restrictions/31652467.html.

36 Vitaly Shmatikov and Ming-Hsiu Wang. Timing analysis in low-latency mix networks: Attacks and defenses. In *Computer Security - ESORICS 2006, 11th European Symposium on Research in Computer Security, Hamburg, Germany, September 18-20, 2006, Proceedings*, volume 4189 of *Lecture Notes in Computer Science*, pages 18–33. Springer, 2006. doi:10.1007/11863908_2.

37 Paulo Silva. Impact of geo-distribution and mining pools on blockchains: A study of ethereum - practical experience report and ongoing phd work. In *50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks, DSN 2020, Valencia, Spain, June 29 - July 2, 2020 - Supplemental Volume*, pages 73–74. IEEE, 2020. doi:10.1109/DSN-S50200.2020.00039.

38 Srivatsan Sridhar, Onur Ascigil, Navin V. Keizer, François Genon, Sébastien Pierre, Yiannis Psaras, Etienne Rivière, and Michal Król. Content censorship in the interplanetary file system. *CoRR*, 2023. doi:10.48550/arXiv.2307.12212.

39 Yixin Sun, Anne Edmundson, Laurent Vanbever, Oscar Li, Jennifer Rexford, Mung Chiang, and Prateek Mittal. RAPTOR: routing attacks on privacy in tor. In *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015*, pages 271–286. USENIX Association, 2015. URL: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/sun.

40 Sergei Tikhomirov, Pedro Moreno-Sanchez, and Matteo Maffei. A quantitative analysis of security, anonymity and scalability for the lightning network. In *IEEE European Symposium on Security and Privacy Workshops, EuroS&P Workshops 2020, Genoa, Italy, September 7-11, 2020*, pages 387–396. IEEE, 2020. doi:10.1109/EUROSPW51379.2020.00059.

41 Saar Tochner, Stefan Schmid, and Aviv Zohar. Hijacking routes in payment channel networks: A predictability tradeoff. *CoRR*, 2019. arXiv:1909.06890.

42 Florian Tramèr, Dan Boneh, and Kenny Paterson. Remote side-channel attacks on anonymous transactions. In *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, pages 2739–2756. USENIX Association, 2020. URL: https://www.usenix.org/conference/usenixsecurity20/presentation/tramer.

43 Theo von Arx, Muoi Tran, and Laurent Vanbever. Revelio: A network-level privacy attack in the lightning network. In *8th IEEE European Symposium on Security and Privacy, EuroS&P 2023, Delft, Netherlands, July 3-7, 2023*, pages 942–957. IEEE, 2023. doi:10.1109/EUROSP57164.2023.00060.

44 Anton Wahrstätter, Jens Ernstberger, Aviv Yaish, Liyi Zhou, Kaihua Qin, Taro Tsuchiya, Sebastian Steinhorst, Davor Svetinovic, Nicolas Christin, Mikolaj Barczentewicz, and Arthur Gervais. Blockchain censorship. In *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*, pages 1632–1643. ACM, 2024. doi:10.1145/3589334.3645431.

45 Xueyang Xu, Zhuoqing Morley Mao, and J. Alex Halderman. Internet censorship in china: Where does the filtering occur? In *Passive and Active Measurement - 12th International Conference, PAM 2011, Atlanta, GA, USA, March 20-22, 2011. Proceedings*, volume 6579 of *Lecture Notes in Computer Science*, pages 133–142. Springer, 2011. doi:10.1007/978-3-642-19260-9_14.

46 Tarun Kumar Yadav, Akshat Sinha, Devashish Gosain, Piyush Kumar Sharma, and Sambuddho Chakravarty. Where the light gets in: Analyzing web censorship mechanisms in india. In *Proceedings of the Internet Measurement Conference 2018*, IMC '18, pages 252–264. Association for Computing Machinery, 2018. doi:10.1145/3278532.3278555.

47 Philipp Zabka, Klaus-Tycho Förster, Stefan Schmid, and Christian Decker. Node classification and geographical analysis of the lightning cryptocurrency network. In *ICDCN '21: International Conference on Distributed Computing and Networking, Virtual Event, Nara, Japan, January 5-8, 2021*, pages 126–135. ACM, 2021. doi:10.1145/3427796.3427837.