Revelio: A Network-Level Privacy Attack in the Lightning Network

Revelio: A Network-Level Privacy Attack in the Lightning Network

Theo von Arx
ETH Zürich

Muoi Tran
ETH Zürich

Laurent Vanbever
ETH Zürich

***Abstract*—The Lightning Network (LN) is a widely-adopted off-chain protocol that not only addresses Bitcoin's scaling problem but also enables anonymous payments. Prior attacks have shown that an adversary controlling several peers at the central position of the network (e.g., by hijacking payment routes) can deanonymize such payments. However, these attacks are highly observable or require many parties to collude.**

This paper presents Revelio, a stealthier, passive network-level privacy attack against LN that exploits its joint centralization at the application and the network layers. Indeed, network-level adversaries can see most of the LN traffic (e.g., five autonomous systems can see up to 80 % of all observable communication channels) despite the encrypted communication between LN nodes and the widespread usage of Tor. This comprehensive view allows Revelio adversaries not only to estimate the payment amount but also to effectively reduce the anonymity size of its endpoints. We show that the Revelio attack is practical: it perfectly deanonymizes the senders or the receiver in almost one-third of tested payments in today's LN and underlying network topologies.

1. Introduction

Bitcoin is, *by far*, the most-used and most-successful cryptocurrency to date, with a market capitalization well above 400 billion USD as of March 2023. One of the key appeals of Bitcoin is decentralization: it relies on no single authority but a sizeable peer-to-peer network of distributed nodes that use consensus algorithms to record transactions into a core data structure known as the blockchain. Bitcoin's decentralization, however, comes at the cost of poor scalability and reduced privacy. Infamously, Bitcoin only supports about seven transactions per second, whereas centralized financial services like Visa and Mastercard process about 40,000 transactions per second at peak times [63]. Moreover, Bitcoin transactions are only pseudo-anonymous and publicly available via the blockchain [13].

The Lightning Network (LN) is an overlay network that aims to address Bitcoin scalability and privacy problems [57]. It enables split-second payments (up to 1,000,000 per second) to be made anonymously, meaning that no one (except the sender) can determine the sender and the receiver of a payment. The critical factor for achieving high transaction rates is the drastic reduction in the number of required records written to the Bitcoin blockchain. More specifically, LN nodes use *off-chain* payment channels to transact as often as needed and only write the proofs of opening or closing these channels to the blockchain. The payments in LN are commonly routed over multiple channels, and the privacy of their information (e.g., sender, receiver, and amount) is provided by an onion-based encryption scheme. LN is particularly popular and is widely used, boasting more than 17,000 nodes as of this writing.

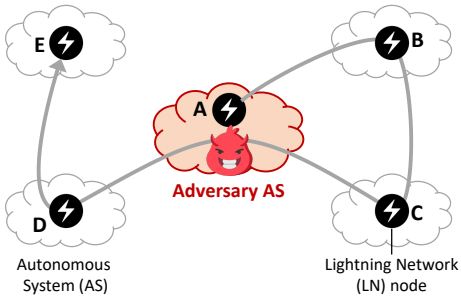

Figure 1. An example of the Revelio attack against onion-based multi-hop payments in Lightning Network (LN). To deanonymize the payment from A to E, an adversary AS correlates network packets from $A \rightarrow B$ and $C \rightarrow D$ based on their timestamps and infers E from public datasets.

Given LN's popularity, multiple attacks against its payment privacy guarantees have been proposed [34], [36], [53], [59], [66], [70]. Thus far, these attacks rely on the adversary's ability to actively run LN nodes and forward transactions. By observing enough channels, the adversary can deanonymize the sender and the receiver of a routed payment. While running malicious nodes is straightforward, maximizing the number of payments they see is not and requires active attacks (e.g., hijacking payment routes [72]). These attacks are, therefore, highly observable. Moreover, they cannot deanonymize single-hop payments.

This paper shows that deanonymization attacks in LN can be both passive (thus, stealthier) and applicable to single-hop payments (thus, more inclusive) with *network-level adversaries*, such as malicious Autonomous Systems (ASes). To that end, we describe *Revelio*, a novel network-level attack that breaks the payment anonymity guarantees of LN. Unlike existing attacks that require running many LN nodes, Revelio adversaries deanonymize LN payments by passively analyzing their network traffic. As the example in Figure 1 shows, LN traffic belonging to the same multi-hop payment may cross an adversarial AS multiple times, either because several nodes are hosted in this AS or because of Internet routing. This broad view allows Revelio adversaries to correlate application-level messages transiting (respectively not transiting) through their network and infer the payment's sender, receiver, and amount despite the LN traffic being encrypted. Understandably, the power of a Revelio adversary depends on

the number of channels she sees. Unfortunately, because the LN topology is centralized at both the application- [44] and the network-level [17], powerful attackers exist. In particular, our updated and more comprehensive analysis shows that a single AS, such as Amazon, can see up to $34\,\%$ of all observable LN channels; and that the top 5 ASes together can see up to $80\,\%$ of all observable LN channels.

Revelio is the first AS-level privacy attack against LN, to the best of our knowledge; yet, network adversaries have been a worrying threat in a broader scope for a long time. While being active, AS attackers can launch BGP hijack [65] to bypass laws [6], obtain bogus Public Key Infrastructure (PKI) certificates [11], compromise Tor's anonymity [68], or disrupt blockchain consensus [4], [62]. Active network attacks also include traffic dropping for censorship attacks [74] or traffic injection for DDoS attacks [47] and adding advertisements [51]. Without using such active capabilities, AS-level attackers (e.g., nation-state adversaries) can still passively analyze network traffic to deanonymize Tor users [26], [33], launch nation-wide surveillance attacks [2], [27], or reveal the transaction history of cryptocurrency users [3] (e.g., for tax enforcement [18] or prosecution [8]). These prior works suggest the threat of network attacks against LN; thus, studying them is crucial.

Deanonymizing LN payments with network adversaries, however, poses some unique challenges. First, the communication between LN is encrypted, and most LN nodes run Tor to hide their IP addresses. We demonstrate that Revelio adversaries can still learn the types of LN messages being transmitted and the communication endpoints as long as at least one of them is not behind Tor. Second, multiple transactions may have their traffic crossing the adversarial AS together, making the network packet correlation difficult. We show how Revelio adversaries can use the arrival timestamps of the packets to distinguish payments, which is indeed possible for realistic transaction throughputs. Third, Revelio adversaries often do not see all channels involved in a payment but only a subset of them. Reconstructing the complete path from these partial observations requires a computationally-intensive process of inverting LN path computations because varying payment amounts result in different paths. To resolve this, we propose an efficient path reconstruction procedure based on binary search, considerably speeding up path calculations. In particular, we show how a Revelio attacker can estimate a narrower range of possible payment amounts purely based on observed channels and their public parameters.

Our evaluations with the live LN network and simulations show that the Revelio attack is highly effective. In the median case, network-level attackers manage to reduce the range of possible payment amounts by five orders of magnitude. Worse yet, we show that the top 5 ASes can reduce the anonymity sets to one sender (respectively receiver) for almost one-third of observed payments.

To sum up, we make the following key contributions:

- We analyze the LN topology from a network-level perspective and show that the five most central ASes on the LN topology can already observe up to $80\,\%$ of all channels (§3).
- We propose Revelio, an AS-level attack that captures LN traffic, groups them into multi-hop payments, and deanonymizes the payment sender, receiver, and amount (§4, §5, §6).
- We show that five colluding ASes can reduce the sender or receiver anonymity set size from more than $17,000$ to 1 in more than 32% of cases (§7).
- We discuss both short- and long-term countermeasures to our attack and sketch possible future works (§8).

2. Background: Lightning Network

The Lightning Network (LN) [57] is a protocol that allows two users with dedicated clients (e.g., lnd [39], eclair [1], CLN [19], electrum [76]) to make arbitrary off-chain transactions via a channel secured by an on-chain Bitcoin transaction. In particular, two LN parties establish a channel by depositing their funds in a 2-of-2 multi-signature Bitcoin contract. The *capacity* of the channel is the sum of two initial funds. After that, two channel endpoints may make payments to each other, which change their funds (or *balances*) accordingly. They can close the channel with signatures from both parties, and the latest balances will be returned to them. By taking most of the transactions off-chain and writing only transactions for channel opening and closing to the blockchain, LN significantly improves the transaction throughput in Bitcoin. Note that the LN network is entirely independent of the Bitcoin network, although a user may run both LN and Bitcoin clients, and LN nodes typically connect to Bitcoin nodes to receive updated blockchain information.

LN nodes form a peer-to-peer (P2P) network when establishing channels with each other. Nodes use a gossiping protocol to share updated information with others in this P2P network, such as their addresses or newly-established channels. By doing so, nodes are quickly updated with the entire LN topology. Each LN node is identified by a public key, called *node ID*, and optionally associated with one or more IPv4, IPv6, or .onion addresses. When a node does not advertise any address to the P2P network, it is considered as *unknown*. Unknown nodes do not receive channel initiations (perhaps, to avoid unwanted channels with unfamiliar nodes); however, they can still actively open channels to others.

Users without a direct channel still can transact using *multi-hop payments*, such as A to node E in Figure 1. The payment is first initiated when the receiver (e.g., E) generates an invoice including the hash $H(s)$ of a secret s, an amount amt in sat (i.e., *satoshi*) where $1\,\mathrm{sat} = 10^{-8}\mathrm{bitcoin}$[1], and sends it to the sender (e.g., A) via an out-of-band medium. Node A then computes a path to node E through one or more intermediate nodes (e.g., B, C, D), which impose a small fee for routing the payment. The selected routing path is usually the shortest and cheapest path from the sender to the receiver, although different LN clients implement the path calculation differently [72]. Also, the cheapest paths between two nodes may differ for different payments because parts of the

1. LN also supports msat (i.e., millisatoshi) where $1\,\mathrm{msat} = 10^{-3}\mathrm{sat}$. Still, it is only usable within LN, e.g., for calculating routing fee.

routing fees are proportional to the routed amounts. Next, the sender constructs an onion-based Sphinx [21] packet destined to the receiver that consists of the invoice (i.e., $H(s)$ and amt) and forwards it to the first node on the route. When an intermediate node (e.g., C) receives such an onion-based packet, it issues a Hashed-Time Locked Contract (HTLC) [57] to ensure the next-hop node (e.g., D) can redeem amt upon revealing the correct secret s. Once the receiver finds the correct information in the packet, it reveals the secret s so intermediate nodes can redeem amt sequentially in the reversed direction (e.g., D, C, then B). A multi-hop payment thus can be considered as consisting of multiple, independent payments between consecutive nodes along the path. If the above steps fail, the sender will receive failure errors from other nodes and have to retry with a different path.

Under the LN specifications (e.g., [40], [42], [43]), LN payments are designed to be not only scalable but also privacy-preserving. First, payments between two users are not broadcast to the LN network via the gossiping protocol but restricted only between them, hence becoming oblivious for off-path nodes. Moreover, LN enforces the onion-based routing for multi-hop payments to ensure that intermediate nodes cannot learn other nodes on the path except their predecessor and successor. At the network level, the transmitting packets are obfuscated under the Noise Protocol Framework so that packets of the same transaction between different hops do not share any correlating information. Without such efforts in protecting payment privacy, LN users would suffer from large-scale privacy invasion or surveillance in which their payment information (e.g., timing, amount, involved parties, correlated IP addresses) is directly exposed to adversaries. Note that existing privacy protections in Bitcoin (e.g., pseudo-anonymous address [35], transaction diffusion [25]) do not apply to LN payments because LN users do not interact with Bitcoin network when transacting off-chain.

3. The AS Topology of LN's P2P Network

In this section, we study the AS topology of the P2P network of LN using publicly-available data. We first present our methodology for constructing this AS topology (§3.1). We then analyze the topology, showing which ASes can intercept LN traffic and in which portion (§3.2). Our study complements existing work on the centralization of LN at the application layer (e.g., [58], [63], [79]) and shows that the LN is also centralized at the AS-level.

3.1. Methodology

Constructing the AS topology from the network of LN nodes requires mapping each LN node to the AS hosting it and a computation of the AS path between any two ASes. Here, we specify how to collect the necessary data and describe the AS topology construction.

Data collection. We first collected the addresses of all LN nodes by taking a snapshot of the entire LN topology on 29 June 2022 using an lnd client and its built-in describegraph command. Because all LN nodes share the identical view of the LN topology most of the time, taking a topology snapshot from a single node is sufficient, as demonstrated in previous works (e.g., [12], [20], [81]). This snapshot, which also is used throughout this paper, consists of 17,644 nodes in its most significant connected component and 80,932 channels in total. As LN nodes may run behind Tor, we downloaded the list of 1524 available Tor exit relays in June 2022, including their IP addresses and the probability of being chosen [69].

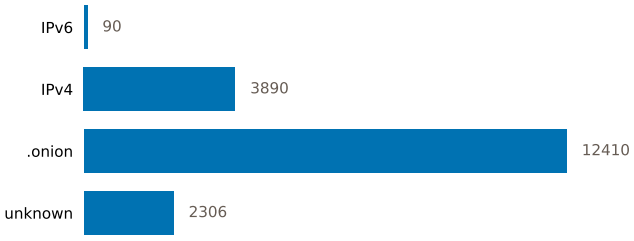

Figure 2. Number of announced address types. Most nodes try to hide their IP address: only 3890 out of 17,644 LN nodes announce an IPv4 address.

For mapping an IP address to its AS, we used the RouteViews IP-prefix-to-AS dataset [75] released in June 2022. At the same time, we also downloaded datasets for the AS path calculation, including the AS business relationships [15] and IXP links [16].[2] In total, this data includes 18,968 ASes, 1063 IXPs, and 6,729,747 links.

AS path calculation. The calculation of AS paths between two given IP addresses has been a research question for decades [46]. Without direct access to the source of the traffic (e.g., LN nodes), one can *estimate* the AS paths with a high accuracy [73], by simulating the de facto AS-level routing protocol, namely Border Gateway Protocol (BGP), at each AS. Following this approach, we estimate AS paths between the ASes hosting two given LN nodes. In particular, we build the AS-level topology of the Internet reflecting the business relationship (e.g., provider, customer, peer) between ASes [15]. After that, we simulate ASes calculate the forwarding paths to each other with the following widely-perceived BGP policies: (1) ASes prefer routing through customers over peers and peers over providers; (2) the shortest AS path is preferred; (3) an arbitrary method (e.g., smaller AS number) is used as a tie-breaker if multiple best paths exist [28].

Note that we calculate not only the AS paths between two LN having IPv4 or IPv6 addresses but also between randomly selected Tor exit relays [69] and LN nodes with an IP address. To simulate the BGP routing of a node with an unknown address, we randomly assign an address to it according to the distribution of known addresses. In our simulation, the BGP paths can be asymmetric [68].

3.2. Topology Analysis

We analyze the AS topology of LN's P2P network and show its centralization. First, we show the address distribution of LN nodes collected in our snapshot in Figure 2. In this snapshot, only about 23% of LN nodes are announcing a public address (e.g., IPv4, IPv6). More than 70 % of the LN nodes have Tor enabled to hide their addresses or listen for other connections over Tor. We

2. The dataset of IXP links is released only in April 2022.

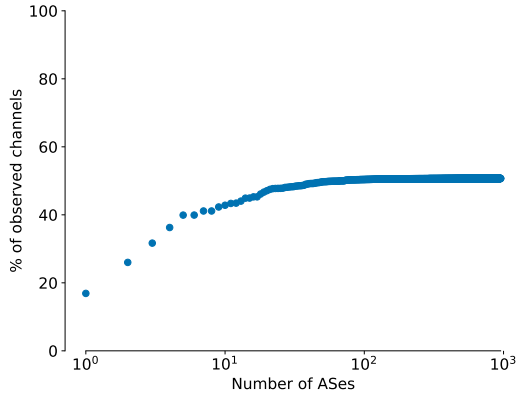

Figure 3. Cumulative percentage of observed channels. A few ASes have an extremely high centrality, e.g., the top 5 ASes see 40 % of all channels, equivalent to 80 % of channels that can be observed.

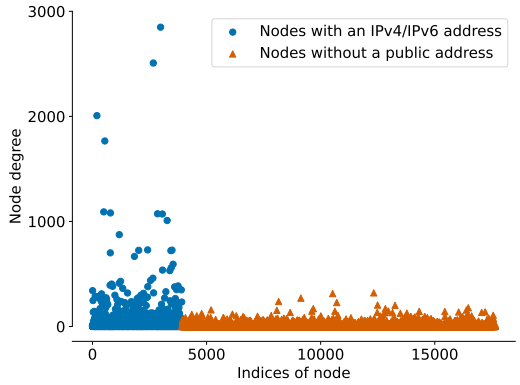

Figure 4. Nodes with high degrees typically use public addresses, e.g., 60 % of all the nodes with a degree of at least 100 advertise an IPv4 or IPv6 address.

also note roughly 13 % of LN nodes are unknown. These results are consistent with previous studies [58], [63], [79].

In Figure 3, we illustrate the portion of LN channels whose traffic can be observed by ASes on the BGP routes between channel endpoints. Note that if both endpoints of a channel use Tor, the traffic is not observable by ASes because it does not leave the Tor network. In total, 50.7 % (41,027 out of 80,932) of channels in our snapshot are observable. Figure 3 shows that a few ASes control a large number of observable channels. For instance, the top 5 central ASes see 40 % of all channels, accounting for 80 % of all channels that can be observed. Among them, AS14618 (Amazon) has the highest centrality for seeing 16.9 % channels, or 34 % of all observable channels, although it hosts only 2.8% of LN nodes in the network.

To understand why many channels are still observable when the majority of nodes are using onion services, we measure the degree of all nodes (i.e., the number of their channels) and plot them in Figure 4. Interestingly, most central LN nodes (i.e., with the highest degrees) do advertise IPv4 and/or IPv6 addresses. For example, 60 % of all the nodes with a degree of at least 100 advertise an IP address. All LN nodes with a degree of no less than 300 also have an IPv4 or IPv6 address. One possible explanation is that advertising an IPv4 or IPv6 address makes an LN node more accessible to others when performing payment routing. The node with a public address, thus, can gradually become a payment hub, effectively earning more routing fees. Inherently, this increases the channels controlled by the underlying ASes, as can be seen previously.

In summary, the AS topology of LN is heavily centralized with a large portion of channels controlled by a few ASes, hinting at a worrying threat from such AS-level adversaries.

4. The Revelio Attack

In this section, we present the overview of the Revelio attack. In §4.1, we introduce the threat model of a passive AS-level adversary, which contrasts models used in prior work that consider colluding central LN nodes. We then present the adversary's goals to learn the payment amount as well as to break sender/receiver anonymity and show the anonymity metric used for the evaluation. Finally, we describe two high-level steps of the Revelio attack in §4.2.

4.1. Threat Model

We consider a passive AS-level adversary in this work. The attacker's primary goal is to deanonymize LN payments by learning the payment amounts as well as the payments' senders and receivers. In the process of deanonymization, the attacker may also learn the timing of the payments and the addresses associated with the payers and payees. We further define the anonymity metric that will be used to evaluate the effectiveness of the attack.

Adversarial capabilities. We consider a network adversary who fully controls a single AS network, which we call a malicious AS. The adversary can inspect and record any packets traversing its network along with their other metadata, such as the timestamps when each packet is forwarded. Typical Internet Service Providers (ISPs) already have such capabilities [47], [51].

We assume the attacker may access the publicly-available data, such as the datasets for AS path estimation. Following this assumption, we allow the adversary to run an LN node and collect the updated topology information (e.g., node IDs, node addresses, channels) of the LN's P2P network. Similarly, the attacker is also aware of the algorithms for estimating AS paths between two ASes (c.f. Section 3) as well as the algorithms for calculating LN payment paths in open-source LN clients (e.g., lnd, eclair, CLN, electrum) [72].

The adversaries, however, remain passive and limited to AS-level, meaning they refrain from active actions (e.g., tampering or delaying packets, making or forwarding LN payments) and only see potentially encrypted traffic at the AS level. While passive adversaries have been considered in prior privacy attacks in LN [34], [53], [59], [66], [70], Revelio attack is the first to consider the privacy threat from AS-level adversaries. Unlike existing works, Revelio adversaries do *not* require controlling multiple LN nodes at central locations, which is non-trivial without payment hijacking attacks [72], to see a large portion of LN traffic. Instead, multiple ASes are already in such a position (c.f. Section 3). On the other hand, the Revelio adversaries do not see application-level information of LN payments like existing attacks, such as the payment amounts.

Adversarial goals. The first goal of the adversary is to break the off-path payment secrecy [34], i.e., parties not involved in the payment routing should not learn anything about the payment amount. Since the computation of the payment path between two LN nodes also depends on the payment amount, breaking payment secrecy helps to achieve the second goal: The Revelio adversary wants to break the on-path relationship anonymity [34], i.e., no one on the path should learn the path's endpoints. More generally, the Revelio adversary aims to reduce the set of possible senders and receivers, which we call the sender and receiver anonymity set, respectively. We formally define the metric related to these two goals below.

Anonymity metric. As a metric for payment secrecy, we compute how close the adversary's estimate is to the payment amount. For this, we consider the order of magnitudes of difference between the amount and the estimation.

The metric for the relationship anonymity is defined as follows: Let $\mathcal{A}$ be a malicious AS. Let $P_{s,t}$ be a multi-hop payment from node s to t. We define the path anonymity set of $P_{s,t}$ with respect to $\mathcal{A}$ as the set of node pairs $\mathcal{P}$, such that for every $(u, v) \in \mathcal{P}$, the adversary $\mathcal{A}$ can not distinguish the multi-hop payment $P_{s,t}$ from $P_{u,v}$ going from u to v. We further define the sender anonymity set $\mathcal{S} = \{u|\exists(u, v) \in \mathcal{P}\}$ and the receiver anonymity set $\mathcal{R} = \{v|\exists(u, v) \in \mathcal{P}\}$, both with respect to the multi-hop payment $P_{s,t}$ and the adversary $\mathcal{A}$. The larger these anonymity sets are, the better the relationship anonymity is for the sender, respectively receiver of the multi-hop payment.

In this work, we directly look at the size of the anonymity sets and refrain from using precision and recall (e.g., [34], [59]) because this metric only models anonymity as a binary decision, i.e., whether the guess is correct. We also choose not to use entropy (e.g., [66]) as an anonymity metric as its computation does not scale for all possible LN paths with different payment amounts.

4.2. Attack Overview

The Revelio attack consists of two high-level steps: a traffic monitoring phase (Step I) and a simulation-based deanonymization phase (Step II).

[Step I] Traffic monitoring (§5). In this step, the attacker monitors packets at the network-level and abstracts the observed traffic into application layer information. As visualized in Figure 5, this phase has three substeps. In Step I-①, the attacker identifies all LN channels having network traffic transited through the adversarial AS. In particular, the attacker maps the IP address of the observed packets' source and destination to the node IDs of the channel endpoints. This knowledge allows the attacker to map a payment observed on the wire to the involved LN parties in Step I-②. As shown in Figure 5, multiple payments may cross the adversarial AS. The attacker then groups them into multi-hop payments based on the recorded timestamps of their associated packets in Step I-③. The result of Step I is a partial path containing all LN nodes involved in a multi-hop payment.

[Step II] Deanonymizing senders and receivers (§6). In this deanonymization step, the adversary aims to deduce the sender and receiver from the partial path of node IDs

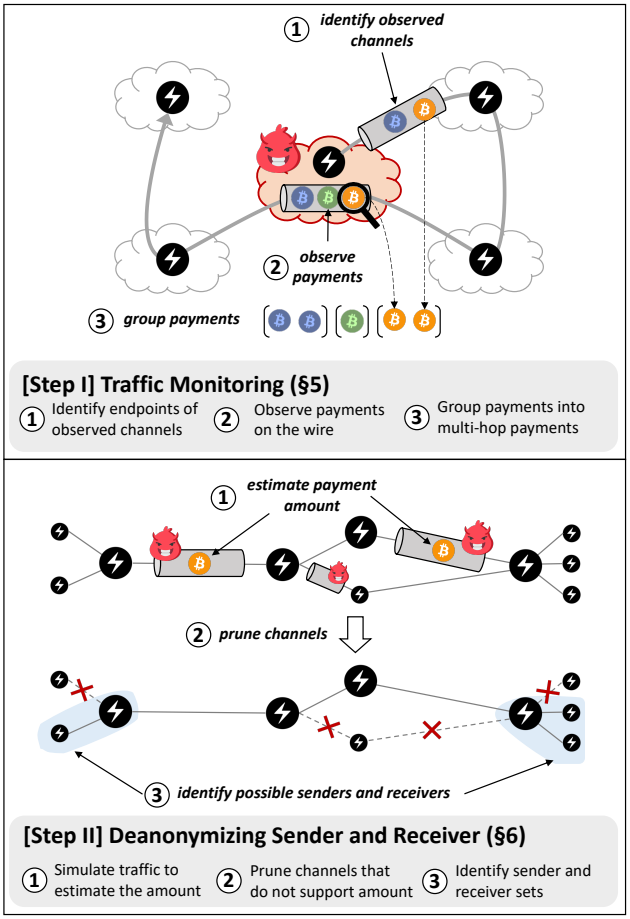

Figure 5. The two main steps of the Revelio attack. In the traffic monitoring step, a malicious AS abstracts a partial path of a multi-hop payment from observed traffic. In the deanonymization step, the adversary identifies possible senders and receivers of the observed payment by running a simulation-based anonymity set reduction.

obtained in Step I. Unlike Step I, this deanonymization step can be done offline as it does not require any interaction with live networks but only some calculations with the recorded partial path. Particularly, in Step II-①, the adversary simulates payment paths and estimates the possible range of the amount of the targeted payment. Then, in Step II-②, the attacker prunes the LN topology by removing all channels which either do not support the estimated amount or do not carry any traffic. The adversary finally identifies the set of possible senders and receivers in Step II-③. Figure 5 shows an example where the attacker identifies the sender (respectively the receiver) is in an anonymity set of two nodes (respectively three nodes).

5. Traffic Monitoring

In the traffic monitoring step, the adversary translates network-level traffic to corresponding LN traffic as the inputs for the deanonymization in the next step. Particularly, the attacker composes a list of observed channels and finds which channels are part of an attacked multi-hop payment. For this, the adversary observes channel opening and matches the IP address of the endpoints to

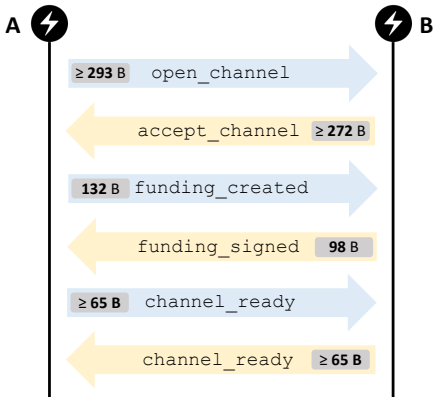

Figure 6. Transcript of messages for a channel opening from node A to node B [41]. Message size is shown in Byte (B).

LN node IDs (§5.1). The adversary then identifies ongoing payments (§5.2) and lastly, groups them into a single multi-hop payment (§5.3).

5.1. Identifying Observed Channels

The main goal of an adversary here is to identify all LN channels that she is on at least one BGP path between two endpoints.[3] Knowing the *observable channels* is critical for mapping network-level traffic to LN node IDs and for the deanonymization strategy presented later in §6. Finding observable LN channels requires running an LN node to learn the updated LN topology, recording LN traffic on the wire, and identifying the channels (e.g., their endpoints). We explain these three steps in more depth as follows.

Learning the updated topology. By running a node and listening to gossiping messages in the LN's P2P network, the adversary learns the entire LN topology and gets informed whenever other nodes change their addresses or their channels are updated. The attacker saves the list of channels and the list of nodes, including their announced addresses (e.g., IPv4, IPv6, .onion addresses), into her database.

Recording LN traffic. An adversarial AS can easily filter LN traffic from all traffic going through her network. Particularly, the adversary can record any TCP packets originating from or destined to port 9735, which is the default port for LN communication [40]. We note that the attacker cannot observe the communication between two nodes using Tor as they connect using Tor hidden services, rendering their traffic encrypted and never leaving the Tor network. It also means any channels with at least one endpoint that is not using Tor are observable by Revelio adversaries.

In LN specifications [40], the communication between two LN nodes involves multiple messages, each encapsulated in a single TCP packet. Despite the messages being encrypted under the Noise Protocol Framework [43], they do not apply any random padding. The adversary, thus, can use the size of the TCP payload to deduce the LN message type. Indeed, the payload size is constant for different messages of the same type, even though the packets we capture in our experiments are slightly bigger than in the protocol description [42]. We also note that some messages may not have a fixed length, yet, their order in a group of messages and their direction fully characterize the message types. For example, when the adversary observes two nodes exchange a 132B-packet followed by a 98B-packet, the adversary can deduce that they are opening a channel and then confirm the type of messages observed before and after these two packets, see Figure 6. When the adversary observes only one direction of communication, the same strategy can still be applied. In short, the adversary AS can easily learn the type of LN messages sent across the wire.

Identifying channel endpoints. The attacker identifies observed channels by matching the nodes' announced addresses with the source and destination addresses of the recorded packets. If both endpoints of a channel match with some announced clearnet (i.e., IPv4 or IPv6) addresses, the adversary can easily identify their corresponding node IDs. However, the addresses observed by the adversary may not correspond to any announced address because a node using Tor may establish a channel to a clearnet node while hiding its address, or unknown nodes may not advertise their addresses. There are thus four possible scenarios for the observed channel's endpoints when that happens:

(1) *clearnet – .onion:* The observed channel is between a node with a clearnet address and a node with a .onion address if the unmatched address is one of the Tor exit relays. The adversary can check available channels of the clearnet node to identify the .onion address and ID of the other endpoint. Since a clearnet node may have multiple channels with nodes over Tor, the adversary should track the channel since its opening by looking for the channel opening messages on the wire (see Figure 6) and a `channel_announcement` message indicating a new available channel, which has the clearnet address and a Tor address as endpoints, is circulated via the gossiping protocol a few minutes later. The adversary then binds the unmatched address (i.e., a Tor exit relay) to the identified node for this channel. Our experiments assume that LN nodes behind Tor use the same circuit (i.e., the exit relay is unchanged) for the same connection because they want to maximize the channel's uptime for better revenue. In practice, when a different exit relay is used, the adversary can still observe a disconnection and then the connection from a new exit relay to the known clearnet address.

(2) *clearnet – unknown:* Using the same strategy presented in scenario (1), the adversary can bind the unmatched address to a node having an unknown address. The only difference is that the observed address is not a Tor exit relay. We also assume these unknown nodes do not change their addresses.

(3) *unknown – .onion:* The endpoints of the observed channel here are similar to scenario (1), except that the observed clearnet address does not match any announced address from LN nodes. Similarly, the adversary can track the channel's opening messages on the wire and look for a new available channel with an unknown node and a Tor node as endpoints. When more than one new channel satisfies such a condition, the adversary can leverage the timings of channels' funding transactions on the Bitcoin

3. The attacker may not see both traffic directions because of asymmetrical BGP routing [68].

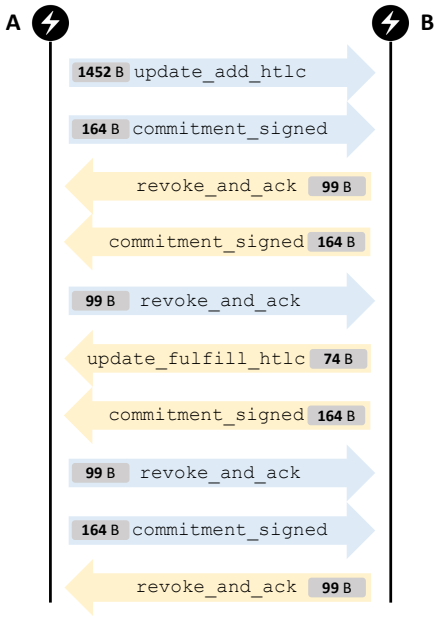

Figure 7. Transcript of messages for a payment from node A to node B [41]. Message size is shown in Byte (B).

blockchain, which are included in the channel announcement, to tiebreak them. In particular, the funding transaction of a new channel opening is broadcasted to the Bitcoin P2P network immediately after the receipt of the `funding_signed` message. Its timestamp can be easily retrieved from a blockchain explorer or a Bitcoin client.[4] The adversary thus can link the observed channel opening to the closest funding transaction and its corresponding channel, effectively binding the LN node with the unknown address to the observed clearnet address.

(4) *unknown – unknown:* Using the same strategy as in scenario (3), the adversary can find out the endpoints of an observed channel with addresses that do not match any announced addresses or Tor exit relays. Here, the adversary cannot distinguish these two endpoints to bind the observed addresses to them until another channel using the same address is observed and cross-checked.

To summarize, the attacker can identify the node IDs of the endpoints for all observed channels, either directly or over the additional attack step. As a result, the attacker has a list of observed channels and a mapping from IP addresses to LN node IDs.

5.2. Observing Payments

The adversary also detects LN payments on the observed channels based on the network-level traffic. We show the transcript of multiple messages involved in an LN payment in Figure 7. Since the lengths of those messages also characterize their type, the adversary can use the same techniques as presented in §5.1, i.e., relying on the TCP payload length to identify the messages. For example, the adversary can identify a new payment made on an observed channel by looking for the `update_add_htlc` message with a payload size of 1452 Bytes. Again, the attack remains feasible even when the adversary can observe only one direction of the communication. For instance, when the adversary sees a `revoke_and_ack` message followed by a `commitment_signed` message in the same direction, she can deduce these are from a payee to a payer without seeing traffic in the opposite direction.

When the attacker detects a payment on an observed channel, she records the timestamps of the associated packets and uses the mapping generated in the previous step to translate the channel to the node IDs of its endpoints. As a result of this step, the attacker produces a list of tuples where each detected payment is represented by a tuple (t, u, v) consisting of the recorded timestamp t of the first observed message (e.g., `update_add_htlc`) as well as of the sender u and the receiver v.

5.3. Grouping Payments into Multi-Hop Payments

Next, the adversary groups the previously observed payments into separate multi-hop payments. The main insight here is that two payments may belong to the same multi-hop payment if the recorded timestamps of their packets are close to each other. In a multi-hop payment, the difference in the packet timestamps recorded at different hops depends on the LN topology (e.g., possible numbers of nodes between them) and the delays of the underlying Internet infrastructure. Note that LN clients do not introduce any additional delay, i.e., they immediately send messages to the next-hop node on the payment path. To estimate the latency between two LN nodes, the Revelio attacker uses the methodology proposed by Rohrer *et al.*, e.g., by sending ICMP ping messages to them [59]. Then, for any two non-neighboring nodes u and v, the adversary maintains an estimated range of the propagation time $\Delta_{u,v} = [t_{lower}, t_{upper}]$. This allows the adversary determine whether two payments with (t_0, u_0, v_0) respectively (t_1, u_1, v_1), where $t_1 > t_0$, belong to the same multi-hop payment. To this end, the adversary computes the time difference $\delta = t_1 - t_0$ and groups two payments if the difference is within the estimated range Δ_{u_0,v_1}, i.e., $t_{lower} \leq \delta \leq t_{upper}$.

Using the same strategy, the adversary iterates through all recorded payments (t, u, v) and groups them into multi-hop payments. Each group of messages reflects one single multi-hop payment at different channels the adversary observes. Next, for every group of messages $\{(t_0, u_0, v_0), (t_1, u_1, v_1), \cdots, (t_k, u_k, v_k)\}$, where $t_0 \leq t_1 \leq \cdots \leq t_k$, the adversary produces an ordered list of involved LN nodes $[u_0, v_0, u_1, v_1, \cdots, u_k, v_k]$. After removing all duplicated nodes, the result is a list of nodes that appear in the multi-hop payment, and their orders are the same as in the payment path. As not all channels are observed by the attackers, there might be some nodes before and after u_0 and v_k, respectively, or in between pairs of (u, v). We call this list a partial payment path and use it as the input for the deanonymization in the next step.

4. The broadcasting timestamp of a transaction should not be confused with the timestamp of the block including it. New transactions are usually propagated to all Bitcoin nodes in a few seconds; hence the timestamps recorded by different clients can be slightly different.

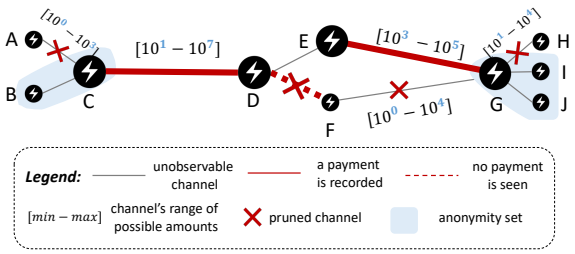

Figure 8. A payment from $B \to C \to D \to E \to G \to I$ can be deanonymized by an attacker observing the channels $C \leftrightarrow D$, $D \leftrightarrow F$, and $E \leftrightarrow G$.

6. Deanonymizing Sender and Receiver

The attacker builds on the traffic monitoring to deanonymize the sender and receiver of LN multi-hop payments. In contrast to the previous step, which requires interactions with live networks, the deanonymization process only consists of the calculations of BGP and LN payment paths using publicly-available datasets and algorithms. Therefore, this step can be done offline and separately from the traffic monitoring step. Particularly, the attacker uses the computed partial payment path as the reference to estimate the range of possible payment amounts. The adversary then prunes the LN topology by removing channels that do not match the observed traffic or the estimated amounts. Finally, the adversary deanonymizes the sender and receiver by minimizing their anonymity sets.

An example. Throughout this section, we present the intuition of the deanonymization strategy using an example of a multi-hop payment made through $B \to C \to D \to E \to G \to I$, see Figure 8. In this example, the adversary monitors the channels $C \leftrightarrow D$, $D \leftrightarrow F$, and $E \leftrightarrow G$. From the traffic monitoring step, the adversary detects payments on channel $C \leftrightarrow D$ and channel $E \leftrightarrow G$ and then constructs $\{C, D, E, G\}$ as the observed partial path of the targeted payment. The attacker also notes there is no payment seen on channel $D \leftrightarrow F$. Moreover, the adversary is aware of the publicly available parameters of all channels, such as their lower (i.e., `htlc_minimum_msat`) and upper bounds (i.e., capacity or `htlc_maximum_msat`) on the payment amount.

6.1. Estimating the Payment Amount

The adversary aims to estimate the amount of a targeted payment (amt) based on its partial path. By default, the payment amount can be any value between 0 and $2^{24}-1$, which is the limit for channel capacity in LN [41]. Hence, the strategy here is to narrow down a range of possible amounts $R = [r_{lower}, r_{upper}]$ from $[0, 2^{24} - 1]$.

First, the adversary leverages the parameters of all channels that appear in the partial path. Particularly, when LN nodes establish a channel, they determine not only the channel capacity, which is the sum of their initial funds, but also the `htlc_minimum_msat` and (optionally) `htlc_maximum_msat` values that respectively impose the lower bound and upper bound for the payment values they are willing to forward in the channel. These channels' parameters are part of gossiping messages; thus, the adversary can learn them when taking a snapshot of the entire LN topology. Because all channels in the partial path can forward the targeted payment, the adversary can deduce that amt must be no less than these channels' capacities. In other words, the attacker sets r_{upper} equal to the smallest channel capacity in the partial path. Similarly, because amt must be larger than the maximum `htlc_minimum_msat` and smaller than the minimum `htlc_maximum_msat` of all involved channels, the adversary can set r_{lower} and r_{upper} accordingly. The example in Figure 8 shows that R can be narrowed down into $[10^3, 10^5]$ thanks to the publicly available parameters of channels $C \leftrightarrow D$ and $E \leftrightarrow G$.

Algorithm 1 Partitioning a given range of amounts into intervals in which corresponding paths are the same.

Require: S, T: the partial path's first and last nodes. r_{lower}, r_{upper}: the lower and upper bounds of possible amounts for payments from S to T.

Ensure: $\mathcal{L} = [(\mathcal{P}_1, l_1, u_1), \cdots, (\mathcal{P}_k, l_k, u_k)]$: A list $\mathcal{L}$ of k tuples, each shows that when $l_i \leq amt \leq u_i$, the path from S to T is $\mathcal{P}_i$.

procedure PARTITIONAMOUNTS(r_{lower}, r_{upper})
  $\mathcal{P}_{lower} \leftarrow$ calculatePath(S, T, r_{lower})
  $\mathcal{P}_{upper} \leftarrow$ calculatePath(S, T, r_{upper})
  if $\mathcal{P}_{lower} = \mathcal{P}_{upper}$ **then** ▷ Paths are the same.
    return $[(\mathcal{P}_{lower}, r_{lower}, r_{upper})]$
  $r_{mid} \leftarrow \lfloor (r_{lower} + r_{upper})/2 \rfloor$ ▷ Binary search.
  $\mathcal{L}_{lower} \leftarrow$ PARTITIONAMOUNTS(r_{lower}, r_{mid})
  $\mathcal{L}_{upper} \leftarrow$ PARTITIONAMOUNTS($r_{mid}+1, r_{upper}$)
  $\mathcal{L} \leftarrow \mathcal{L}_{lower} + \mathcal{L}_{upper}$
  $\mathcal{L} \leftarrow$ mergeRangeOfTheSamePath($\mathcal{L}$)
  return $\mathcal{L}$

Second, the attacker computes the payment paths from the first node to the last node in the partial path with varying payment amounts. The adversary then narrows the range of possible amounts R to the ones with corresponding paths that match the observed partial path. This strategy comes from an observation that if two payment paths between the same endpoints differ, their amounts must also differ because there cannot be more than one path for the same amount.[5] A naive attacker would compute the paths for *every* payment amounts possible, which is not feasible within a reasonable computation time when r_{upper} can be as high as $2^{24}-1$. Instead, Revelio adversary uses a binary search to efficiently calculate the paths with an assumption that if the payment paths between two nodes are the same for two amounts, then they are also the same for all amounts between them. In theory, this might not be the case as a path could be the cheapest path for only one specific amount, but unusable for all other amounts in the same range, e.g., by containing a channel that supports only this specific amount. As we will show in §7.2, this rarely occurs, and we thus ignore it.

We present the pseudo-code of this binary search in Algorithm 1. Particularly, the adversary partitions the range of amounts, starting with the $[r_{lower}, r_{upper}]$ from the previous step (e.g., $[10^3, 10^5]$), into smaller intervals of amounts so that the computed paths with amounts

5. On the other hand, the cheapest paths for different amounts can be the same.

from the same interval are the same. For each loop, the attacker computes the paths for the lowest and the highest amount in the given range. The computation here follows publicly-available path-finding algorithms in open-source LN clients. The specific algorithm can be chosen accordingly to the client that the nodes are running [59]. If the paths are the same, all paths are the same for all other amounts in that range, which is marked as an interval (c.f. line 5). Otherwise, the adversary recursively applies the binary search for the lower and upper halves of the current range until all intervals of amounts are found. The attacker finally merges the intervals of the same path $\mathcal{P}_i$ to form its continuous range of $[l_i, u_i]$ (c.f. line 10). The final output of this amount estimation step is a list $\mathcal{L}$ of intervals of possible amounts and their corresponding paths. For instance, by running Algorithm 1, the adversary may learn that if the amount of the targeted payment is within $[10^3, 10^4]$, the path would be $\{C, D, F, G\}$ (e.g., because channel $F \leftrightarrow G$ imposes less fee than channel $E \leftrightarrow G$) and when it is within $[10^4 + 1, 10^5]$, the path would be $\{C, D, E, G\}$, see Figure 8.

6.2. Pruning Channels

With the possible paths and their corresponding ranges of possible amounts, the adversary can prune all channels in the LN topology that do not carry payment traffic or do not support an amount within these ranges. Specifically, the adversary compares the computed paths in each interval of amounts with the given partial path and removes them if nodes in the partial path do not appear in the same order in the computed paths. For example, in Figure 8, the path cannot be $\{C, D, F, G\}$ because channel $D \leftrightarrow F$ is observable, yet no correlating payments are seen. The amount of the targeted payment thus cannot be within $[10^3, 10^4]$. In the same example, with the possible amounts now narrowed down into $[10^4 + 1, 10^5]$, the adversary can prune channels with a capacity lower than $10^4 + 1$ or `htlc_minimum_msat` parameter higher than 10^5, such as channels $A \leftrightarrow C$ and $G \leftrightarrow H$. The result of this pruning step is a simplified LN topology that includes only possible channels for the targeted payment.

6.3. Identifying Sender and Receiver Anonymity Sets

The attacker identifies the sender and receiver anonymity sets of the targeted payment based on the pruned LN topology. At a high level, the adversary splits the nodes in the LN graph into two components such that they contain the first (e.g., node C in Figure 8), respectively the last node (e.g., node G) of the partial path. The two components are only connected via a payment path that matches the observed partial path. More specifically, the adversary first creates an initial anonymity set for the sender and receiver, which contains all LN nodes. The adversary then removes nodes that cannot reach to the first node, respectively from the last node, on the partial path because of pruned channels. Following this, the adversary removes nodes from their anonymity sets if their corresponding paths do not match the partial path. The intuition here is that the sender, respectively receiver, of the payment must be in the same connected component as the first, respectively the last hop of the observed partial path. In Figure 8, the final sender and receiver anonymity sets are $\{B, C\}$ and $\{G, I, J\}$, respectively.

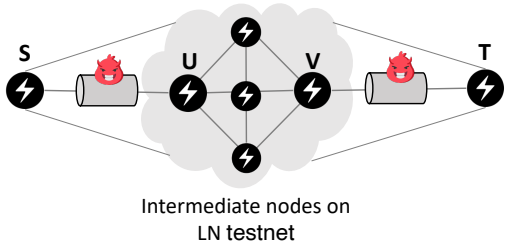

Figure 9. Topology setup for payment grouping experiment. Our nodes S and T connect to five other nodes, which also connect to form a near full mesh, on LN testnet. The attacker aims to group payments observed at channels $S \leftrightarrow U$ and $V \leftrightarrow T$ into multi-hop payments.

7. Evaluation

This section evaluates the two phases of the Revelio attack. First, we show that traffic monitoring, e.g., detecting and grouping payments into multi-hop payments, is indeed feasible in live network settings (§7.1). Second, we evaluate the effectiveness of the deanonymization step via several thousands of simulations (§7.2), showing that the attack is successful in up to $32.34\,\%$ of cases.

7.1. Effectiveness of the Traffic Monitoring

This subsection shows that monitoring LN traffic to identify multi-hop payments is practical. We present the experiment setup and show that traffic monitoring for payment grouping succeeds in realistic scenarios.

Experiment setup. We perform controlled traffic monitoring experiments in a small network of LN nodes. Note that evaluating this attack step at the scale of the entire LN network is challenging without controlling an AS because traffic monitoring requires capturing packets on the wire. We depict the topology of this network in Figure 9. In particular, we control two LN nodes S and T, running lnd version 0.14.0 on two separate virtual machines (VMs). We establish channels from these two nodes to a cluster of five intermediate nodes, each advertising an IPv4 address. The five intermediate nodes are connected in a full mesh except for one missing channel.[6] Moreover, these nodes are located in four different regions, i.e., Europe, Oceania, North America, and South America. With the widespread locations of the intermediate nodes, we can thus capture the realistic delays of LN payments in practice. We run this experiment on LN testnet, a testing network that mimics the main network (mainnet) and dedicates to testing purposes.

To implement a Revelio adversary in our experiment, we first select two specific nodes from the five nodes on LN testnet, called U and V. We then capture the traffic between our nodes and the five nodes by running pcap on the machine hosting the two VMs and filter the traffic between S and U, respectively, between V and T. We

6. As we do not control the intermediate nodes, we cannot build a full mesh.

implement an attack script that uses these traffic traces as inputs, emulating an adversary observing channels $S \leftrightarrow U$ and $V \leftrightarrow T$. In the topology we consider, there are 18 unique paths for multi-hop payments from S to T such that each path includes at least one channel observed by the adversary — six paths use only $S \leftrightarrow U$, another six paths use only $V \leftrightarrow T$, and six paths include both channels.

In each experiment, we send 200 multi-hop payments from S to T over a path chosen independently and uniformly at random from the set of the aforementioned 18 paths. We leverage the built-in commands of lnd client (e.g., buildroute, sendtoroute) to construct the chosen payment path and send the payment along this path. The main goal of the adversary here is to group the traffic traces collected at channels $S \leftrightarrow U$ and $V \leftrightarrow T$ into individual multi-hop payments. When the adversary correctly identifies that either only channel $S \leftrightarrow U$, only channel $V \leftrightarrow T$, or both of the channels appear on a payment path, we consider it as a correct grouping. We let the overall success rate of channel grouping be the number of correct guesses divided by the number of payments.

In all experiments, we consider a range of delay between S and T is $\Delta_{S,T} = [0.8\,\mathrm{s}, 2.2\,\mathrm{s}]$, which is measured using the methodology proposed by Rohrer *et al.* [59]. We also send transactions with a fixed rate of 0.1 tx/s, causing the adversary to observe, on average, one transaction every 15 seconds (or 0.066 tx/s) on each of channels $S \leftrightarrow U$ and $V \leftrightarrow T$. We chose the transaction rate of 0.066 tx/s per channel because the transaction rate in today's LN network is likely to be much lower. Although there is no prior study on the average transaction rate of the entire LN network, one can estimate it from publicly available data reported by large entities. For example, LNBIG reports an extraordinarily high volume of LN payments with 11,510 transactions recorded on April 26, 2022 [45]. Since LNBIG controls about 7000 channels, this converts into a per-channel transaction rate of 0.000019 tx/s (or each channel sees one transaction every 14 hours). Even when we do not consider a uniform transaction rate throughout the day and channels but a bursty one, e.g., all transactions happen within one hour and within 10% of all channels, the transaction rate would only be 0.004 tx/s per channel (or each channel sees one transaction every 4 minutes).

To evaluate the effectiveness of traffic monitoring, we focus on two aspects. First, we test if channels and payments can be detected from packet traces. Second, we compare the effectiveness of payment grouping in varying settings as follows.

- *Attack strategies:* Recall that the adversary must identify what channel(s), i.e., only $S \leftrightarrow U$, only $V \leftrightarrow T$, or both, are included on a targeted payment path. With a *random guessing* strategy, one can already achieve an expected success rate of $\frac{1}{3}$. Hence, our grouping strategy using estimated delay must have a higher success rate than $\frac{1}{3}$ to be considered effective.
- *Observed communication direction(s):* We also test different attack capabilities in terms of the observed communication directions. Particularly, for each of the two channels $S \leftrightarrow U$ and $V \leftrightarrow T$, we create two scenarios, i.e., the adversary sees only direction from payer to payee or vice versa.[7] In total, we have four different scenarios of attack capabilities.
- *Transaction rate:* One of the main factors for the effectiveness of the payment grouping is the transaction rate per channel, that is, if there are too many payments on the same channel in a short amount of time, it is hard for an adversary to separate them since the encrypted traffic is indistinguishable. We, therefore, test the success rate of channel grouping for varying per-channel transaction rates.

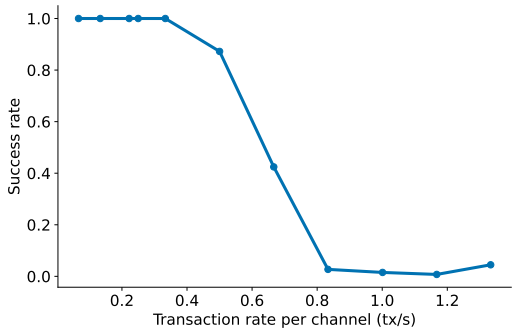

Figure 10. The success rate of payment grouping with varying per-channel transaction rates. The attacker can always group payments correctly with payment rates of below $0.33\,\mathrm{tx/s}$ per channel.

Results. We found that it is trivial to identify channels and detect payments from packet traces. Particularly, the source IP addresses encoded in the packets sent by the five nodes on LN testnet do match with their gossiped addresses in the LN's P2P network. Also, LN traffic can be easily filtered by the port number (i.e., 9735) included in the packets' header and the size of the packets' payload clearly reveals the type of the carrying LN messages as well as the payment direction (c.f. Figure 7).

With the per-channel payment rate of 0.066 tx/s, our Revelio adversary can correctly group payments in *all* cases, that is, achieving a success rate of 100%. Compared to an attacker with the random guessing strategy (i.e., with a success rate of $\frac{1}{3}$), our attacker demonstrates a clear advantage. This result holds true even when the adversary sees only one direction of the communication on the observed channels.

We present the success rate of channel grouping with varying transaction rates per channel in Figure 10. It shows that the adversary can perfectly group payments (i.e., maintaining the success rate of 100%) with a per-channel transaction rate of up to 0.33 tx/s. The success rate drops to 42% when the transaction rate approaches 0.66 tx/s (or two transactions every 3 seconds). We also notice the attack becomes ineffective as it performs worse than random guessing when the transaction rate is higher than 0.83 tx/s per channel. While this transaction rate may look small at first, it is in fact, 4-orders of magnitude larger than the estimated average transaction rate in current LN (i.e., 0.000019 tx/s per channel). More concretely, assuming the

7. We skip the scenario where the adversary sees both directions because only messages from payer to payee are used for easier detection.

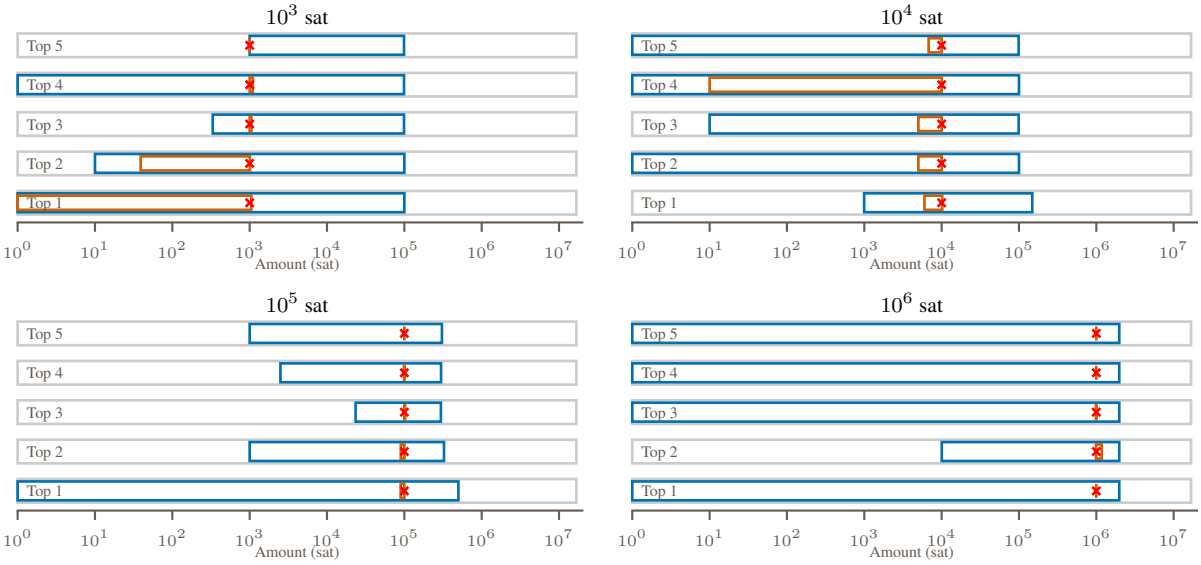

Figure 11. Attackers with a broader view can estimate payment values more accurately. The plot shows, for different payment amounts (marked with ×), the average estimated lower bound and upper bound (blue boxes), and the best estimated range with the least discrepancies compared to the actual amount (orange boxes).

transaction rate in LN doubles annually [5], it will take about *15 years* to render Revelio attack ineffective.

7.2. Effectiveness of the Deanonymization Attack

Here, we evaluate the effectiveness of the deanonymization step. We first introduce the experiment setup, focusing on our simulation framework. Then, we present the effectiveness of the payment amount estimation as well as how effectively the attacker can reduce the anonymity sets of the sender and receiver.

Experiment setup. We evaluate the deanonymization attacks at the scale of the entire LN topology because they do not require interaction with live networks as in the traffic monitoring experiments but only some LN payment path calculation and BGP simulation.

In our evaluation, we simulate thousands of multi-hop payments on the actual LN topology. First, we consider four payment amounts that are also considered in prior works (e.g., [59], [79]), i.e., $10^3, 10^4, 10^5, 10^6$ sat. By the time of writing, these values correspond to roughly 0.2 to 200$, respectively. A payment simulation chooses a random amount among these values and a random pair of LN nodes as the sender and receiver. We ensure a uniform distribution of the number of senders and receivers here, i.e., each node is equally likely to send and receive payment. To calculate the multi-hop payment path from the sender to the receiver in each simulation, we adapt the Python codebase of Electrum [76] wallet, which closely reflects the algorithms of lnd clients. This path-finder takes the previously collected snapshot of the LN topology as input and returns the payment paths between any two nodes. Note that we choose the path-finding algorithm of lnd in our experiments simply because the vast majority of the clients (> 91%) are running it [49].

We evaluate the attack effectiveness in five different scenarios of the adversary. In particular, we consider "Top n" adversaries, where n ASes with the broadest view, i.e., the ASes which can see the most channels (c.f. Section 3), collude. We vary the number of colluding ASes from one to five (i.e., Top 1 to Top 5) since the cumulative percentage of observed channels stagnates after five colluding ASes. In each attack scenario, we simulate 1008 payments. For every payment, we calculate the LN payment path, compute the BGP paths between any two consecutive nodes (c.f. Section 3), and mark the corresponding channels as observed by the adversaries if the BGP paths include a malicious AS. We assume that the channel grouping in the previous step is successful, meaning that the adversaries can perfectly construct a partial path consisting of the channels they observe for a targeted multi-hop payment.

We run our deanonymization attacks on a server with 48 cores of AMD EPYC™ 7742 CPU running at 2.25 GHz, 128 GB of memory, and the Ubuntu 18.04.6 LTS operating system. A single payment deanonymization in one attack scenario, including the LN payment path computation and BGP simulation, is complete within 30 seconds on average.

Results. In Figure 11, we show the estimated range of payment amounts in different attack scenarios versus the actual payment amounts. For each adversary (e.g., Top 5 attacker) and a payment amount (e.g., 10^3 sat), we show the original range (i.e., $[0, 2^{24} - 1]$), the average estimated lower bound and upper bound, and the best estimated range with the least discrepancies compared to the actual amount. Figure 11 shows that the Top 5 attacker who has the most comprehensive view of the LN topology can limit the amount to a range of fewer than 5 magnitudes in the median. For amounts of 10^5 sat and 10^6 sat, the attacker often guesses exactly the magnitude of the payment amount. This result demonstrates a worrying threat as payments of high amounts are considerably more sensitive than payments of smaller amounts. Figure 11

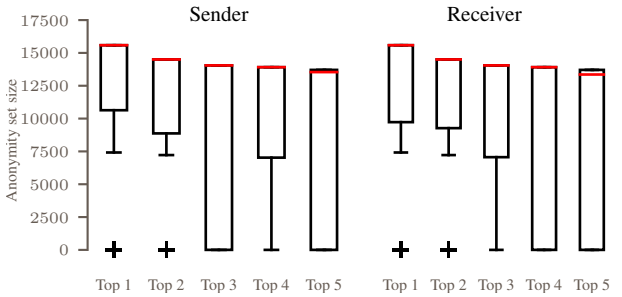

Figure 12. While the anonymity sets are large, the attacker achieves better reduction with an increasing view.

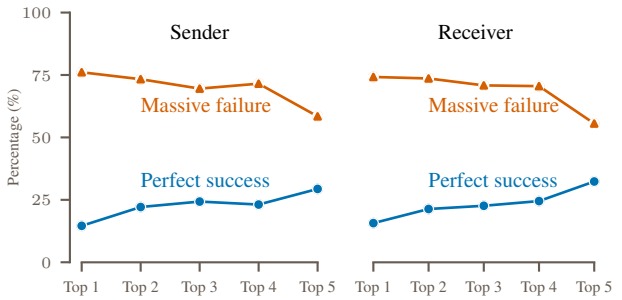

Figure 13. The attack is a serious threat, e.g., the Top 5 ASes reduce the anonymity sets to one (*perfect success*) for more than 30 % of observed payments. *Massive failure* denotes anonymity set sizes above 10,000.

further suggests that an attacker such as Top 1 that sees a comparably smaller fraction of channels naturally fails to guess accurate lower bounds more frequently. As we can see in the Top 2 and 3 scenarios, an increasing number of observed channels increases the effectiveness of the attack. More generally, we observe that the magnitude of the estimated upper bounds is closer to the payment amount compared to the lower bounds. Our attack also almost certainly includes the payment amount in the estimated range. Particularly, the actual amount is not included in the estimated range in only less than 0.2 % of the tested scenarios.

We now focus on the attack effectiveness in deanonymizing payments, as shown in Figure 12. The distribution of the sizes of the anonymity sets suggests that the attacker can only hardly deanonymize the sender or the receiver: The median size of the anonymity sets is higher than 10,000 in all scenarios. However, when looking at Figure 13, we see that appearances are deceptive: The attack either perfectly succeeds (i.e., reducing the anonymity set size to 1) or massively fails (i.e., the anonymity set size remains higher than 10,000) to deanonymize senders and receivers. Even in the Top 1 scenario, the attacker achieves in almost 15 %, respectively 16 % to deanonymize the sender, respectively, the receiver. Similar results hold for the attack in the Top 2 and the Top 3 scenarios where the success rates increase up to 24 % and 22 %, respectively. If even more ASes collude (i.e., the Top 5), then up to 32 % of all receivers can be perfectly deanonymized.

These results show that the Revelio attack not only estimates the payment amount effectively but also deanonymizes a significant fraction of senders and receivers of payments.

8. Discussion

In this section, we discuss several key points. We first explain why the limitations of the Revelio attack do not hinder its practicality (§8.1). We then highlight the inherent weaknesses of LN that have been the root causes for not only Revelio but also for many existing privacy attacks against it (§8.2). Some of these weaknesses exist in many other overlay networks, suggesting the possibility of generalizing Revelio into a more prominent family of privacy attacks. Based on the discussed root causes, we further present both readily deployable defenses and longer-term countermeasures to the Revelio attack (§8.3) Finally, we present the ethical considerations in this paper (§8.4).

8.1. Revelio's Limitations

We discuss two limitations of the proposed Revelio attack, that is, traffic to Tor's hidden services is unobservable, and the exact payment amount is unknown. We discuss how these limitations can be addressed and why our attack is still a serious threat for privacy in LN.

Observing Tor's hidden services. The proposed Revelio attack does not apply to channels with both endpoints using Tor because the communication from one node to the other's hidden service is additionally encrypted. To observe these channels, adversaries with extended capabilities can launch additional attacks against Tor hidden services, such as running malicious entry guards [13], [37], [44], [55], disrupting guard nodes selectively [32], exploiting side-channels [50], and fingerprinting traffic [30], [56]. When that happens, the adversaries will have a much more comprehensive view of the LN topologies, and the attack effectiveness will thus be drastically increased. Therefore, we consider the current results as a lower bound with respect to this limitation and leave the adaptation of additional attacks for future works.

Learning the exact payment amount. Our evaluation in Section 7.2 shows that the Revelio adversary usually ends up with a notable range of possible values, which inherently leads to insignificant anonymity set reduction. Here, we note a few practical ways to learn the exact payment amount, although the attack requires extra steps and may not remain passive. For example, the adversary can specifically target popular services, such as podcasts, streaming, or social apps, that publicly announce their payment amounts. Moreover, Revelio adversaries can launch channel probing attacks [12], [20], [71], [81] to monitor the balance of all channels. Yet, such an attack requires having LN nodes sending actual payments, and it may cost several thousand dollars for every probing. Alternatively, the adversary can run an intersection attack (e.g., [10], [22]) when observing the same payment multiple times. Lastly, the adversarial AS can further tighten the range of possible amounts by looking for payment failures that happen due to insufficient balance and cause a different routing path (i.e., with extra boundary conditions) to be chosen.[8] Indeed, if the same node issues another payment within a short time, the adversary can deduce that the same amount is routed twice.

8. In case of payment failure, `update_fail_htlc` messages ($\geq 44B$) are sent instead of `update_fulfill_htlc` messages.

8.2. LN's Inherent Weaknesses

LN is designed to provide better payment privacy than Bitcoin. Unfortunately, it still has several inherent weaknesses that render many privacy attacks, such as Revelio, possible. While some of the weaknesses pertain to the LN protocol and can be hardened with short-term defenses or removed with long-term countermeasures, some others pertain to the inherent vulnerability of the Internet infrastructure and are harder to fix.

LN protocol. Since LN is permissionless, an attacker can easily create many nodes in the LN topology and observe a large portion of multi-hop payments forwarded by them. This weakness has been the root causes of several existing attacks that deanonymize payment endpoints [59], [66], or channel balances [12], [20], [71], [81].

With the cheapest-path routing mechanism, the LN network becomes centralized with a small number of high-degree nodes [77] and, thus, is prone to structural attacks, such as route hijacking (and payment deanonymization as a follow-up attack) [72], or Revelio attacks in which malicious ASes see much of LN traffic.

The open-source routing algorithms also let Revelio attackers easily simulate payments. The LN's topology, including channels' parameters, is publicly available to any node joining the network, thus enabling the Revelio adversary to confirm observable channels as well as deanonymize payment amounts (c.f. Section 6).

End-to-end communication. Revelio adversaries can easily filter out LN traffic due to the usage of a fixed port number (i.e., 9735). Moreover, LN utilizes the Noise Protocol Framework with `ChaChaPoly-1305` as the encryption scheme for the communication between nodes [43]. Initially, `ChaChaPoly` and `AES` encryption schemes were the two options considered by LN developers due to their wide adoption [61]. Eventually, `ChaChaPoly` was selected for implementation as it can be up to three times faster than `AES` [52]. However, `ChaChaPoly-1305` is not length-hiding, i.e., the length of the ciphertext reveals the approximate length of the plaintext. The Revelio attack exploits this weakness to learn the type of LN messages even when encrypted.

Internet routing. Connections between LN nodes are inherently routed over multiple ASes, enabling Revelio attacker to perform the traffic analysis in the first place. Many existing attacks, such as against Bitcoin [3], Crowds [78], I2P [24], Tor [23], [26], [33], [37], follow the same spirit, that is, combining the access to the Internet infrastructure with application-layer information for deanonymization attacks. Such cross-layer attacks are stealthy as they are usually passive and challenging to be mitigated since they do not rely on inherent protocol shortcomings. This suggests the possibility of generalizing Revelio and similar attacks into a family of passive attacks against privacy of overlay networks. We leave a detailed analysis of this family of attacks for future work.

8.3. Countermeasures

Guided by the discussed root causes, we now suggest countermeasures against the Revelio attack.

Patching LN protocols. To prevent the Revelio adversaries from simulating payment paths, LN routing algorithms can be made more unpredictable. For example, nodes can select one out of the k best paths to the destinations. A sufficiently large k would render deanonymizing the sender or receiver ineffective due to their prohibitive anonymity sets. This parameter k should be carefully tested, since if k is too small (e.g., 3 in eclair clients [1]), the obfuscation becomes ineffective [59]. In practice, such a multi-path routing mechanism may have higher payment fees, lower routing success rates, and longer routing paths if k best paths have different costs.

To contribute towards a less centralized LN network, nodes may follow suggestions from Weintraub *et al.* and avoid unknown nodes that pay the complete transaction fees when establishing channels [77].

The Revelio adversaries can only deanonymize payment amounts if they know the topology (e.g., available channels and their parameters). Eradicating this information from the protocol is impractical because the source-based payment routing in LN relies on it to be functional. Still, particular LN nodes that do not route payments (e.g., as a service) can render Revelio attacks against them more difficult by not broadcasting their channel information to the P2P network. Revelio adversaries may still observe their traffic, yet, with less information regarding their possible payment amounts.

Hardening end-to-end communication. Revelio adversaries cannot identify the LN message types if another length-hiding encryption scheme (e.g., with random padding) is used instead of `ChaChaPoly-1305`. A caveat for such an encryption scheme is a non-negligible delay added to the end-to-end communication between LN nodes. Moreover, it does not rule out the possibility of more sophisticated traffic analysis attacks (e.g., based on packet count [38], [64]), which is beyond the scope of this paper.

To make traffic filtering based on port numbers (e.g., 9735 in LN) harder, nodes can use random ports to communicate after negotiation over the default port [4]. The transition between ports should be properly obfuscated (e.g., with some randomized delays) so that no communication pattern (e.g., a new port is open shortly after a communication over the default port) exists.

Avoiding adversarial ASes. Common defenses to mitigate network-based traffic analysis attacks like Revelio include using third-party proxies (e.g., Tor, VPNs) and incorporating AS-awareness into payment routing decisions. Indeed, the proposed Revelio attack does not apply to channels between two LN nodes running Tor hidden services. However, strictly hiding behind a Tor hidden service is not foolproof against all attacks (c.f. §8.1). Moreover, routing multi-hop payment over several Tor relays between each hop may notably increase the payment latency. Proxies like VPNs provide an alternative technique for LN nodes to prevent their hosting ASes from analyzing their traffic while connecting to other nodes. Nevertheless, multi-hop payments are still partially observable by Revelio adversaries unless all nodes along the paths also use proxies. Unfortunately, providing proxies to tens of thousands of LN nodes is challenging in practice and allows deanonymization attacks from popular VPN services when most connections are routed through them.

To avoid specific ASes on the payment path, the LN routing algorithm can be AS-aware, that is, to take the

AS paths between consecutive nodes into account. AS-awareness in Tor and Bitcoin shows that this defense works in most cases, although its inaccuracies may enable more sophisticated attacks and requires additional application-based countermeasures [73].

Combining network information (e.g., AS paths between LN nodes) and defenses at other layers suggests a recipe to counter the discussed family of passive attacks. That is, cross-layer attacks require combining countermeasures from multiple layers to be effective. The complete exploration of such generic defenses for overlay networks is an exciting avenue for future work.

8.4. Ethical Considerations

Guided by ethical principles of the Menlo Report [7], we minimized interacting with all productive systems. First, we solely used the LN's testnet, a testing network that is completely segregated from the productive network (i.e., mainnet), for experimenting with the feasibility of the payment grouping attack (c.f. §7.1). Second, our deanonymization attack was tested exclusively using simulation (c.f. §7.2) and with the publicly available LN topology (c.f. §3). Third, we did not interact with the Tor network — exit nodes are selected based on statistical data provided by the Tor project itself [69].

9. Related Work

9.1. Studies on LN's P2P Topology

The P2P topology of LN has been an active topic of study. Martinazzi observed the long-standing high degree nodes after analyzing the first year data of the LN topology [48]. Seres *et al.* show that LN's topology is resilient against random failures yet structurally vulnerable to adversaries controlling important nodes [63]. Rohrer *et al.* re-confirm this finding via a more detailed analysis [58]. A recent study shows that a significant fraction of the transactions is still handled by a few highly influential nodes, and the centrality of LN topology has been increasing over the years [79]. The centralization of LN topology can also be observed by looking at the nodes' geographical locations [80] or hosting ASes [17]. Our LN's AS-level topology analysis (c.f. §3) complements these prior studies since we consider all LN nodes (i.e., both with public IPs and behind Tor) as well as the ASes on the routing path between any two nodes.

9.2. Privacy Attacks in the LN

We highlight prior attacks that also target privacy in the LN particularly. For discussions about other attacks in the LN, in other payment channel networks, and in a broader scope of other cryptocurrencies, we refer readers to the comprehensive studies by Tikhomirov *et al.* [70], by Gudgeon *et al.* [29], and by Bonneau *et al.* [14], respectively.

Our work is closely related to deanonymization attacks that target the privacy of LN payments. Béres *et al.* observe the short payment paths may statistically reveal their endpoints from their LN traffic simulator [9]. Built upon this finding, Kappos *et al.* show that an adversary can achieve reasonable accuracy by simply guessing the immediate predecessor (respectively, successor) is the sender (respectively, receiver). Given the centralization of LN topology, 1% of top degree nodes can collude and confidently identify the payment originators in a quarter of observed payments using Bayesian inference [66]. By simulating the payment routing of popular LN clients, a malicious LN node on the payment path can deanonymize the payment endpoints with high precision [36]. Alternatively, Rohrer *et al.* show that on-path adversarial nodes can also launch timing attacks on the received LN messages to infer the payment's sender and receiver [59]. While the Revelio attack comprises some similar strategies, such as timing analysis and simulating payment routes, it does not require active participation in the payments and, thus, is stealthier than existing attacks.

There is also a line of attacks that target the privacy of LN channels and nodes. Nowostawski *et al.* identify channel opening and closing by looking for specific Bitcoin transaction types on the blockchain [54]. Kappos *et al.* use the same methodology to identify private channels (i.e., between two unknown LN nodes) [34]. These on-chain transactions can further be used to cluster Bitcoin wallets and LN nodes into the same entity [60]. Aiming to learn the channel balances (e.g., that lead to large-scale payment deanonymization [34]), several probing attacks have been proposed. The strategies include sending payments through that channel that trigger failures [31], [53], [71], [81], sending payments only between adversarial nodes [20], [34], and jamming the channels when nodes have multiple channels in parallel [12].

9.3. Network-level Privacy Attacks

The Perimeter attack [3] has the same attack capabilities (e.g., being an AS) and a similar goal (i.e., deanonymizing cryptocurrency transactions) as our attack. However, the Revelio attack is purely passive and works in the presence of Tor-based connections, whereas Perimeter only works with unencrypted traffic or with extra connections to the victim's peers. Besides cryptocurrencies, anonymous overlay networks like Tor have been the prominent target for network-level privacy attacks (e.g., [23], [26], [33], [37], [68]). Like Revelio, these attacks commonly deanonymize users by analyzing the observed Tor traffic. For an extended discussion on other network-level attacks, interested readers are referred to a recent article by Sun *et al.* [67].

10. Conclusion

The Lightning Network has significantly contributed to cryptocurrency adoption thanks to its cheap, fast, and anonymous payments. This paper challenges payment privacy in the LN with a new threat of AS-level adversaries, who stealthily analyze network traffic. Our evaluation of the real-world data shows that LN topology is worryingly centralized at the AS-level, which allows almost one-third of payments to be deanonymized by this novel attack. We hope this work guides the direction for developing additional countermeasures against network-level attacks in not only the LN but also other payment channel networks.

References

[1] ACINQ, *Eclair*, 2023. https://github.com/ACINQ/eclair.

[2] J. Angwin, C. Savage, J. Larson, H. Moltke, L. Poitras, and J. Risen, "AT&T Helped U.S. Spy on Internet on a Vast Scale," *The New York Times*, 2015.

[3] M. Apostolaki, C. Maire, and L. Vanbever, "Perimeter: A Network-Layer Attack on the Anonymity of Cryptocurrencies," in *FC*, 2021.

[4] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking Bitcoin: Routing Attacks on Cryptocurrencies," in *IEEE S&P*, 2017.

[5] Arcane Research, "The State of Lightning: Volume 2," 2022.

[6] A. Arnbak and S. Goldberg, "Loopholes for Circumventing the Constitution: Warrantless Bulk Surveillance on Americans by Collecting Network Traffic Abroad," in *HotPETs*, 2014.

[7] M. Bailey, D. Dittrich, E. Kenneally, and D. Maughan, "The Menlo Report," *IEEE S&P*, 2012.

[8] P. Bajpai, E. Rasure, and V. Velasquez, *Countries Where Bitcoin Is Legal and Illegal*, 2022. https://www.investopedia.com/articles/forex/041515/countries-where-bitcoin-legal-illegal.asp.

[9] F. Béres, I. A. Seres, and A. A. Benczúr, "A Cryptoeconomic Traffic Analysis of Bitcoin's Lightning Network," *Cryptoeconomic Systems*, 2021.

[10] O. Berthold, A. Pfitzmann, and R. Standtke, "The Disadvantages of Free MIX Routes and How to Overcome Them," *DIAU*, 2001.

[11] H. Birge-Lee, Y. Sun, A. Edmundson, J. Rexford, and P. Mittal, "Bamboozling certificate authorities with BGP," in *USENIX Security*, 2018.

[12] A. Biryukov, G. Naumenko, and S. Tikhomirov, "Analysis and Probing of Parallel Channels in the Lightning Network," in *FC*, 2022.

[13] A. Biryukov and S. Tikhomirov, "Deanonymization and Linkability of Cryptocurrency Transactions Based on Network Analysis," in *IEEE EuroS&P*, 2019.

[14] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "Sok: Research perspectives and challenges for bitcoin and cryptocurrencies," in *IEEE S&P*, 2015.

[15] CAIDA, *AS Relationships Dataset*, 2022. https://www.caida.org/catalog/datasets/as-relationships/.

[16] CAIDA, *Internet eXchange Points (IXPs) Dataset*, 2022. https://www.caida.org/catalog/datasets/ixps/.

[17] P. Casas, M. Romiti, P. Holzer, S. B. Mariem, B. Donnet, and B. Haslhofer, "Where is the Light(ning) in the Taproot Dawn? Unveiling the Bitcoin Lightning (IP) Network," in *IEEE CloudNet*, 2021.

[18] Coincub, *Coincub annual crypto tax ranking 2022*, 2022. https://coincub.com/ranking/coincub-annual-crypto-tax-ranking-2022/.

[19] Core Lightning, *CLN*, 2023. https://github.com/ElementsProject/lightning.

[20] G. van Dam, R. A. Kadir, P. N. E. Nohuddin, and H. B. Zaman, "Improvements of the Balance Discovery Attack on Lightning Network Payment Channels," in *IFIP SEC*, 2020.

[21] G. Danezis and I. Goldberg, "Sphinx: A Compact and Provably Secure Mix Format," in *IEEE S&P*, 2009.

[22] G. Danezis and A. Serjantov, "Statistical Disclosure or Intersection Attacks on Anonymity Systems," in *IH*, 2005.

[23] M. Edman and P. Syverson, "AS-awareness in Tor path selection," in *ACM CCS*, 2009.

[24] C. Egger, J. Schlumberger, C. Kruegel Kruegel, and G. Vigna, "Practical Attacks Against the I2P Network," in *RAID*, 2013.

[25] G. Fanti and P. Viswanath, "Deanonymization in the Bitcoin P2P Network," in *NeurIPS*, 2017.

[26] N. Feamster and R. Dingledine, "Location diversity in anonymity networks," in *ACM WPES*, 2004.

[27] B. Gellman, C. Timberg, and S. Rich, "Secret NSA documents show campaign against Tor encrypted network," *The Washington Post*, 2013.

[28] P. Gill, M. Schapira, and S. Goldberg, "A survey of interdomain routing policies," *ACM SIGCOMM CCR*, 2013.

[29] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais, "Sok: Layer-two blockchain protocols," in *FC*, 2020.

[30] J. Hayes and G. Danezis, "k-fingerprinting: A Robust Scalable Website Fingerprinting Technique," in *USENIX Security*, 2016.

[31] J. Herrera-Joancomarti, G. Navarro-Arribas, A. Ranchal-Pedrosa, C. Perez-Sola, and J. Garcia-Alfaro, "On the difficulty of hiding the balance of lightning network channels," in *ACM AsiaCCS*, 2019.

[32] R. Jansen, F. Tschorsch, A. Johnson, and B. Scheuermann, "The Sniper Attack: Anonymously Deanonymizing and Disabling the Tor Network," in *NDSS*, 2014.

[33] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson, "Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries," in *ACM CCS*, 2013.

[34] G. Kappos, H. Yousaf, A. Piotrowska, *et al.*, "An Empirical Analysis of Privacy in the Lightning Network," in *FC*, 2021.

[35] P. Koshy, D. Koshy, and P. McDaniel, "An analysis of anonymity in bitcoin using P2P network traffic," in *FC*, 2014.

[36] S. P. Kumble, D. Epema, and S. Roos, "How lightning's routing diminishes its anonymity," in *ARES*, 2021.

[37] A. Kwon, M. AlSabah, D. Lazar, M. Dacier, and S. Devadas, "Circuit Fingerprinting Attacks: Passive Deanonymization of Tor Hidden Services," in *USENIX Security*, 2015.

[38] B. N. Levine, M. K. Reiter, C. Wang, and M. K. Wright, "Timing Attacks in Low-Latency Mix Systems," in *FC*, 2004.

[39] Lightning Labs, *lnd*, 2023. https://github.com/lightningnetwork/lnd.

[40] Lightning Network Specifications, *BOLT #1: Base Protocol*, 2023.

[41] Lightning Network Specifications, *BOLT #2: Peer Protocol for Channel Management*, 2023.

[42] Lightning Network Specifications, *BOLT #4: Onion Routing Protocol*, 2023.

[43] Lightning Network Specifications, *BOLT #8: Encrypted and Authenticated Transport*, 2023.

[44] Z. Ling, J. Luo, K. Wu, and X. Fu, "Protocol-level Hidden Server Discovery," in *IEEE INFOCOM*, 2013.

[45] LNBIG, *Payment volume for the last 24 hours*, 2022. https : / / twitter . com / lnbig _ com / status / 1518997142374797312.

[46] Z. M. Mao, L. Qiu, J. Wang, and Y. Zhang, "On AS-level path inference," in *ACM SIGMETRICS PER*, 2005.

[47] B. Marczak, N. Weaver, J. Dalek, *et al.*, "An Analysis of China's "Great Cannon"," in *USENIX FOCI*, 2015.

[48] S. Martinazzi, "The Evolution of Lightning Network's Topology During its First Year and the Influence Over its Core Values," 2019.

[49] A. Mizrahi and A. Zohar, "Congestion Attacks in Payment Channel Networks," in *FC*, 2021.

[50] S. J. Murdoch, "Hot or Not: Revealing Hidden Services by Their Clock Skew," in *ACM CCS*, 2006.

[51] G. Nakibly, J. Schcolnik, and Y. Rubin, "Website-Targeted False Content Injection by Network Operators," in *USENIX Security*, 2016.

[52] Y. Nir and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols," RFC 8439, 2018.

[53] U. Nisslmueller, K.-T. Foerster, S. Schmid, and C. Decker, "Toward Active and Passive Confidentiality Attacks On Cryptocurrency Off-Chain Networks," in *ICISSP*, 2020.

[54] M. Nowostawski and J. Tøn, "Evaluating Methods for the Identification of Off-Chain Transactions in the Lightning Network," *Applied Sciences*, 2019.

[55] L. Øverlier and P. Syverson, "Locating Hidden Servers," in *IEEE S&P*, 2006.

[56] A. Panchenko, A. Mitseva, M. Henze, F. Lanze, K. Wehrle, and T. Engel, "Analysis of Fingerprinting Techniques for Tor Hidden Services," in *WPES*, 2017.

[57] J. Poon and T. Dryja, "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments," 2016.

[58] E. Rohrer, J. Malliaris, and F. Tschorsch, "Discharged Payment Channels: Quantifying the Lightning Network's Resilience to Topology-Based Attacks," in *IEEE S&B*, 2019.

[59] E. Rohrer and F. Tschorsch, "Counting Down Thunder: Timing Attacks on Privacy in Payment Channel Networks," in *ACM AFT*, 2020.

[60] M. Romiti, F. Victor, P. Moreno-Sanchez, P. S. Nordholt, B. Haslhofer, and M. Maffei, "Cross-Layer Deanonymization Methods in the Lightning Protocol," in *FC*, 2021.

[61] R. Russell, *Laundry list of inter-peer wire protocol changes*, Lightning-dev Mailinglist, 2016. https://lists.linuxfoundation.org/pipermail/lightning-dev/2016-January/000408.html.

[62] M. Saad and D. Mohaisen, "Three Birds with One Stone: Efficient Partitioning Attacks on Interdependent Cryptocurrency Networks," in *IEEE S&P*, 2023.

[63] I. A. Seres, L. Gulyás, D. A. Nagy, and P. Burcsi, "Topological Analysis of Bitcoin's Lightning Network," in *MARBLE*, 2020.

[64] A. Serjantov and P. Sewell, "Passive Attack Analysis for Connection-Based Anonymity Systems," in *ESORICS*, 2003.

[65] P. Sermpezis, V. Kotronis, P. Gigis, *et al.*, "ARTEMIS: Neutralizing BGP hijacking within a minute," in *IEEE/ACM Trans. Netw.*, 2018.

[66] P. K. Sharma, D. Gosain, and C. Diaz, "On the Anonymity of Peer-To-Peer Network Anonymity Schemes Used by Cryptocurrencies," in *NDSS*, 2023.

[67] Y. Sun, M. Apostolaki, H. Birge-Lee, *et al.*, "Securing internet applications from routing attacks," in *Communications of the ACM*, 2021.

[68] Y. Sun, A. Edmundson, L. Vanbever, *et al.*, "RAPTOR: Routing Attacks on Privacy in Tor," in *USENIX Security*, 2015.

[69] The Tor Project, *Tor Metrics*, 2022. https://metrics.torproject.org/.

[70] S. Tikhomirov, P. Moreno-Sanchez, and M. Maffei, "A Quantitative Analysis of Security, Anonymity and Scalability for the Lightning Network," in *IEEE EuroS&PW*, 2020.

[71] S. Tikhomirov, R. Pickhardt, A. Biryukov, and M. Nowostawski, *Probing Channel Balances in the Lightning Network*, 2020.

[72] S. Tochner, S. Schmid, and A. Zohar, "Hijacking Routes in Payment Channel Networks: A Predictability Tradeoff," 2019.

[73] M. Tran, A. Shenoi, and M. S. Kang, "On the Routing-Aware Peering against Network-Eclipse Attacks in Bitcoin," in *USENIX Security*, 2021.

[74] M. C. Tschantz, S. Afroz, V. Paxson, *et al.*, "Sok: Towards grounding censorship circumvention in empiricism," in *IEEE S&P*, 2016.

[75] University of Oregon, *Route Views Archive Project*, 2022. http://archive.routeviews.org/.

[76] T. Voegtlin, *Electrum*, 2023. https://github.com/spesmilo/electrum.

[77] B. Weintraub, C. Nita-Rotaru, and S. Roos, "Structural Attacks on Local Routing in Payment Channel Networks," in *IEEE EuroS&PW*, 2021.

[78] M. K. Wright, M. Adler, B. N. Levine, and C. Shields, "The predecessor attack: An analysis of a threat to anonymous communications systems," *ACM TISSEC*, 2004.

[79] P. Zabka, K.-T. Foerster, C. Decker, and S. Schmid, "Short Paper: A Centrality Analysis of the Lightning Network," in *FC*, 2022.

[80] P. Zabka, K.-T. Förster, S. Schmid, and C. Decker, "Node classification and geographical analysis of the lightning cryptocurrency network," in *ICDCN*, 2021.

[81] Z. Zhao, C. Ge, L. Zhou, and H. Wang, "Fully Discover the Balance of Lightning Network Payment Channels," in *WASA*, 2021.