Analysis and Probing of Parallel Channels in the Lightning Network

Alex Biryukov[1], Gleb Naumenko[2], and Sergei Tikhomirov[1]

[1] University of Luxembourg
alex.biryukov@uni.lu, sergey.s.tikhomirov@gmail.com
[2] thelab31.xyz
gleb@thelab31.xyz

Abstract. Bitcoin can process only a few transactions per second, which is insufficient for a global payment network. The Lightning Network (LN) aims to address this challenge. The LN allows for low-latency bitcoin transfers through a network of payment channels. In contrast to regular Bitcoin transactions, payments in the LN are not globally broadcast. Thus it may improve not only Bitcoin's scalability but also privacy. However, the probing attack allows an adversary to discover channel balances, threatening users' privacy. Prior work on probing did not account for the possibility of multiple (parallel) channels between two nodes. Naive probing algorithms yield false results for parallel channels.
In this work, we develop a new probing model that accurately accounts for parallel channels. We describe jamming-enhanced probing that allows for full balance information extraction in multi-channel hops, which was impossible with earlier probing methods. We quantify the attacker's information gain and propose an optimized algorithm for choosing probe amounts for N-channel hops. We demonstrate its efficiency based on real-world data using our own probing-focused LN simulator. Finally, we discuss countermeasures such as new forwarding strategies, intra-hop payment split, rebalancing, and unannounced channels.

Keywords: Lightning network · Bitcoin · payment channels · privacy

1 Introduction

To ensure public verifiability on widely available hardware, the throughput of Bitcoin [20] is limited to around 7 transactions per second. Second-layer (L2) protocols [9] aim to address this issue. The most prominent L2 protocol for Bitcoin is a payment channel network called the Lightning Network [25]. A payment channel is a trust-minimized two-party protocol for low-latency bitcoin[3] payments [12] with minimal interaction with the base layer. A channel network allows for multi-hop payments between users who do not share a channel.

Bitcoin transactions are globally broadcast, harming users' privacy [1, 18]. This is not the case for L2 payments, hence the LN may also be seen as a privacy-enhancing technology. However, attacks on LN privacy have been described,

[3] Similar protocols are possible for other cryptocurrencies.

including balance probing. Probing allows for cheaply revealing channel balances by sending fake payments (probes) [13, 16, 42, 39]. Probing can be used as a building block to spy on payments or node balances, or to deanonymize LN users. Prior work on probing assumed at most one channel between each pair of nodes[4]. However, the LN allows multiple parallel channels between the same pair of nodes. Naive probing algorithms may give false results for multi-channel hops. Moreover, for certain configurations of parallel channel capacities and balances, full balance extraction may be impossible.

Our contributions After providing the necessary background (Section 2), we introduce the probing model (Section 3) and propose N-dimensional binary search (NBS) to choose probe amounts that maximize probing speed. We enhance the probing attack by combining it with jamming or fee selection. Using simulations based on a real-world data, we show that enhanced probing extracts full balance information in parallel channels, which was impossible with earlier probing methods (Section 4). Moreover, NBS increases probing speed by up to 15%, compared to single-dimensional binary search (BS). We discuss model limitations, attack cost and trade-offs, payment flow discovery, and countermeasures in Section 5. We review related work in Section 6 and conclude in Section 7.

2 Background

To open a payment channel, Alice and Bob lock coins into a cooperatively owned Bitcoin address, establishing the initial channel state. To make a payment, the parties negotiate a new state, thereby provably invalidating the old one [9]. Any party can close the channel and withdraw their coins on-chain at any time.

The total number of coins in a channel, constant throughout its life, is called *capacity*. The number of coins owned by one party is called its *balance* and changes as payments are made. We refer to a pair of adjacent nodes along with all channels they share as a *hop*. Parallel channels may have different security and fee policies [3]. A node may disable a channel direction (e.g., before an expected loss of connectivity or channel settlement), making the channel *unidirectional*[5].

An LN user can send *multi-hop payments* without establishing a channel with the receiver. Nodes gossip about availability, capacities, and fee policies of public channels[6]. The receiver generates a *payment secret* and sends a hash of it (the *payment hash*) to the sender. The sender chooses the payment path (an ordered list of nodes) based on its local view of the network graph[7]. If an intermediary hop in the path contains parallel channels, a routing node may choose any of

[4] The paper [21] writes: "Our tool failed to produce accurate results in this [multi-channel hops] scenario [...] further research on how to deal with this complication would be highly appreciated."

[5] Not to be confused with an earlier unidirectional channel construction [12].

[6] Users may keep their channels unannounced. A 2020 study estimated that 28.7% of LN channels were unannounced [26].

[7] Alternative approaches are trampoline [36] and rendezvous routing [44].

them (*non-strict forwarding*). Upon receiving a payment, the receiver propagates the payment secret along the path back to the sender. This ensures that balances along the path are shifted atomically as they all depend on the same secret being revealed[8]. LN nodes are only aware of payments that they forward. Intermediary nodes see the amount as well as the immediate previous and next node in the path, but not the ultimate sender and receiver, due to onion routing.

The sender only knows the *capacities* of remote channels, but their forwarding ability is determined by their *balances*. Therefore, multi-hop payment attempts may fail due to low balance at an intermediary hop. In that case, the erring node notifies the sender which error has occurred and where. The sender may have to make multiple payment attempts using different paths until one succeeds.

The three major LN implementations – LND, C-LIGHTNING, and ECLAIR – use different channel selection strategies when routing payments through a multi-channel hop. ECLAIR selects the channel with the lowest capacity (among the channels with the same capacity, it prefers a lower balance)[9]. LND chooses a random channel[10]. C-LIGHTNING does not support parallel channels.

Attacks on Lightning Multiple attacks on the LN have been described (see Section 6). Most relevant for our work are probing and jamming.

Probing allows an attacker to reveal the balance of any forwarding channel (assuming no multi-channel hops) by sending probes through it [13, 16, 39]. A probe is a payment with amount a that contains a random number instead of a payment hash. A probe fails either at an intermediary node due to insufficient balance, or at the receiver because it does not know the preimage of the payment hash[11]. The location of the erring node within the path reveals whether the probe has reached the receiver, hence, whether the balance of the target channel is above or below a. The attacker sends probes with different amounts to infer the target channel balance with high accuracy. Assuming uniform balance distribution, the best strategy for choosing probe amounts is binary search.

Jamming is a family of denial-of-service attacks on LN channels [7, 37]. An attacker initiates a payment along a route that goes through a target channel and terminates at another node controlled by the attacker, and refuses to reveal the payment secret, locking the funds along the path. Shortly before timelocks expire, the attacker fails the payment to release their coins without paying routing fees. In *capacity-based jamming*, an attacker initiates payments of a given (presumably high) value [22]. In *slot-based jamming*, an attacker sends a series of small[12] payments to reach the limit of *payment slots* for in-flight payments (at most 483 in each direction; channel parties may set lower limits) [38]. We re-

[8] It may be argued though that the wormhole attack [17] violates atomicity.

[9] `https://github.com/ACINQ/eclair/blob/5f9d0d/eclair-core/src/main/scala/fr/acinq/eclair/payment/relay/ChannelRelay.scala\#L199`

[10] `https://github.com/lightningnetwork/lnd/blob/f98a3c/htlcswitch/switch.go\#L1091`

[11] We do not consider other potential errors for simplicity.

[12] Above the dust limit of 546 satoshis.

fer to jamming payments as *jams*. Onion routing complicates protection against jamming: the victim does not know who is sending the jams.

3 Probing model

We assume the following threat model. The goal of the attacker is to reveal exact channel balances in target hops as quickly as possible[13]. The attacker only uses public knowledge about nodes and channels. The attacker can run multiple LN nodes, open channels, and maintain them for the duration of the attack[14]. The attacker can run modified software but has no control over other users' software.

We define channel direction as follows. Direction *dir0* is the direction from the node with the alphanumerically smaller ID to the other node. Direction *dir1* is the opposite. We define channel balance (in satoshis[15]) as the balance of the node with the alphanumerically smaller ID. Note that the *dir0* / *dir1* notation is defined by random node IDs. It neither depends on the viewpoint (local / remote) nor on who opened the channel (inbound / outbound).

A hop with N channels is characterized by channel capacities $C = (c_1, \ldots, c_N)$ and balances $B = (b_1, \ldots, b_N)$. Let E^d be the set of channels enabled in direction d, where $d \in \{dir0, dir1\}$. The forwarding ability of a hop is determined by the maximal balances among the channels enabled in a given direction, which we denote as h for *dir0* and g for *dir1*: $h = \max_{i \in E^{dir0}} b_i$; $g = \max_{i \in E^{dir1}} (c_i - b_i)$. C, E^{dir0}, E^{dir1} are public knowledge; B, h, and g are private.

In the general case, probes only give the attacker information about h or g, not about individual balances[16]. The attacker maintains the current lower and upper bounds[17] for h and g: $h^l < h \leq h^u$ and $g^l < g \leq g^u$, initially set to $h^l = g^l = -1$, $h^u = \max_{i \in E^{dir0}} c_i$, $g^u = \max_{i \in E^{dir1}} c_i$.

Let F be the set of all possible values of B, as per the attacker's current knowledge. $S(F)$ is the number of values F contains. Each probe cuts F in two parts, one of which is excluded from further consideration. Assuming uniform balance distribution, an optimal probe should cut F in half.

3.1 Examples

As the simplest example, consider a hop with one channel of capacity c (Figure 1). Let b^l and b^u be the current lower and upper bounds for the true balance b, respectively. Initially, $b^l = 0$ and $b^u = c$. $F = [b^l, b^u]$. The prober chooses $a = (b^l + b^u)/2$. If the probe fails, F is updated to $[b^l, a]$, otherwise to $[a, b^u]$.

[13] We assume that all target channels are equally interesting for the attacker.

[14] Sending one probe normally takes a few seconds.

[15] The LN operates with millisatoshi precision off-chain, but such amounts cannot be settled on-chain. For simplicity, our model operates with a satoshi-level precision.

[16] Enhanced probing techniques described in Section 3.4 overcome this limitation.

[17] Note that for lower bound is strict, and the upper bound is non-strict. If the probe of amount a in direction *dir0* succeeds, h is *greater or equal* to a, but if the probe fails, it is *strictly less* than a (same with g and *dir1*). Our definitions reflect this asymmetry and thus allow for uniform calculations when deriving Equation 1.

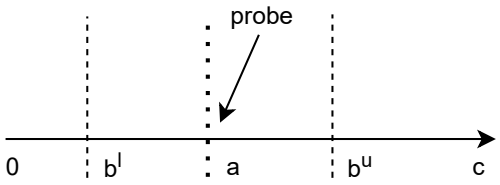

Fig. 1. Probing one-channel hop with single-dimensional binary search.

Next, consider a two-channel hop with capacities $c_1 = c_2 = c$ and balances b_1, b_2 (Figure 2). Initially, $S(F) = c^2$. The probe value a should cut F in half: $a = c/\sqrt{2}$. Note that $a = c/2$ would be suboptimal, as it divides $S(F)$ in the proportion 3:1, not 1:1.

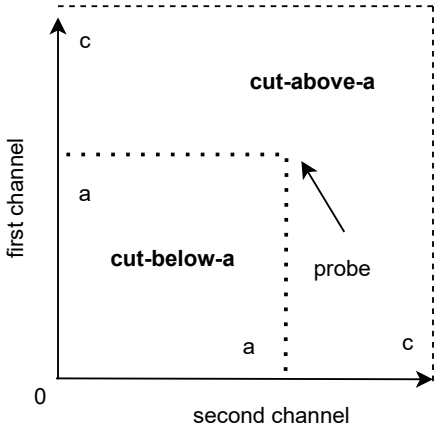

Fig. 2. Probing two-channel hop with two-dimensional binary search.

3.2 Generalized geometrical model

In the general case, we consider an N-channel hop, where some channels may be disabled in some directions. The prober wants to shrink F using as few probes as possible. The key task is to choose each next probe amount optimally, regardless of hop properties and current bounds.

We can think of an N-channel hop as an N-dimensional (hyper-)rectangle R, with sides parallel to the axes[18]. Each side corresponds to one channel. The i-th side is defined by the coordinates $[0, c_i]$.

[18] The model applies to N dimensions, but we continue using two-dimensional terms such as "rectangle" and "area" for clarity.

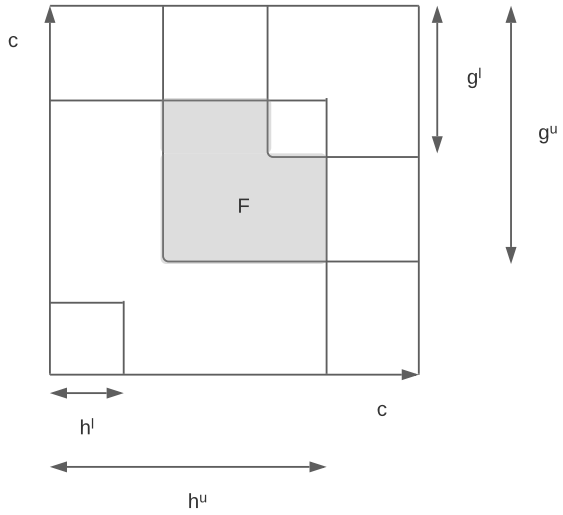

Fig. 3. Defining F using probe rectangles.

A probe with amount a "cuts" an a-sided cube from one of the two opposite corners of R (depending on probe direction). If the probe fails, all coordinates of B are lower than a (a new upper bound[19]), otherwise at least one coordinate of B is greater than or equal to a (a new lower bound). We can define F in terms of four rectangles (Figure 3 illustrates a two-dimensional case) corresponding to the four bounds: h^l, h^u, g^l, g^u. Denoting $\overline{x} = x + 1$ and using subscript i for the i-th coordinate, we calculate $S(F)$ as follows (for full derivation, see Appendix A):

$$S(F) = \prod_{i=1}^{N}(\overline{h_i^u}+\overline{g_i^u}-\overline{c_i}) - \prod_{i=1}^{N}(\overline{h_i^l}+\overline{g_i^u}-\overline{c_i}) - \prod_{i=1}^{N}(\overline{h_i^u}+\overline{g_i^l}-\overline{c_i}) + \prod_{i=1}^{N}(\overline{h_i^l}+\overline{g_i^l}-\overline{c_i}) \quad (1)$$

In prior probing algorithms, each next amount a is chosen as the mid-point between the current lower and upper bounds (*simple binary search*, or BS). BS is suboptimal in the multi-dimensional case, as illustrated earlier (Section 3.1). *N-dimensional binary search* (NBS) chooses a that cuts F in half[20]. Let S_a be the area under the cut. The prober finds a such that $S_a = \frac{S(F)}{2}$ as follows. Initially, set $a^l = h^l + 1$, $a^u = h^u$, and consider a candidate value $a = (a^l + a^u)/2$ (for *dir0*; *dir1* in handled analogously). If $S_a < \frac{S}{2}$, set $a^l = a$, else set $a^u = a$. Repeat until S_a is close enough[21] to $\frac{S}{2}$.

[19] More precisely, we use *effective amounts* as bounds, as defined in Appendix A.
[20] For $N = 1$, NBS is equivalent to BS.
[21] It is not always possible to cut F in half precisely, i.e., when $S(F)$ is odd.

3.3 Metrics

The uncertainty U of a hop is the number of bits required to encode the position of B, given the current attacker's knowledge. As the result of probing, U decreases from $U_{before} = \sum_{i=1}^{N} \log_2(c_i + 1)$ to $U_{after} = \log_2(S(F))$. For a set T of target hops, the final achieved information gain is:

$$I = 1 - \sum_{t \in T}(U^t_{after}) / \sum_{t \in T}(U^t_{before}) \tag{2}$$

Assuming m messages sent in total, the probing speed is defined as:

$$S = \frac{\sum_{t \in T}(U^t_{before}) - \sum_{t \in T}(U^t_{after})}{m} \tag{3}$$

Messages include probes (including those that did not reach the target hop, for remote probing) and jams (for jamming-enhanced probing).

3.4 Enhanced probing

There are two reasons why information gain for multi-channel hops may be bounded.

First, if there is a large difference between channel capacities in a hop, larger channels can "mask" smaller ones by forwarding all probes that go through the hop. This case is illustrated in Figure 4: no probe can shrink F along the vertical axis, as all probes go through the channel with the larger capacity c_1.

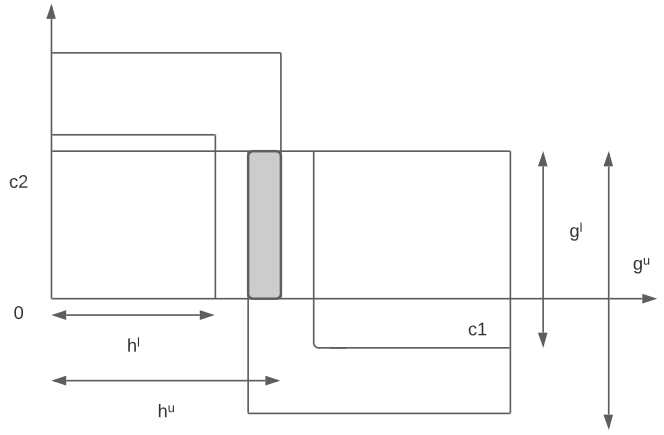

Fig. 4. The challenge of probing a multi-channel hop with different channel capacities.

Second, hops with more than two channels cannot be fully probed due to dimensionality. To get an intuition why, consider the intersection of two cubes

that represent probes in a 3-channel hop with equal capacities c (Figure 5). The green and blue cubes correspond to precise bounds on h and g, respectively (thus further probes would give no new information). F is their intersection (six red intervals). It is impossible to shrink F into single points, in contrast to lower-dimensional cases[22].

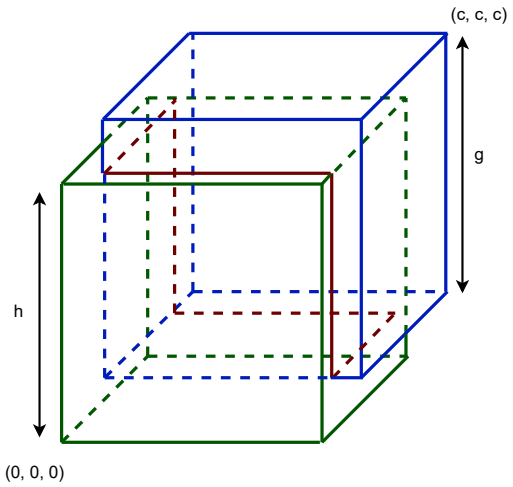

Fig. 5. A 3-channel hop that cannot be fully probed due to dimensionality.

The only way for the attacker to gain more balance information in these cases would be to force probes to go through specific channels. The attacker cannot affect how other nodes choose channels, but can reduce the set of *suitable* channels the victim picks from.

We consider two probing enhancement techniques to achieve this goal. In *jamming-enhanced probing*, the prober jams all channels in a target hop except one, and then probes the remaining channel. In *fee-aware probing* [27], the prober sets the fee offered along with the probe such that the probe can only be forwarded through a subset of cheapest channels in the target hop. In the best case (for the attacker), fees for all channels in the target hop are different. In the worst case, all channels require equal fees, and fee-aware probing yields no advantage. Jamming-enhanced and fee-aware probing may be combined, which allows for probing individual channels inside one fee level. More generally, the prober may tune other parameters, such as timeouts, instead of or in addition to fee levels (*policy-aware probing*).

We use an isolated testing environment based on real LN nodes to confirm that enhanced probing indeed allows a prober to infer individual balances of parallel channels. Setup details are provided in Appendix B.

[22] The only exception is a degenerate case when all three balances are equal.

4 Evaluation

4.1 Data source

We captured an LN snapshot using our own C-LIGHTNING node on 2021-09-09. The snapshot contains 13627 nodes and 65824 channels[23] with a total capacity of 2505 BTC. This is in line with public explorers such as the one ran by ACINQ[24] (the developers of ECLAIR), which on the same day reported 13778 nodes and 65909 channels. 52985 channels (80%) are enabled in both directions. Multi-channel hops hold a disproportionately large share of capacity (Table 1) and thus presumably play a more important role in routing than single-channel hops.

Channels in a hop	**Share of hops (%)**	**Share of capacity (%)**
1	94.1	71.6
2	4.9	11.3
3	0.7	6.7
≥ 4	0.3	10.4

Table 1. Share of hops by the number of channels and by total capacity.

4.2 Results

For each channel in the snapshot, we generate a balance uniformly at random between 0 and the channel capacity. We probe randomly chosen target hops with various probing configurations in the simulator. We average the results across multiple experiment runs. We only consider hops with 1 to 5 channels (hops with more channels are rare in the snapshot). We measure the information gain and probing speed for two probe amount choice methods (BS and NBS) and two types of probing (direct and remote). In *direct probing*, the attacker opens a channel to one of the parties of the target hop and sends probes via the 2-hop path, avoiding failed routes (all probes reach the target hop). Opening a channel requires paying on-chain fees, locking up capital, and cooperation of the counterparty (though public nodes usually allow opening channels to them if a user fully funds it). In *remote probing*, the attacker sends probes along multi-hop routes. This introduces potential routing errors, especially for larger amounts. On the other hand, each probe yields information not only about the target hop but also about intermediary hops. The attacker accumulates this data to avoid sending probes via hops that have previously failed routing smaller amounts.

First, consider the information gain. For non-enhanced probing, the information gain decreases as N increases (Figure 6), as expected (see Section 3.4).

[23] We only consider the largest connected component, which contains 99.1% of nodes and 99.9% of channels.

[24] `https://explorer.acinq.co/`

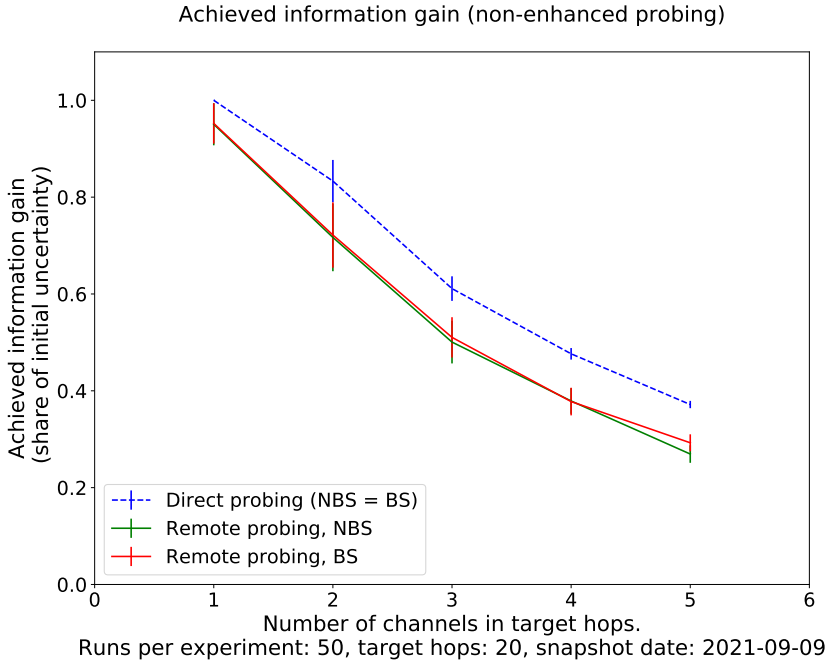

Fig. 6. Final information gain for non-enhanced probing.

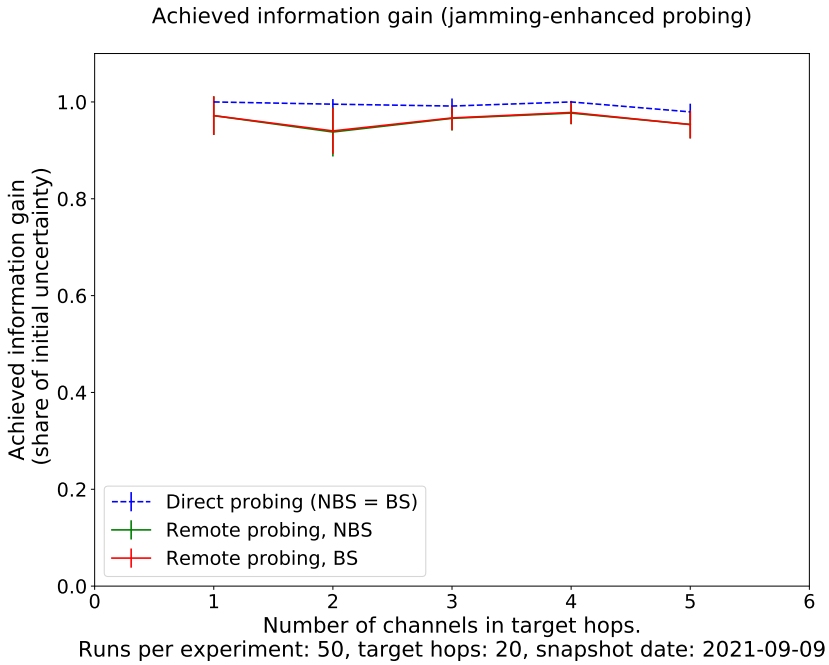

Fig. 7. Final information gain for jamming-enhanced probing.

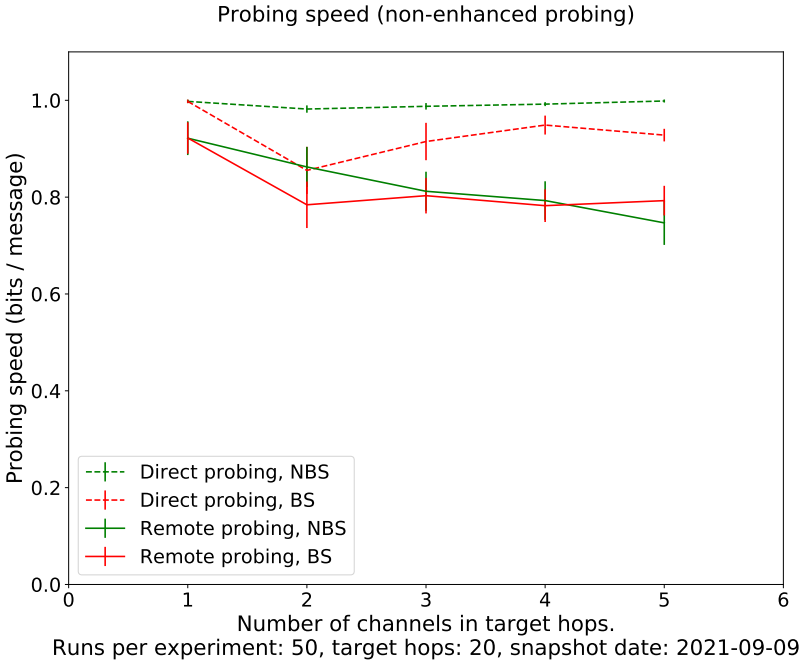

Fig. 8. Probing speed for non-enhanced probing.

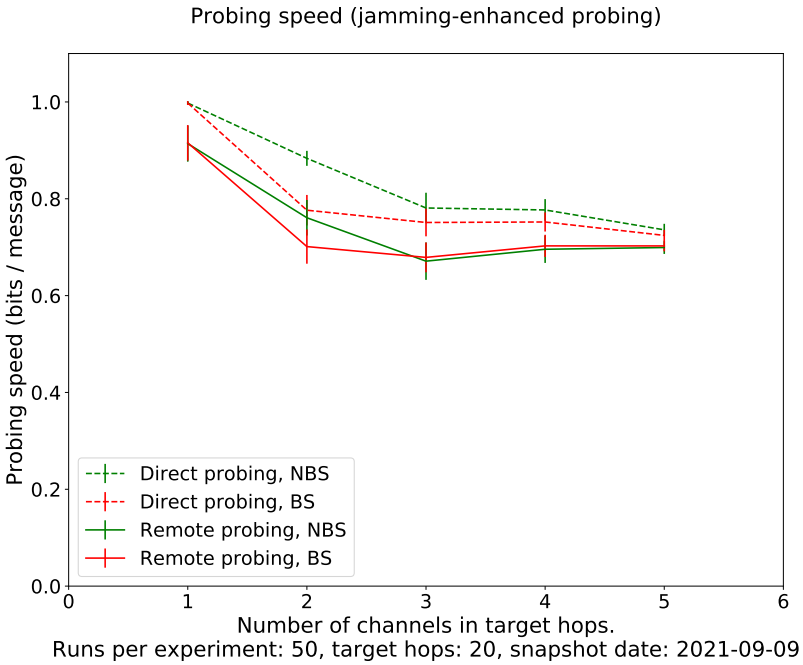

Fig. 9. Probing speed for jamming-enhanced probing.

E.g., 5-channel hops can only be probed up to 0.4 information gain. This applies for direct as well as remote probing. The difference in information gain between BS and NBS is small. Jamming-enhanced probing achieves high information gain (above 0.9) for all values of N (Figure 7). We thus confirm that jamming allows the prober to extract balance information that is otherwise unavailable. Lower information gain for remote probing is explained by routing issues.

In terms of probing speed, direct probing consistently outperforms remote probing, as no probes are wasted (Figures 8 and 9). NBS outperforms BS for all values of N in direct probing and for $N = 2$ for remote probing. For remote probing, BS and NBS perform similarly at $N = 3$ and $N = 4$. For $N = 5$, BS slightly outperforms NBS, which is explained by the fact NBS generally chooses higher amounts (e.g., $1/2$ vs $1/\sqrt{2}$ in a 1-by-1 square) that are more likely to fail in multi-hop probes. This effect is more pronounced for high values of N.

We also observe that jamming-enhanced probing (Figure 9) lowers the probing speed compared to non-enhanced probing (Figure 8). This is expected because jamming-enhanced probing implies sending jams in addition to probes. NBS outperforms BS for all values of N in direct probing and for $N = 2$ for remote probing (for $N > 2$, the two methods perform similarly).

Additional simulations show how the ratio between capacities in two-channel hops affects information gain (see Appendix C).

5 Discussion

The simulations have demonstrated that jamming-enhanced probing allow to extract nearly full balance information, which is otherwise impossible for hops with three channels or more, and that NBS amount selection increases probing speed. We now discuss the limitations of our model and avenues for future work.

5.1 Limitations

Our model ignores regular LN activity. If a target hop is heavily used, balances may shift between probes, making attacker's estimations incorrect. This is one of the reasons why minimizing the attack time is important: it reduces the probability of interference with honest payments. Moreover, we do not model in-flight payments. Our model assumes that the two channel balances sum up to its capacity, which allows us to derive one balance from the other. In the real network, channel capacity is composed of the two balances and in-flight payments. We assume that in-flight payments resolve quickly enough to have no effect on probing results. We also do not account for LN routing fees in multi-hop probes.

We make some simplifying assumptions about jamming. First, we assume that the attacker can jam any hop. In practice, jamming requires additional liquidity and channel slots, which may be unavailable. Second, we assume that the attacker can jam a specific channel within a remote hop. In practice, intermediary nodes choose which parallel channel to forward the jam through (just like with regular payments). As a result, the attacker only knows how many

channels are jammed but does not know which ones. Even if the attacker derives N channel balances exactly, they are only known up to a permutation. Third, we assume that the attacker can jam channels in both directions. In practice, leaf hops can only be jammed in one direction[25].

5.2 Attack cost and trade-offs

Probing is relatively cheap. The attacker pays on-chain fees for opening and closing channels, but never pays LN routing fees, because probing payments never complete. There is a trade-off between direct and remote probing. Direct probing increases probing speed but requires more on-chain fees and locked capital. We leave the evaluation of this trade-off for future work.

Jamming-enhanced probing brings additional costs. Capacity-based jamming requires at least one high-capacity channel. The amount of funds locked is close to the aggregate balance of all parallel target channels. Slot-based jamming requires opening many low-capacity channels. The exact number of attacker's channels equals the number of channels to be jammed: the attacker's path is limited by the same number of slots[26].

Jamming might be challenging for certain hop configurations. For example, it would be impossible to slot-jam more than one channel in a multi-channel target hop that is only connected to the rest of the network with a single channel. The same applies for capacity jamming[27]. To overcome this limitation, the attacker needs to connect to the target hop via several disjoint paths.

5.3 Payment flow inference

Probing can be a building block for more advanced attacks, for instance, payment flow inference. Given a series of balance snapshots, the attacker can construct a balance difference graph, in which edges with non-zero value correspond to payments. The attacker can then discover the sender, the receiver, and the amount, as the balances along the route are shifted by the same amount (modulo fees). Note that payments that pass through the same hop distort the picture, therefore, snapshots should be frequent. Prior work [16] has shown that 30-second snapshots allow revealing payments with 66% success rate, assuming relatively low network usage (2000 payments per day). Obtaining a full network snapshot that quickly is challenging: each probe takes a few seconds. A more realistic goal could be to infer payment flows between a given pair of nodes by tracking balances in a few shortest paths between them. LN diameter is 6 hops [34], typical path lengths are 3–6 hops, and the target sub-network may be comprised of around 50 nodes.

[25] The attacker may still distinguish between parallel channels in leaf hops using fee-aware probing (see Section 3.4).

[26] Assuming all channels have the same number of slots. The attacker may have higher limits than the victim, but no channel can have more than 483 slots per direction.

[27] Note that channels might be bottlenecked by slots, but available by capacity.

5.4 Countermeasures

The fact that failed payment attempts are free in the LN makes probing cheap. Proposals to incur upfront fees for all payment attempts are being discussed [14]. Assuming no such changes to the LN protocol, we now discuss countermeasures that individual nodes can apply.

Alternative forwarding strategies A routing node can try to obfuscate the state of its channels if probing is detected (i.e., if it notices a series of failed payments with amounts that follow the binary search pattern). In particular, routing nodes may choose forwarding channels to minimize changing h and g. A heavily used routing node could execute payments in batches. Within one batch, payments can be re-ordered so that they partially or even fully cancel each other out. More generic flow concealment strategies are also possible.

Intra-hop payment split A routing node can potentially divide a payment among parallel channels in the next hop, which may optimize hop bandwidth and hinder probing. This technique is being discussed as part of the future switch to a new type of channel construction [24, 45]. From the prober's viewpoint, a multi-channel hop with intra-hop payment split is equivalent to a single-channel hop. The prober can thus reveal the sum of channel balances. Note the difference compared to multi-part payments (MPP): in MPP, the sender fully determines how to split the payment [6], whereas in intra-hop split, such decisions are made locally by routing nodes.

Rebalancing and JIT routing Channel rebalancing is a process by which an LN node sends (presumably circular) payments to bring the ratio of local to remote balance in its channels closer to some desirable value (i.e., make them equal). Just-in-time (JIT) routing [23] is a form of rebalancing done while forwarding another payment. If a routing node is asked to forward a payment for which it lacks balance, it first moves some funds to the local side of one of its channels using a circular payment, and then proceeds with the forwarding. From a prober's standpoint, rebalancing changes the properties of a hop mid-probe, distorting the estimates. Without intra-hop splitting, a multi-channel hop between Alice and Bob with JIT routing becomes equivalent to a single-channel hop with balances $b_A = \max_{i\in[0,N]} b_{i,A}$ and $b_B = \max_{i\in[0,N]} b_{i,B}$, where $b_{i,A}$ and $b_{i,B}$ are balances of the i-th channel at Alice's and Bob's sides, respectively.

Unannounced channels A node may open unannounced channels parallel to public ones to hide public channel balances. Depending on the relation between the balances of announced and unannounced channels, the attacker may still be able to discover unannounced channel balances (e.g., if the balance of the unannounced channel exceeds the balances of public channels). Even in that case, the standard probing technique needs to be modified.

6 Related work

Attacks on the LN can be grouped into privacy-related [31, 29, 2, 16, 42, 39, 13, 21, 30], DoS-related [33, 10, 38, 19, 40, 22, 29] and incentive-related [41].

Prior work on channel probing introduced the general idea [13], suggested probing channels from both ends [42], controlling both the sender and the receiver of probes [16], and multi-hop probing [39]. Multiple LN simulators have been designed to analyze honest economic activity [2, 43, 5] or the cost of opening payment channels [4, 8]. Rate-limiting has been proposed to mitigate issues like probing and jamming [37, 32, 28, 15]. The fee structure [2] and the tension between privacy and utility of routing nodes [35, 11] have also been discussed. Other relevant prior work focused on channel jamming [38, 19], channel policy exploitation [27], and improved payment forwarding [45].

7 Conclusion

In this work, we have developed a comprehensive model for LN balance probing that accounts for parallel channels. We have introduced enhanced versions of the probing attack, combining it with channel jamming and fee targeting. Enhanced probing overcomes the limit on information gain in multi-channel hops and allows for nearly full balance extraction. Moreover, we have proposed binary-search-based algorithm (NBS) for choosing probe amounts that improves probing speed.

We have confirmed our findings experimentally in an isolated testing environment and using a specially developed probing-focused LN simulator. The simulations based on a real-world network snapshot show that NBS speeds up probing by up to 15% compared to single-dimensional binary search (two-channel hops, direct non-enhanced probing). The experiments also illustrate the trade-off between opening channels to target hops vs probing through multi-hop paths. Finally, we have outlined potential countermeasures and avenues for future work.

The LN promises to significantly improve Bitcoin's scalability and privacy. In order for the LN to realize its potential, it should defend against attacks such as balance probing and channel jamming. We hope that this work helps improve the trade-offs between scalability, security, and privacy for the LN, while preserving its permissionless nature.

Acknowledgments

We thank Antoine Riard for thoughtful feedback. This work was partially supported by the Luxembourg National Research Fund (FNR) project FinCrypt (C17/IS/11684537). Contributions of Gleb Naumenko were supported with a grant by 100x Group, the holding structure for the BitMEX platform.

References

1. Elli Androulaki, Ghassan Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. Evaluating user privacy in Bitcoin. In Ahmad-Reza Sadeghi, editor,

Financial Cryptography and Data Security - 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers, volume 7859 of *Lecture Notes in Computer Science*, pages 34–51. Springer, 2013.
2. Ferenc Béres, István András Seres, and András A. Benczúr. A cryptoeconomic traffic analysis of Bitcoin's Lightning network. *CoRR*, abs/1911.09432, 2019.
3. BOLT. Lightning network specifications. `https://github.com/lightningnetwork/lightning-rfc`, 2019.
4. Simina Brânzei, Erel Segal-Halevi, and Aviv Zohar. How to charge Lightning. `https://arxiv.org/abs/1712.10222`, 2017.
5. Marco Conoscenti, Antonio Vetrò, J. Martin, and Federico Spini. The CLoTH simulator for HTLC payment networks with introductory Lightning network performance results. *Inf.*, 9(9):223, 2018.
6. LND developers. Multi-path payments in lnd: Making channel balances add up. `https://lightning.engineering/posts/2020-05-07-mpp/`, 2020.
7. EmelyanenkoK. Payment channel congestion via spam-attack. `https://github.com/lightningnetwork/lightning-rfc/issues/182`, 2017.
8. Felix Engelmann, Henning Kopp, Frank Kargl, Florian Glaser, and Christof Weinhardt. Towards an economic analysis of routing in payment channel networks. *Proceedings of the 1st Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*, Dec 2017.
9. Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. SoK: Layer-two blockchain protocols. In Joseph Bonneau and Nadia Heninger, editors, *Financial Cryptography and Data Security - 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10-14, 2020*, volume 12059 of *Lecture Notes in Computer Science*, pages 201–226. Springer, 2020.
10. Jona Harris and Aviv Zohar. Flood & loot: A systemic attack on the Lightning network. In *AFT '20: 2nd ACM Conference on Advances in Financial Technologies, New York, NY, USA, October 21-23, 2020*, pages 202–213. ACM, 2020.
11. Tankred Hase and Valentine Wallace. Smarter autopilot. `https://blog.lightning.engineering/announcement/2019/04/23/mainnet-app.html`, Apr 2019.
12. Mike Hearn and Jeremy Spilman. Anti dos for tx replacement. `https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2013-April/002417.html`, 2013.
13. Jordi Herrera-Joancomartí, Guillermo Navarro-Arribas, Alejandro Ranchal Pedrosa, Cristina Pérez-Solà, and Joaquín García-Alfaro. On the difficulty of hiding the balance of Lightning network channels. In Steven D. Galbraith, Giovanni Russello, Willy Susilo, Dieter Gollmann, Engin Kirda, and Zhenkai Liang, editors, *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, AsiaCCS 2019, Auckland, New Zealand, July 09-12, 2019*, pages 602–612. ACM, 2019.
14. Joost Jager. A proposal for up-front payments. `https://lists.linuxfoundation.org/pipermail/lightning-dev/2020-March/002585.html`, 2020.
15. Joost Jager. Circuit breaker. `https://github.com/lightningequipment/circuitbreaker`, 2021.
16. George Kappos, Haaroon Yousaf, Ania M. Piotrowska, Sanket Kanjalkar, Sergi Delgado-Segura, Andrew Miller, and Sarah Meiklejohn. An empirical analysis of privacy in the Lightning network. *CoRR*, abs/2003.12470, 2020.
17. Giulio Malavolta, Pedro Moreno-Sanchez, Clara Schneidewind, Aniket Kate, and Matteo Maffei. Anonymous multi-hop locks for blockchain scalability and interoperability. In *26th Annual Network and Distributed System Security Symposium,*

NDSS 2019, San Diego, California, USA, February 24-27, 2019. The Internet Society, 2019.
18. Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. A fistful of bitcoins: Characterizing payments among men with no names. *login Usenix Mag.*, 38(6), 2013.
19. Ayelet Mizrahi and Aviv Zohar. Congestion attacks in payment channel networks. *CoRR*, abs/2002.06564, 2020.
20. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`, 2008.
21. Utz Nisslmueller, Klaus-Tycho Foerster, Stefan Schmid, and Christian Decker. Toward active and passive confidentiality attacks on cryptocurrency off-chain networks. In Steven Furnell, Paolo Mori, Edgar R. Weippl, and Olivier Camp, editors, *Proceedings of the 6th International Conference on Information Systems Security and Privacy, ICISSP 2020, Valletta, Malta, February 25-27, 2020*, pages 7–14. SCITEPRESS, 2020.
22. Cristina Pérez-Solà, Alejandro Ranchal-Pedrosa, Jordi Herrera-Joancomartí, Guillermo Navarro-Arribas, and Joaquín García-Alfaro. Lockdown: Balance availability attack against Lightning network channels. In Joseph Bonneau and Nadia Heninger, editors, *Financial Cryptography and Data Security - 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10-14, 2020 Revised Selected Papers*, volume 12059 of *Lecture Notes in Computer Science*, pages 245–263. Springer, 2020.
23. René Pickhardt. Just in time routing (JIT-routing) and a channel rebalancing heuristic as an add on for improved routing success in BOLT 1.0. `https://lists.linuxfoundation.org/pipermail/lightning-dev/2019-March/001891.html`, 2019.
24. Andrew Poelstra. Lightning in scriptless scripts. https://lists.launchpad.net/mimblewimble/msg00086.html, Mar 2017.
25. Joseph Poon and Thaddeus Dryja. The Bitcoin Lightning network: Scalable off-chain instant payments. Technical report, 2016.
26. BitMEX Research. Proportion of public vs private channels. `https://blog.bitmex.com/lightning-network-part-7-proportion-of-public-vs-private-channels/`, 2020.
27. Antoine Riard. Route blinding. `https://github.com/lightningnetwork/lightning-rfc/pull/765\#pullrequestreview-511147029`, Oct 2020.
28. Antoine Riard and Gleb Naumenko. Stake certificates. `https://thelab31.xyz/stake-certificates`, 2020.
29. Elias Rohrer, Julian Malliaris, and Florian Tschorsch. Discharged payment channels: Quantifying the Lightning network's resilience to topology-based attacks. In *2019 IEEE European Symposium on Security and Privacy Workshops, EuroS&P Workshops 2019, Stockholm, Sweden, June 17-19, 2019*, pages 347–356. IEEE, 2019.
30. Elias Rohrer and Florian Tschorsch. Counting down thunder: Timing attacks on privacy in payment channel networks. In *AFT '20: 2nd ACM Conference on Advances in Financial Technologies, New York, NY, USA, October 21-23, 2020*, pages 214–227. ACM, 2020.
31. Matteo Romiti, Friedhelm Victor, Pedro Moreno-Sanchez, Bernhard Haslhofer, and Matteo Maffei. Cross-layer deanonymization methods in the Lightning protocol. *CoRR*, abs/2007.00764, 2020.

32. Rusty Russel. A proposal for up-front payments. https://lists.linuxfoundation.org/pipermail/lightning-dev/2019-November/002275.html.
33. Rusty Russel. Loop attack with onion routing.. https://lists.linuxfoundation.org/pipermail/lightning-dev/2015-August/000135.html, Aug 2015.
34. István András Seres, László Gulyás, Dániel A. Nagy, and Péter Burcsi. Topological analysis of Bitcoin's Lightning network. In *MARBLE*, pages 1–12. Springer, 2019.
35. Weizhao Tang, Weina Wang, Giulia C. Fanti, and Sewoong Oh. Privacy-utility tradeoffs in routing cryptocurrency over payment channel networks. In Edmund Yeh, Athina Markopoulou, and Y. C. Tay, editors, *Abstracts of the 2020 SIGMETRICS/Performance Joint International Conference on Measurement and Modeling of Computer Systems, Boston, MA, USA, June, 8-12, 2020*, pages 81–82. ACM, 2020.
36. Bastien Teinturier. Trampoline onion format (feature 24/25). https://github.com/lightningnetwork/lightning-rfc/pull/836.
37. Bastien Teinturier. Spamming the Lightning network. https://github.com/t-bast/lightning-docs/blob/master/spam-prevention.md\#costless-channel-probing, Nov 2020.
38. Sergei Tikhomirov, Pedro Moreno-Sanchez, and Matteo Maffei. A quantitative analysis of security, anonymity and scalability for the Lightning network. In *2020 IEEE European Symposium on Security and Privacy Workshops, EuroS&P Workshops 2020, September 7-11, 2020*. IEEE, 2020.
39. Sergei Tikhomirov, René Pickhardt, Alex Biryukov, and Mariusz Nowostawski. Probing channel balances in the Lightning network. *CoRR*, abs/2004.00333, 2020.
40. Saar Tochner, Stefan Schmid, and Aviv Zohar. Hijacking routes in payment channel networks: A predictability tradeoff. *CoRR*, abs/1909.06890, 2019.
41. Itay Tsabary, Matan Yechieli, and Ittay Eyal. MAD-HTLC: because HTLC is crazy-cheap to attack. *CoRR*, abs/2006.12031, 2020.
42. Gijs van Dam, Rabiah Abdul Kadir, Puteri N. E. Nohuddin, and Halimah Badioze Zaman. Improvements of the balance discovery attack on Lightning network payment channels. In Marko Hölbl, Kai Rannenberg, and Tatjana Welzer, editors, *ICT Systems Security and Privacy Protection - 35th IFIP TC 11 International Conference, SEC 2020, Maribor, Slovenia, September 21-23, 2020, Proceedings*, volume 580 of *IFIP Advances in Information and Communication Technology*, pages 313–323. Springer, 2020.
43. Y. Zhang, D. Yang, and G. Xue. Cheapay: An optimal algorithm for fee minimization in blockchain-based payment channel networks. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pages 1–6, 2019.
44. ZmnSCPxj. Outsourcing route computation with trampoline payments. https://lists.linuxfoundation.org/pipermail/lightning-dev/2019-April/001950.html, 2019.
45. ZmnSCPxj. A payment point feature family. https://lists.linuxfoundation.org/pipermail/lightning-dev/2019-October/002225.html, Oct 2019.

A Derivation of Equation 1

Consider an N-dimensional grid of points with integer coordinates. We can define a rectangle $R(L, U)$ using its lower-left vertex $L = (l_1, \ldots, l_N)$ (closest to the origin) and upper-right vertex $U = (u_1, \ldots, u_N)$ (opposite to L). If $l_i \leq u_i \forall i \in [1, N]$, the area of $R(L, U)$ is:

$$S(R) = \prod_{i=1}^{N}(u_i - l_i + 1) \tag{4}$$

Both L and U belong to $R(L,U)$, hence the $+1$. If $\exists i \in [1,N] : l_i > u_i$, then we define $R(L,U) = \emptyset$, and $S(R(L,U)) = 0$. The intersection of $R_1 = R(L_1,U_1)$ and $R_2 = R(L_2,U_2)$ is a rectangle: $R = R_1 \cap R_2 = R(L_2,U_1)$. We can calculate its area using Equation 4. The area of a difference of rectangles (which is not necessarily a rectangle) is $S(R_1 \setminus R_2) = S(R_1) - S(R_1 \cap R_2)$.

Let us now define $S(F)$ in terms of rectangles. Let $\mathbf{0}$ be the origin (the vector of N zeros). Let us denote $\hat{g}^l = (c_i - g_i^l)$ and $\hat{g}^u = (c_i - g_i^u)$. Each probe corresponds to a rectangle. For probes in *dir0*, the lower-left vertex is $\mathbf{0}$; for probes in *dir1*, the upper vertex is C. The other vertex reflects the probe amount. Each of the four current bounds (h^l, h^u, g^l, g^u) defines a rectangle:

$$R^{\rightarrow l} = R(\mathbf{0}, h^l)$$
$$R^{\rightarrow h} = R(\mathbf{0}, h^u)$$
$$R^{\leftarrow l} = R(\hat{g}^l, C)$$
$$R^{\leftarrow h} = R(\hat{g}^u, C)$$

The upper bounds h^u and g^u imply that B is within the intersection of their corresponding rectangles, which we will call R_{in}:

$$R_{in} = R^{\rightarrow h} \cap R^{\leftarrow h}$$

The lower bounds h^l and g^l imply that B is outside their corresponding rectangles. Hence, we exclude from R_{in} the points that belong to $R^{\rightarrow l}$ and $R^{\leftarrow l}$ (Figure 3 shows an example for $N = 2$ and $c_1 = c_2 = c$):

$$F = R_{in} \setminus (R^{\rightarrow l} \cup R^{\leftarrow l}) \tag{5}$$

The area $S(F)$ can be calculated as:

$$S(F) = S(R_{in}) - S(R_{in} \cap R^{\rightarrow l}) - S(R_{in} \cap R^{\leftarrow l}) + S(R^{\rightarrow l} \cap R^{\leftarrow l}) \tag{6}$$

The last component corresponds to $R^{\rightarrow l} \cap R^{\leftarrow l}$. We must add the area of this intersection to compensate for having subtracted it twice. Note that $R^{\rightarrow l} \cap R^{\leftarrow l} \subseteq R_{in}$, which follows from the definition of lower and upper bounds[28].

Let us now calculate $S(F)$ following Equation 6. By definition, probes with amounts h^l and h^u are issued in *dir0*, and g^l and g^u – in direction *dir1*. Hence we omit the directions: $h_i^l = h_i^{l,dir0}$, $h_i^u = h_i^{u,dir0}$, $g_i^l = g_i^{l,dir1}$, $g_i^u = g_i^{u,dir1}$.

First, consider R_{in}. To calculate $S(R_{in})$ using Equation 4, we need to know where the lower-left and upper-right vertices of R_{in} are. The upper-right vertex

[28] Indeed, $R^{\rightarrow l} \subseteq R^{\rightarrow h}$ and $R^{\leftarrow l} \subseteq R^{\leftarrow h}$, hence $R^{\rightarrow l} \cap R^{\leftarrow l} \subseteq R^{\rightarrow h}$ and $R^{\rightarrow l} \cap R^{\leftarrow l} \subseteq R^{\leftarrow h}$. Therefore, $R^{\rightarrow l} \cap R^{\leftarrow l} \subseteq R_{in}$, and $R_{in} \cap R^{\rightarrow l} \cap R^{\leftarrow l} = R^{\rightarrow l} \cap R^{\leftarrow l}$.

is defined by the upper-bound probe in *dir0*, therefore its i-th coordinate is h_i^u. Let us denote $\overline{x} = x + 1$. The corresponding rectangle $R^{\rightarrow h}$ cuts $h_i^u + 1$ points along the i-th dimension. The lower-left vertex is defined by the upper-bound probe in *dir1*, therefore its i-th coordinate is $\overline{c_i} - g_i^u$. The corresponding rectangle $R^{\leftarrow h}$ cuts $g_i^u + 1$ points along the i-th dimension. Applying Equation 4, we get:

$$S(R_{in}) = \prod_{i=1}^{N} (\overline{h_i^u} + \overline{g_i^u} - \overline{c_i}) \tag{7}$$

This formula has a geometrical interpretation. Each of the two probes – in *dir0* and *dir1* – cuts an interval along the i-th dimension. The former probe cuts $[0, h_i^u]$. This means that b_i can take any of $h_i^u + 1$ values from 0 to h_i^u. The latter probe cuts $[c_i - g_i^u, c_i]$. This means that b_i can take any of $g_i^u + 1$ values from $c_i - g_i^u$ to c_i. Adding up the lengths of the two intervals would "cover" all points in $[0, c_i]$, and the points at the intersection would be covered twice. We can calculate its length as the sum of the two lengths minus the length of the whole interval $[0, c_i]$: $(h_i^u + 1) + (g_i^u + 1) - (c_i + 1) = \overline{h_i^u} + \overline{g_i^u} - \overline{c_i}$.

Now consider the lower bounds (this corresponds to subtracting $R^{\rightarrow l} \cup R^{\leftarrow l}$ in Equation 1). The probe with amount h^l in *dir0* defines $R^{\rightarrow l}$. The intersection $R_{in} \cap R^{\rightarrow l} = R(\overline{c_i} - g_i^u, h_i^l)$ has the area:

$$S(R_{in} \cap R^{\rightarrow l}) = \prod_{i=1}^{N} (\overline{h_i^l} + \overline{g_i^u} - \overline{c_i}) \tag{8}$$

Analogously for $R^{\leftarrow l} = R(\overline{c_i} - g_i^l, h_i^u)$:

$$S(R_{in} \cap R^{\leftarrow l}) = \prod_{i=1}^{N} (\overline{h_i^u} + \overline{g_i^l} - \overline{c_i}) \tag{9}$$

Finally, for $R^{\rightarrow l} \cap R^{\leftarrow l}$:

$$S(R^{\rightarrow l} \cap R^{\leftarrow l}) = \prod_{i=1}^{N} (\overline{h_i^l} + \overline{g_i^l} - \overline{c_i}) \tag{10}$$

Combining equations 6, 7, 8, 9, and 10, we get Equation 1:

$$S(F) = \prod_{i=1}^{N} (\overline{h_i^u} + \overline{g_i^u} - \overline{c_i}) - \prod_{i=1}^{N} (\overline{h_i^l} + \overline{g_i^u} - \overline{c_i}) - \prod_{i=1}^{N} (\overline{h_i^u} + \overline{g_i^l} - \overline{c_i}) + \prod_{i=1}^{N} (\overline{h_i^l} + \overline{g_i^l} - \overline{c_i})$$

To add jamming-enhanced probing to the model, we assume that the attacker can jam and unjam any channel in any hop. The attacker probes a target hop without jamming, and then iterates through channels whose balances are not precisely known, jams all other channels, and probes the only unjammed channel.

We introduce a rectangle $B = R((b_1^l, \ldots, b_N^l), (b_1^h, \ldots, b_N^h))$, where b_i^l and b_i^h are the current balance bounds: $b_i^l < b_i \leq b_i^h$. Similarly to Equation 6, we define:

$$F_B = R_{in,B} \setminus (R_B^{\rightarrow l} \cup R_B^{\leftarrow l}) \tag{11}$$

where $R_{in,B} = R_{in} \cap B$, $R_B^{\rightarrow l} = R^{\rightarrow l} \cap B$, and $R_B^{\leftarrow l} = R^{\leftarrow l} \cap B$. In the pre-jamming phase, individual balance bounds are also updated where possible (in addition to the bounds on h and g). At each step of jamming-enhanced probing, B shrinks in half along the i-th (currently unjammed) dimension. Ultimately, F_B is reduced to a single point.

Effective probe amounts Small probes that are forwarded through a large intermediary channel do not give the attacker any new information. Similarly, probing give no information about channels that are disabled in the probe direction. To account for this in our model, we introduce the notion of the *effective probe amount for channel i and direction dir0* as follows:

$$a_i = \begin{cases} a, \, i \in E \text{ and } a \leq c_i \\ c_i + 1, \text{ otherwise} \end{cases} \tag{12}$$

We also define the effective lower bound h_i^l for h along the dimension i:

$$h_i^l = \begin{cases} a - 1, \, i \in E \text{ and } a \leq c_i \\ c_i, \text{ otherwise} \end{cases} \tag{13}$$

The definitions for h_i^u, g_i^l, g_i^u are analogous. The intuition here is that if a channel is disabled in a given direction, no probe in this direction gains any information about it. To reflect this fact, the length of the rectangle being cut along the i-th dimension is either a or c_i+1, or, equivalently, h_i^l+1. This notation allows for generalized formulas regardless of hop configuration.

B Experimental setup in an isolated network

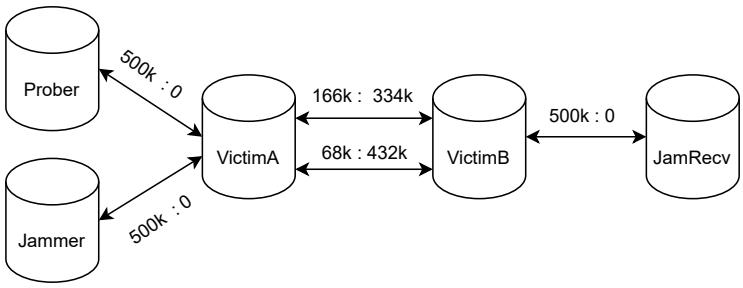

Fig. 10. Experimental setup in an isolated network.

Our setting consisted of five nodes running on different ports on the same machine (Figure 10)[29]. For Prober and Jammer, we used the C-LIGHTNING implementation to build on the previous work on channel probing. We use a probing tool implemented as a plugin for C-LIGHTNING. For JamRecv, used a modified C-LIGHTNING implementation. We changed the source code so that the node waits for 120 seconds after receiving a payment for an unknown invoice, so that the channels on the route from the Jammer are indeed jammed. For VictimA and VictimB, we used ECLAIR (C-LIGHTNING does not support parallel channels). Experiments only involved our own nodes; no LN users were impacted.

For both experiments, we opened two channels between VictimA and VictimB and allocated the balances (in satoshis) as $68k$:$432k$ and $166k$:$334k$ (we write Xk for X thousand). We also opened channels from the attacker's node to VictimA. Non-enhanced probing provided an upper bound of $167k$ satoshis for both target channels and the lower bound $165k$ satoshis for the entire hop.

Jamming-enhanced probing To infer the balance of the smaller channel, the attacker sent a payment of $150k$ satoshis from Jammer to JamRecv, which held it for two minutes. The available balances were $68k$:$432k$ and $16k$:$334k$ (note that the in-flight balance was unavailable for either direction). Since the $67k$ channel had the largest balance in the hop, the attacker could probe it. Probing yielded an estimate of $[66k : 68k]$, which indeed represented the second channel balance. After the probing was done, JamRecv failed the in-flight payment. Thus, we confirmed that channel jamming can indeed improve balance estimated when probing multi-channel hops.

Fee-aware probing For fee-aware probing, only three nodes were relevant: Prober, VictimA, and VictimB. We updated the larger channel from the previous experiment to require non-zero fees. The Prober node was first configured to send probes with sufficient fees for the more expensive channel. By sending such probes, the attacker yielded the same result, inferring the balance of the larger (non-zero-fee) channel. To probe the smaller channel, the attacker configured the Prober node to send only zero-fee probes and successfully inferred the balance of the smaller channel: $[66k : 68k]$.

C Experiments on synthetic hops

We now demonstrate how the hop structure influences information gain. In this experiment, we use synthetic two-channel hops with considerably different capacities (in satoshis): either $c_{big} = 2^{20}$ or $c_{small} = 2^{15}$. We do not consider channels disabled in both directions. We denote a hop configuration as "x-y-c1-c2" if x channels are enabled in *dir0*, and y channels are enabled in *dir1*. If capacities are different, we denote them as c1 and c2 (if they are the same, it

[29] Note that Prober and Jammer can be the same node with two channels for each activity, but we make them distinct for simplicity.

makes no difference whether they equal c_{big} or c_{small}, so we omit this part of the notation). For example, type "2-1" means that two equal-capacity channels are enabled in *dir0*, but only one is enabled in *dir1*. Accounting for symmetry, there are 12 hop configurations (Table 2).

Configuration	**First channel**			**Second channel**		
	Capacity	***dir0***	***dir1***	**Capacity**	***dir0***	***dir1***
2-2	c_{big}	+	+	c_{big}	+	+
2-2-big-small	c_{big}	+	+	c_{small}	+	+
2-2-small-big	c_{small}	+	+	c_{big}	+	+
1-1	c_{big}	+		c_{big}		+
1-1-big-small	c_{big}	+		c_{small}		+
1-1-small-big	c_{small}	+		c_{big}		+
2-1	c_{big}	+	+	c_{big}	+	
2-1-big-small	c_{big}	+	+	c_{small}	+	
2-1-small-big	c_{small}	+	+	c_{big}	+	
2-0	c_{big}	+	+	c_{big}		
2-0-big-small	c_{big}	+	+	c_{small}		
2-0-small-big	c_{small}	+	+	c_{big}		

Table 2. Configurations of two-channel hops (+ means enabled).

We generate synthetic hops for each configuration and measure the information gain and probing speed achieved using BS and NBS probe amount selection methods (Table 3). Hop configurations 2-2 and 1-1 are most vulnerable. The configuration least prone to probing is 2-0 (0.49 information gain). In general, asymmetric hop configurations are less prone to probing compared to hops with equal balances, except for "2-1-small-big". The intuition is that in a "2-1-small-big" hop all probes go through the larger channel, while the smaller one remains "masked" (if it is not the only enabled channel in a given direction). Practically speaking, users should only enable channels in directions they intend to use. If payments in both direction are needed, users should avoid the configuration "2-1-small-big" (i.e., a small channel enabled in both directions and a large channel enabled in one direction).

Configuration	**Information gain**	**Speed (bits / message)**		
		BS	**NBS**	**Advantage**
2-2	0.95	0.97	1.0	0.02
2-2-big-small	0.58	0.52	0.97	0.85
2-2-small-big	0.59	0.53	0.97	0.83
1-1	0.98	1.0	1.0	0.0
1-1-big-small	0.97	1.0	1.0	0.0
1-1-small-big	0.97	1.0	1.0	0.0
2-1	0.75	0.77	0.98	0.28
2-1-big-small	0.58	0.52	0.97	0.87
2-1-small-big	0.96	0.99	1.0	0.01
2-0	0.49	0.99	0.99	0.0
2-0-big-small	0.57	1.0	1.0	0.0
2-0-small-big	0.57	1.0	1.0	0.0

Table 3. Probing results for various configurations of two-channel hops.