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 multi-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 is limited by design [23]. Second-layer (L2) protocols [10] aim to address this issue. The most prominent L2 protocol for Bitcoin[3] is a payment channel network called the Lightning Network (LN) [29]. A payment channel is a trust-minimized two-party protocol for low-latency cryptocurrency payments [13] with minimal interaction with the underlying blockchain. A channel network allows for multi-hop payments between users who do not share a channel.

In contrast to Bitcoin transactions, which are public and provide very limited privacy [2, 21], L2 payments are not globally broadcast. Hence the LN may be

[3] Similar protocols are possible for other cryptocurrencies.

seen as a privacy-enhancing technology. However, attacks on LN privacy have been described, including balance probing. Probing allows for cheaply revealing channel balances by sending fake payments (probes) [7,14,17,44]. It can be used as a building block to spy on payments or to deanonymize users.

The LN allows nodes to share multiple *parallel* channels. Alice, for instance, may want to open a new channel to Bob if all funds in their existing channel are on Bob's side, preventing her from sending further payments. That would allow Alice to send without losing the ability to receive through the older channel. Earlier probing algorithms assume at most one channel between each pair of nodes and may give false or incomplete results if parallel channels are present.[4]

Our Contributions After providing the necessary background (Section 2), we introduce the probing model (Section 3) and propose an optimized amount selection method to maximize probing speed. We enhance the probing attack by combining it with jamming or fee targeting. Using simulations[5] based on a real-world data, we show that enhanced probing extracts full balance information in parallel channels, which was impossible with prior methods (Section 4). Moreover, optimized amount selection increases probing speed by up to 15%, compared to single-dimensional binary search. In Section 5, we discuss the limitations of our approach, attack cost and trade-offs, payment flow discovery, and countermeasures. 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 address, establishing the initial channel state. To make a payment, the parties negotiate a new state, thereby provably invalidating the old one [10]. Any party can close the channel and withdraw their coins on-chain at any time. A penalty mechanism ensures security of channel state updates. If one party tries to cheat by closing the channel with an outdated state (claiming more funds than the latest state prescribes), the other party is granted a time window to withdraw all funds from the channel.

The total number of coins in a channel, constant throughout its lifetime, is called *capacity* (Figure 1). The number of coins owned by each party is called its *balance* and changes as payments are made. We refer to a pair of adjacent nodes along with all (parallel) channels that they share as a *hop*. Parallel channels within one hop may have different fees and routing policies [4]. A node may disable a channel direction (e.g., before an expected loss of connectivity or channel settlement), making the channel *unidirectional*.[6]

[4] The paper [25] writes on probing in the presence of multiple channels between the same nodes: "Our tool failed to produce accurate results in this scenario [...] further research on how to deal with this complication would be highly appreciated."

[5] The code is at https://github.com/s-tikhomirov/ln-probing-simulator.

[6] Not to be confused with an earlier unidirectional channel construction [13].

Fig. 1. A channel with capacity 5 and balances 3 and 2 for Alice and Bob, respectively

LN nodes and channels are identified by persistent IDs. Node IDs are random; channel IDs are derived from the parameters of the respective opening transactions. Nodes can (but do not have to) announce the availability, capacities, and policies of their channels in the P2P network.[7]

An LN user can send *multi-hop payments* without establishing a channel with the receiver. To initiate a payment, the receiver generates a *payment secret* and sends its hash (the *payment hash*) to the sender. The sender routes the payment along the *payment path* (an ordered list of *routing*[8] *nodes* chosen based on the sender's local view of the network). Routing nodes usually charge fees by forwarding a bit less than they receive. If an intermediary hop contains parallel channels, a routing node may use any of them (*non-strict forwarding*). Upon receiving a payment, the receiver propagates the payment secret along the path back to the sender. This ensures atomicity of balance shifts along the path as they all depend on the same secret being revealed.[9]

LN nodes are only aware of payments that they send, receive, or forward. Due to onion routing, intermediary nodes only know the previous and the next node in the path, but not the ultimate sender or receiver. Intermediaries do, however, learn the amounts of payments that they forward.

The forwarding ability of a channel is determined by its *balance* in the direction of the payment. However, the sender only knows the *capacities* of announced channels.[10] Therefore, multi-hop payments 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 attempts using different paths until the payment succeeds.

The three major LN implementations (LND, CORE-LIGHTNING, and ECLAIR) use different channel selection strategies for multi-channel hops.[11] ECLAIR selects the channel with the lowest capacity (among the channels with the same capacity, it prefers the one with a lower balance).[12] LND chooses a random channel.[13] CORE-LIGHTNING does not support parallel channels.

[7] A 2020 study estimated that 28.7% of LN channels were unannounced [30].
[8] Routing nodes may also be referred to as *forwarding* or *intermediary* nodes. Alternative approaches are trampoline [41] and rendezvous routing [48].
[9] It may be argued though that the wormhole attack [20] violates atomicity.
[10] Obviously, nodes also know the balances of their own channels, even if unannounced.
[11] Path selection algorithms also differ [19].
[12] https://github.com/ACINQ/eclair/blob/5f9d0d/eclair-core/src/main/scala/fr/acinq/eclair/payment/relay/ChannelRelay.scala#L199
[13] https://github.com/lightningnetwork/lnd/blob/f98a3c/htlcswitch/switch.go#L1091

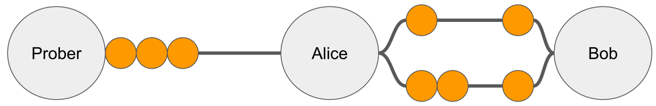

Fig. 2. A probing setup for a two-channel target hop: the attacker does not know which channel the probes go through.

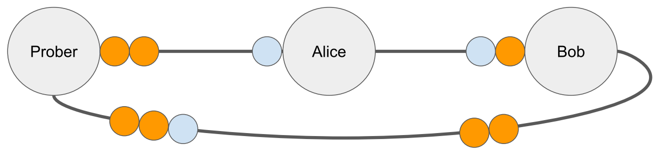

Fig. 3. Jamming attack: a jam (light-colored circle) is blocking other potential payments through the channel from Alice to Bob.

Attacks on Lightning

For our work, the most relevant attacks on the LN 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 [14, 17, 44]. A probe is a payment with amount a that contains a random value instead of a payment hash. A probe fails either at an intermediary node due to insufficient balance, or at the receiver because of the unknown hash preimage.[14] The location of the erring node within the path reveals whether the balance of the erring channel is above or below a. We say that a probe that reaches[15] the target hop *succeeds* if it goes through or that it *fails* if it does not. By sending probes with different amounts, the attacker can infer the balance in the target channel with high accuracy. Assuming uniform balance distribution, the best strategy for choosing probe amounts is binary search. If the target hop contains parallel channels, probing may provide incorrect results (Figure 2).

Jamming is a family of denial-of-service attacks on LN channels [8, 42]. An attacker initiates a payment (a *jam*) along a circular[16] path, which includes the target channel, and refuses to reveal the payment secret, locking up the funds along the path (Figure 3). 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 [26]. In *slot-based jamming*, an attacker sends a series of small payments (each above a certain dust limit) to reach the limit of *payment slots* for in-flight payments (at most 483 in each direction; channel parties may set lower limits) [43]. Onion routing complicates protection against jamming: the victim does not know who is sending the jams.

[14] We do not consider other potential errors for simplicity.
[15] When probing via multi-hop paths, probes may fail before reaching the target hop.
[16] Alternatively, the path may terminate at a different node controlled by the attacker.

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.[17] 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.[18] The attacker can run modified software but has no control over other users' software.

We define channel direction as follows: *dir0* is the direction from the node with the alphanumerically smaller ID to the other node; *dir1* is the opposite direction. We define channel balance (in satoshis[19]) as the balance of the node with the alphanumerically smaller ID. Note that the *dir0* / *dir1* notation depends neither on the payment direction nor on who opened the channel.

A hop with N channels is defined 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*:

$$h = \max_{i \in E^{dir0}} b_i \tag{1}$$

$$g = \max_{i \in E^{dir1}} (c_i - b_i) \tag{2}$$

In the general case, probes only give the attacker information about h or g, not about individual balances.[20] The attacker maintains the current lower and upper bounds[21] for h and g: $h^l < h \leq h^u$ and $g^l < g \leq g^u$, initially set to:

$$h^l = g^l = -1 \tag{3}$$

$$h^u = \max_{i \in E^{dir0}} c_i \tag{4}$$

$$g^u = \max_{i \in E^{dir1}} c_i \tag{5}$$

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.

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

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

[19] 1 satoshi equals 10^{-8} BTC and is the smallest sub-unit of bitcoin. The LN operates with millisatoshi precision off-chain, but such amounts cannot be settled on-chain precisely. For simplicity, our model operates with satoshi-level precision.

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

[21] 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, and analogously for g and *dir1*. Our definitions reflect this asymmetry and thus allow for uniform calculations when deriving Equation (8).

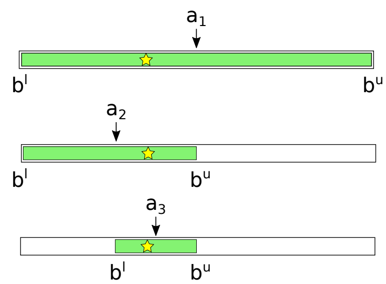

Fig. 4. Probing a one-channel hop with simple binary search. The star denotes the true balance. The colored rectangle represents the attacker's current estimates.

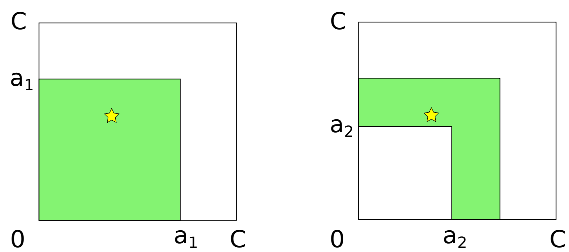

Fig. 5. A geometrical model for the first two probes of a two-channel target hop. The first probe (left) fails (upper bound); the second probe (right) succeeds (lower bound).

3.1 Examples

As the simplest example, consider a hop containing a single channel with capacity c (Figure 4). Let b^l and b^u be the current lower (strict) and upper (non-strict) bounds for the true balance b, respectively.[22] Initially, $b^l = -1$ and $b^u = c$. $F = (b^l, b^u]$. For each next i^{th} probe, the attacker chooses the amount as:

$$a_i = (b^l + b^u + 1)/2 \tag{6}$$

If the probe fails, b^u is updated to $a_i - 1$, otherwise b^l is updated to $a_i - 1$.

Next, consider a two-channel hop with equal capacities $c_1 = c_2 = c$ (Figure 5). Initially, $S(F) = (c+1)^2$. The first probe amount should be:

$$a_1 = (c+1)/\sqrt{2} \tag{7}$$

Note that $a_1 = (c+1)/2$ would divide $S(F)$ in the proportion $3:1$, not $1:1$.

The probe failing indicates that the balance is within a smaller area (the colored square in Figure 5, left). The second probe divides that area in half (Figure 5, right), and so on.

[22] The definition is asymmetric to maintain uniformity with the generalized model introduced later in Section 3.2.

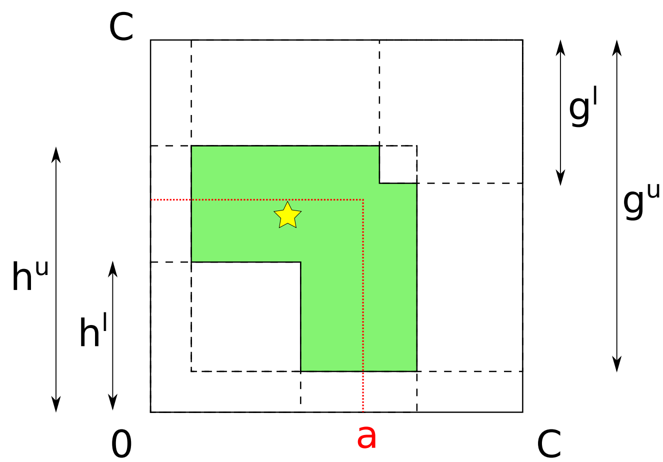

Fig. 6. A geometrical model for probing a two-channel hop

3.2 Generalized Geometrical Model

We can think of an N-channel hop as an N-dimensional (hyper-)rectangle R, with sides parallel to the axes.[23] Each side corresponds to one channel (some channels may be unidirectional). Along the i^{th} dimension, R is defined by the coordinates $[0, c_i]$. The coordinates of each point within R correspond to a possible balance vector. One of the vertices of R is the origin point $(0, \ldots, 0)$.

A probe with amount a "cuts" an a-sided square either from the origin point (for *dir0*) or from the opposite vertex (for *dir1*). If the probe fails, all coordinates of B are lower than a (a new upper bound), otherwise at least one coordinate of B is greater than or equal to a (a new lower bound). If both directions have at least one channel enabled, the attacker may choose any direction for the probe.

Figure 6 illustrates a two-dimensional case with $c_1 = c_2 = c$. The attacker currently knows that the balance cannot be within the two smaller squares with sides h^l and g^l because the corresponding probes have succeeded. At the same time, the balance must be within the two larger squares with sides h^u and g^u because the corresponding probes have failed.

We can define F (colored) as the intersection of two L-shaped figures, reflecting the current bounds on h and g. F may take different shapes, depending on how the bounds relate to each other and to the hop configuration. The attacker chooses the next probe amount a to cut F in half.

Consider an illustrative probing of a two-channel hop with both channels enabled in both directions (Figure 7). Note that in the final stages of probing F consists of two disjoint diagonally symmetric rectangles, reflecting the fact that channel balances can only be revealed up to permutation.

---

[23] We continue using 2D-terms such as "rectangle" and "area" for clarity.

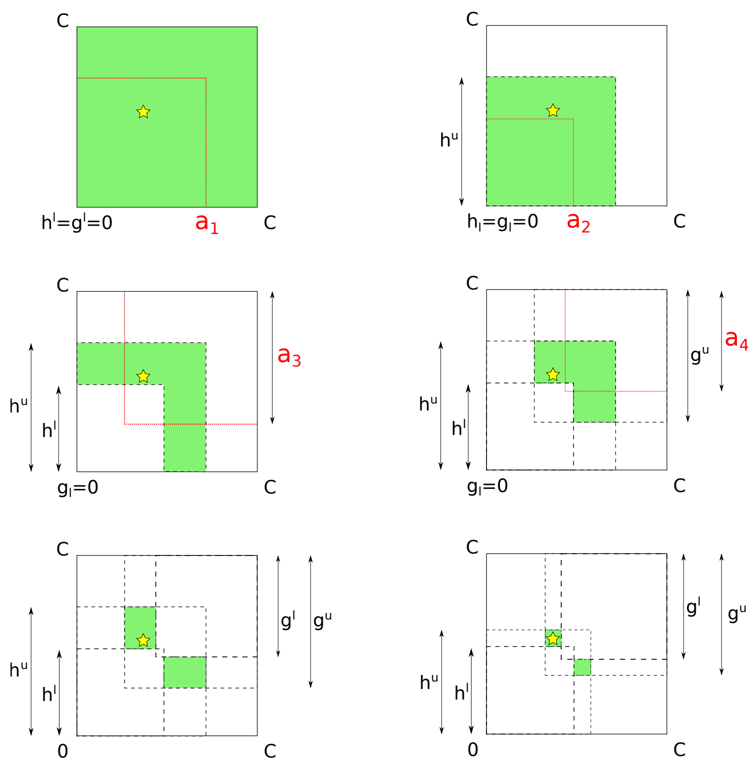

Fig. 7. Probing a two-channel hop step by step. Probing steps omitted between the bottom-left and the bottom-right (final) figures.

Let us denote $\overline{x} = x + 1$ and use subscript i for the i^{th} coordinate. In the general case, we calculate $S(F)$ as follows. For full derivation, see Appendix A.

$$S(F) = \prod_{i=1}^{N}(\overline{h_i^u}+\overline{g_i^u}-\overline{c_i})-\prod_{i=1}^{N}(\overline{h_i^l}+\overline{g_i^u}-\overline{c_i})-\prod_{i=1}^{N}(\overline{h_i^u}+\overline{g_i^l}-\overline{c_i})+\prod_{i=1}^{N}(\overline{h_i^l}+\overline{g_i^l}-\overline{c_i}) \quad (8)$$

In prior probing algorithms, each next amount a was chosen as the midpoint between the current bounds (*single-dimensional binary search*), which is suboptimal in the multi-dimensional case (Section 3.1). Instead, we propose an optimized amount choice algorithm to cut F in half. It works as follows. Initially, set $a^l = h^l + 1$, $a^u = h^u$, and consider a candidate value $a = (a^l + a^u)/2$. Let S_a be the area under the potential cut. If $S_a < S/2$, set $a^l = a$, else set $a^u = a$. Repeat until S_a is as close as possible[24] to $S/2$. For $N = 1$, the two methods are equivalent.

[24] It is usually impossible to cut F in half precisely: increasing a by 1 satoshi adds multiple points to $S(F)$ in multi-channel hops (depending on hop configuration).

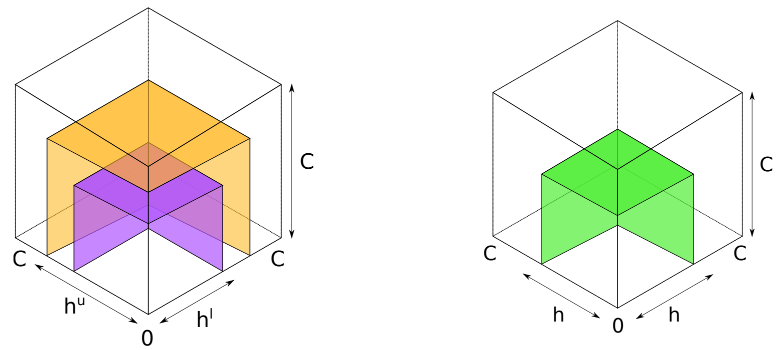

Fig. 8. Probing a 3-channel hop from direction *dir0*: in progress (left), finished (right)

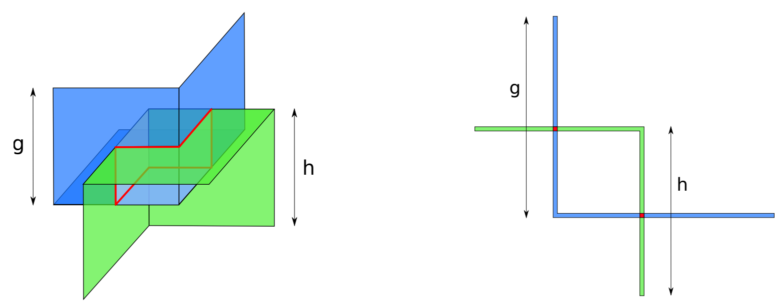

Fig. 9. The final result of probing a 3-channel hop (left) and a 2-channel hop (right). Exact balances in the 3-channel hop are unknown even after fully revealing h and g.

3.3 Challenge of Probing Multi-channel Hops

Hops with three channels or more cannot be fully probed due to dimensionality. Consider a three-channel hop with equal-capacity channels. Each probe in *dir0* cuts an a-sided cube from the corner of the larger C-sided cube. Bounds on h are represented by two surfaces, each composed of three faces of the respective cube (Figure 8, left). The smaller surface represents h^l, and the larger surface represents h^u. Each probe brings the two surfaces closer until they collapse into one surface representing the true value of h (Figure 8, right). Analogously, probes in *dir1* cut cubes from the opposite corner of the large cube.

Consider the final state of the attack when h and g have been fully revealed (Figure 9, left). The true balance point lies at the intersection of two surfaces, each composed of three perpendicular squares. In the general case, this intersection is composed of six intervals and cannot be shrunk to single points. In contrast, in a 2-channel hop, exact balances are revealed (up to permutation) as an intersection of two L-shapes, i.e., two points (Figure 9, right).

Another reason why fully probing multi-channel hops may be impossible is a vast difference in channel capacities, which allows larger channels to "mask" smaller ones. See Appendix B for details.

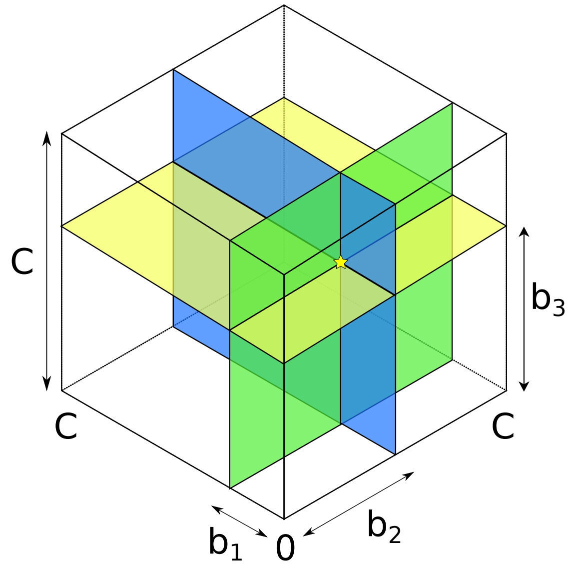

Fig. 10. A geometrical representation of jamming-enhanced probing for a 3-channel hop with equal capacities. The three balances are revealed separately.

3.4 Enhanced Probing

The only way for the attacker to gain more balance information for multi-channel hops would be to force probes to go through specific channels. The attacker cannot affect the channel choice strategy of a routing node. However, it is possible to reduce the set of *suitable* channels the routing node picks from.

We consider two probing enhancement techniques to achieve this goal. In *jamming-enhanced probing*, the attacker jams all channels in a target hop except one, and then probes the remaining channel. In geometrical terms, this allows for making cuts parallel to the axes, which ultimately leads to revealing the exact balance point as the intersection of three perpendicular planes (Figure 10).

In *fee-aware probing* [31], the attacker 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 used an isolated testing environment based on real LN implementations to confirm that enhanced probing indeed allows an attacker to infer individual balances of parallel channels. Setup details are provided in Appendix C.

4 Evaluation

4.1 Data Source

We captured an LN snapshot on 2021-12-09 using our own CORE-LIGHTNING node. The snapshot contains 17068 nodes and 78076 channels[25] with a total capacity of 3370 BTC.[26] This is in line with public explorers such as the one operated by ACINQ[27] (the developers of ECLAIR), which on the same day reported 16977 nodes and 77906 channels. 63697 channels (82%) 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.

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

Channels in a hop	Share of hops (%)	Share of capacity (%)
1	95.4	77.6
2	4.2	10.7
3	0.3	2.7
4	0.1	2.0
≥ 5	0.02	0.3

4.2 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. It is calculated as $\log_2(S(F_i))$, where F_i is the set of all possible balance points after the i^{th} probe. After P probes, U decreases from $U_{before} = \log_2(S(F_0))$ to $U_{after} = \log_2(S(F_P))$. For a set T of target hops, the final achieved information gain is:

$$I = 1 - \sum_{t \in T} U^t_{after} \Big/ \sum_{t \in T} U^t_{before} \tag{9}$$

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

$$S = \frac{1}{m} \left(\sum_{t \in T} U^t_{before} - \sum_{t \in T} U^t_{after} \right) \tag{10}$$

Messages include probes and jams (for jamming-enhanced probing).

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

[26] For an earlier version of this paper, we used a snapshot taken on 2021-09-09. Within three months between the snapshots, the number of nodes increased by 25%, the number of channels by 19%, and the total capacity by 35%.

[27] https://explorer.acinq.co/

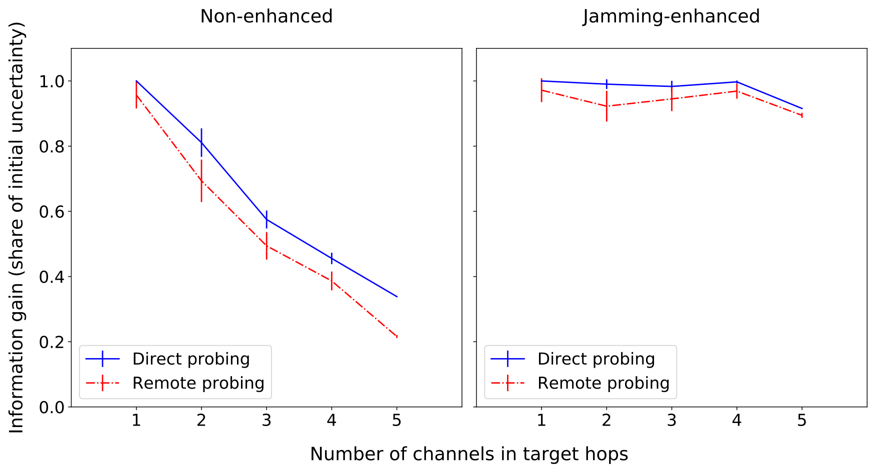

Fig. 11. Achieved information gain for non-enhanced and jamming-enhanced probing

4.3 Results

For each channel in the snapshot, we generate a balance uniformly at random between 0 and the channel capacity. We simulate probing attacks on target hops with 1 to 5 channels (hops with more channels are rare in the snapshot). We model 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. Direct probing is efficient (all probes reach the target) but requires paying on-chain fees for opening channels to each target hop. Moreover, it requires the victim to accept channel opening (though public nodes usually do so if the initiator fully funds it).

In *remote probing*, the attacker opens channels to a few well-connected nodes and sends probes through multi-hop paths. This approach allows for amortizing the on-chain cost of channel openings over multiple target hops. Another benefit is that remote probing yields information about intermediary hops in addition to the target hop. The main drawback of remote probing is that some probes do not reach the target hop due to low balance in an intermediary channel (this effect is more pronounced for larger amounts), although the attacker can decrease the number of such probes by using balance information from earlier probes.

We measure information gain and probing speed for two probing methods (non-enhanced and jamming-enhanced), two probe amount selection methods (optimized and non-optimized), and two types of probing (direct and remote). For each parameter combination, we average the results across 100 simulations. For each simulation, we probe 20 target hops chosen at random.

Information gain decreases as N increases (Figure 11) for non-enhanced probing. This is expected due to the dimensionality issue (Section 3.4). For example,

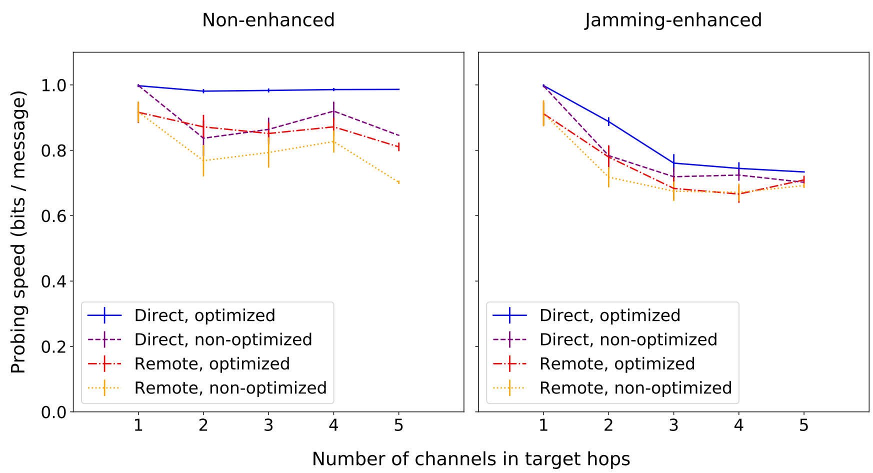

Fig. 12. Probing speed for non-enhanced and jamming-enhanced probing

5-channel hops can only be probed to around 0.4 information gain. This applies to both direct and remote probing. In contrast, jamming-enhanced probing achieves high information gain (above 0.9) for all values of N, illustrating the advantage of such technique. A slight drop for $N = 5$ is caused by one atypical 5-channel hop in the snapshot that has most of its channels disabled. Lower information gain for remote probing compared to direct probing is explained by routing issues.

In terms of probing speed, the optimized amount selection method consistently outperforms the non-optimized method for all values of $N \geq 2$ (Figure 12). (Information gain is the same for the two amount selection methods. The optimized method only allows for getting the same information faster rather than getting more information.) The speedup mostly decreases with increasing N, which is explained by the fact that the optimized method generally chooses higher amounts (for example, $1/2$ vs $1/\sqrt{2}$ in a two-channel hop with $c_1 = c_2 = 1$), which are more likely to fail. Direct probing is always faster than remote probing because all probes reach the target hop. Jamming-enhanced probing lowers the probing speed compared to non-enhanced probing as it implies sending jams in addition to probes. Finally, we note that the optimized method performs better than or similarly to the non-optimized one for all N in both direct and remote probing.

Additional simulations show how the capacity ratio in two-channel hops affects information gain (see Appendix D).

5 Discussion

The simulations have demonstrated that jamming-enhanced probing achieves nearly full balance information extraction, which is otherwise impossible for multi-channel hops in the general case. Moreover, optimized amount selection increases probing speed. The highest speedup is achieved for two-channel hops, which are the most prevalent multi-channel hops in the network.

5.1 Limitations

Our model does not provide theoretical guarantees on the performance of the attack. Simulation-based estimations may serve as rough upper bounds as they assume that remote nodes with sufficient balance always forward payments. In real-world scenarios, the result would depend on network topology, attacker's connectivity, routing policies of other nodes, and other factors.

Our model ignores regular LN activity. If a target hop is heavily used, balances may shift between probes, outdating attacker's estimations. This is one of the reasons why speeding up the attack is important for the attacker: 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 routing fees.

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, routing nodes are free to choose which channel to forward the jam through in multi-channel hops (just like with regular payments). As a result, the attacker only knows how many channels are jammed but does not know which ones. Moreover, even if the attacker reveals N channel balances precisely, 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.[28] Finally, channels disabled in both directions cannot be probed, even with jamming.

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 routing fees, because probes 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-up capital. We leave the evaluation of this trade-off for future work.

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

Jamming-enhanced probing brings additional costs. Capacity-based jamming requires at least one high-capacity channel. The amount of funds locked should be 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 because the attacker's path is limited by the same number of slots.[29]

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. Similar limitations apply for capacity jamming.[30] To overcome this issue, 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, such as payment flow inference. Given a series of balance snapshots, the attacker can construct a balance difference graph where edges with non-zero value correspond to payments. The attacker can then discover the sender, the receiver, and the amount, as balances along the path are shifted by the same amount (modulo fees). Snapshots should be frequent because payments that pass through the same hop distort the picture. Prior work [17] has shown that 30-second snapshots allow revealing payments with 66% success rate, assuming low network usage (2000 payments per day). Obtaining a full network snapshot so quickly is challenging: each probe takes a few seconds. A more realistic goal could be to infer payment flows between given nodes by tracking balances in a few shortest paths between them. This looks feasible: the LN diameter is 6 hops [39], typical path lengths are 3–6 hops, and the target sub-network may be comprised of around 50 nodes.

5.4 Countermeasures

Probing is cheap because failed payment attempts are free. Proposals to limit the number of payment attempts a node can make, e.g., by demanding fees upfront, are being discussed [15, 24]. 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 (e.g., if it notices a series of failed payments with amounts that follow the binary search pattern). In particular, routing nodes may select channels in a way that minimizes changes to h and g. A heavily used routing node could execute payments in batches. Within one batch, payments can be re-ordered so that they cancel each other out, at least partially. More generic flow concealment strategies are also possible.

[29] 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.

[30] Note that channels with sufficient capacity might be limited by slots.

Intra-hop Payment Split A routing node can potentially divide a payment among parallel channels toward the next hop, optimizing hop bandwidth and hindering probing. This technique is being discussed as part of the future switch to a new type of channel construction [28, 49]. From the prober's viewpoint, a multi-channel hop with intra-hop payment split is equivalent to a single-channel hop. The prober can reveal the sum of channel balances. Note the difference compared to multi-path[31] payments (MPP): in MPP, the sender fully determines how to split the payment [1], whereas in intra-hop split, such decisions are made locally by routing nodes.

Channel Rebalancing Channel rebalancing [3,18] is a process by which an LN node initiates (presumably circular) payments to bring the ratio of its channel balance to channel capacity closer to some desirable value (e.g., 50%). Just-in-time (JIT) routing [27] is a form of rebalancing done while forwarding another payment. If a routing node is asked to forward a payment for which all its channels lack 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 equal to

$$\min\left(\sum_{i \in E^{dir0}} b_i, \max_{i \in E^{dir0} \cap E^{dir1}} c_i\right) \tag{11}$$

on the Alice's side and

$$\min\left(\sum_{i \in E^{dir1}} (c_i - b_i), \max_{i \in E^{dir0} \cap E^{dir1}} c_i\right) \tag{12}$$

on the Bob's side. Indeed, ignoring network topology, Alice can concentrate all her local balances in one channel, if the total does not exceed the capacity of the largest bidirectional channel. Note that for JIT routing to work, at least one channel must be enabled in both directions (i.e., $E^{dir0} \cap E^{dir1} \neq \emptyset$).

Unannounced Channels To hide public channel balances, a node may open unannounced channels in parallel to announced ones. 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 announced channels). Even in that case, the standard probing technique needs to be modified.

[31] Also referred to as multi-part payments.

6 Related Work

Attacks on the LN can be grouped into DoS-related [11, 22, 26, 33, 34, 38, 43, 45], privacy-related [5, 7, 14, 17, 25, 35, 36, 44], and incentive-related [46].

Prior work on channel probing introduced the general idea [14], suggested probing channels from both ends [7], controlling both the sender and the receiver of probes [17], and multi-hop probing [44]. Multiple LN simulators have been designed to analyze honest economic activity [5, 6, 47] or the cost of opening payment channels [9]. Rate-limiting has been proposed to mitigate issues like probing and jamming [16, 24, 32, 42]. The fee structure [5, 37] and the tension between privacy and utility of routing nodes [12, 40] have also been discussed. Other relevant prior work focused on channel jamming [22, 43], channel policy exploitation [31], and improved payment forwarding [49].

7 Conclusion

In this work, we have developed a comprehensive model for channel balance probing in the Lightning Network. Our model is the first one to account for parallel channels. We have introduced enhanced versions of the probing attack, combining it with channel jamming and fee targeting. Enhanced probing allows for nearly full extraction of balance information in multi-channel hops, which was impossible with prior methods. Moreover, we have proposed an optimized amount selection algorithm based on N-dimensional binary search that increases probing speed.

We have confirmed our findings experimentally in an isolated testing environment and using a new probing-focused Lightning simulator. The simulations based on a real-world network snapshot show that the optimized amount selection algorithm makes probing up to 15% faster compared to single-dimensional binary search (two-channel target hops, direct non-enhanced probing). The experiments also illustrate the trade-off between direct and multi-hop probing. Finally, we have outlined potential countermeasures and avenues for future work.

The Lightning Network promises to significantly improve Bitcoin's scalability and privacy. To fully realize its potential, Lightning 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 Lightning, 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. Contributions of Sergei Tikhomirov were partially supported by Chaincode Labs.

A Derivation of Equation (8)

Consider an N-dimensional grid with integer coordinates. Let $\mathbf{0}$ be the origin point. We define a rectangle $R(L, U)$ with its lower-left (closest to $\mathbf{0}$) and upper-right vertices $L = (l_1, \dots, l_N)$ and $U = (u_1, \dots, u_N)$. If $\exists i \in [1, N] : l_i > u_i$, then $R(L, U) = \emptyset$ and $S(R(L, U)) = 0$. Otherwise, the area of $R(L, U)$ is:

$$S(R) = \prod_{i=1}^{N} (u_i - l_i + 1) \tag{13}$$

Both L and U belong to $R(L, U)$, hence the $+1$. The intersection of rectangles is a rectangle: $R_1 \cap R_2 = R(L_2, U_1)$. We can calculate its area using Equation (13). The area of a difference of rectangles (not necessarily a rectangle) is:

$$(R_1 \setminus R_2) = S(R_1) - S(R_1 \cap R_2) \tag{14}$$

Let us now define $S(F)$ in terms of rectangles. 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 *dir0*, the lower-left vertex is $\mathbf{0}$; for *dir1*, the upper-right vertex is C. The opposite vertex reflects the probe amount. Each of the four bounds (h^l, h^u, g^l, g^u) defines a rectangle:

$$R^{\to l} = R(\mathbf{0}, h^l) \tag{15}$$

$$R^{\to h} = R(\mathbf{0}, h^u) \tag{16}$$

$$R^{\leftarrow l} = R(\hat{g}^l, C) \tag{17}$$

$$R^{\leftarrow h} = R(\hat{g}^u, C) \tag{18}$$

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}:

$$R_{in} = R^{\to h} \cap R^{\leftarrow h} \tag{19}$$

The lower bounds h^l and g^l imply that B is outside their corresponding rectangles. Hence, we exclude[32] from R_{in} the points that belong to $R^{\to l}$ and $R^{\leftarrow l}$:

$$F = R_{in} \setminus (R^{\to l} \cup R^{\leftarrow l}) \tag{20}$$

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

$$S(F) = S(R_{in}) - S(R_{in} \cap R^{\to l}) - S(R_{in} \cap R^{\leftarrow l}) + S(R^{\to l} \cap R^{\leftarrow l}) \tag{21}$$

[32] Figure 6 shows an example for $N = 2$ and $c_1 = c_2 = c$.

We must add the last component $(R^{\to l} \cap R^{\leftarrow l})$ to compensate for having subtracted it twice. Note that $R^{\to l} \cap R^{\leftarrow l} \subseteq R_{in}$, which follows from the definition of lower and upper bounds.[33]

We define the *effective probe amount for channel i and direction* dir0 as:

$$a_i = \begin{cases} a,\ i \in E \text{ and } a \leq c_i \\ c_i + 1, \text{ otherwise} \end{cases} \tag{22}$$

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

$$h_i^l = \begin{cases} a - 1,\ i \in E \text{ and } a \leq c_i \\ c_i, \text{ otherwise} \end{cases} \tag{23}$$

The rationale here is that a probe can only give information about channels that are enabled in the direction of the probe and whose capacity is higher than the probe amount (which is not always the case for intermediary hops in remote probing). To reflect this fact, the length of the rectangle being "cut" along the i^{th} dimension is either a or $c_i + 1$, which can be written uniformly as $h_i^l + 1$. The definitions for h_i^u, g_i^l, g_i^u are analogous. This notation allows for generalized formulas for all hop configurations.

By definition, probes with amounts h^l and h^u are issued in direction *dir0*, and g^l and g^u are issued in direction *dir1*. Hence our notation omits the directions: $h_i^l = h_i^{l,dir0}$, $h_i^u = h_i^{u,dir0}$, $g_i^l = g_i^{l,dir1}$, $g_i^u = g_i^{u,dir1}$.

Let us now calculate $S(F)$ following Equation (21). First, consider R_{in}. To calculate $S(R_{in})$ using Equation (13), we need to know where the lower-left and the upper-right vertices of R_{in} are. The upper-right vertex 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 $R^{\to h}$ 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 (13), we get:

$$S(R_{in}) = \prod_{i=1}^{N} (\overline{h_i^u} + \overline{g_i^u} - \overline{c_i}) \tag{24}$$

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 $[0, c_i]$: $(h_i^u + 1) + (g_i^u + 1) - (c_i + 1) = \overline{h_i^u} + \overline{g_i^u} - \overline{c_i}$.

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

Now consider the lower bounds (this corresponds to subtracting $R^{\rightarrow l} \cup R^{\leftarrow l}$ in Equation (8)). 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:

$$S(R_{in} \cap R^{\rightarrow l}) = \prod_{i=1}^{N} (\overline{h_i^l} + \overline{g_i^u} - \overline{c_i}) \tag{25}$$

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

$$S(R_{in} \cap R^{\leftarrow l}) = \prod_{i=1}^{N} (\overline{h_i^u} + \overline{g_i^l} - \overline{c_i}) \tag{26}$$

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

$$S(R^{\rightarrow l} \cap R^{\leftarrow l}) = \prod_{i=1}^{N} (\overline{h_i^l} + \overline{g_i^l} - \overline{c_i}) \tag{27}$$

Combining eqs. (21) and (24) to (27), we get Equation (8):

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

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 eq. (21), we define:

$$F_B = R_{in,B} \setminus (R_B^{\rightarrow l} \cup R_B^{\leftarrow l}) \tag{28}$$

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.

B Probing Hops with Vastly Different Capacities

If channel capacities in a hop differ significantly, larger channels can "mask" smaller ones by forwarding all probes in both directions. This scenario is illustrated in Figure 13: no probe can cut F (the colored figure) horizontally.

Jamming-enhanced or policy-aware probing helps overcome this challenge by allowing the attacker to probe channel balances separately (Figure 14), analogously to how it solves the high dimensionality issue discussed in Section 3.3.

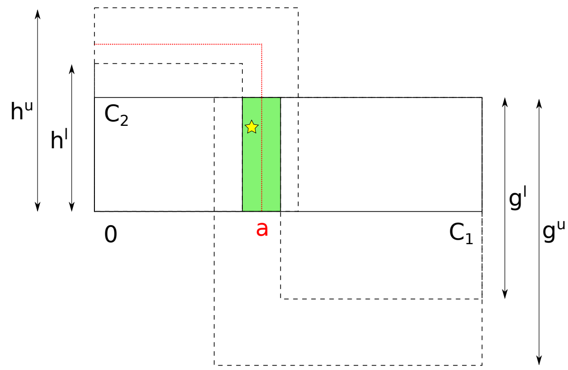

Fig. 13. A 2-channel hop that cannot be fully probed due to vastly different capacities

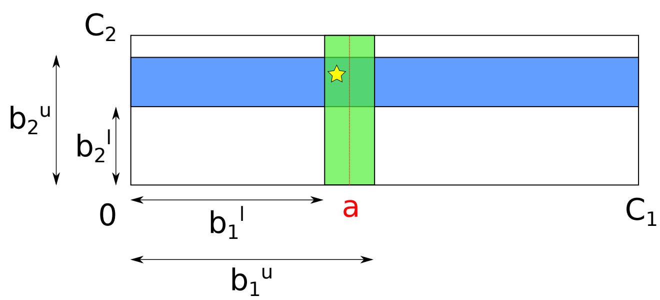

Fig. 14. Jamming-enhanced probing allows for fully probing a “long” two-channel hop.

C Experimental Setup in an Isolated Network

Our setting consisted of five nodes[34] running on different ports on the same machine (Figure 15). For Prober and Jammer, we used CORE-LIGHTNING with a probing tool implemented as a plugin. For JamRecv, we modified the CORE-LIGHTNING implementation so that the node would wait for 120 seconds after receiving a payment for an unknown invoice. For VictimA and VictimB, we used ECLAIR (CORE-LIGHTNING does not support parallel channels). Experiments only involved our own nodes; no LN users were affected.

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

[34] Prober and Jammer can be the same node with a separate channel for each activity, but we make them distinct for simplicity.

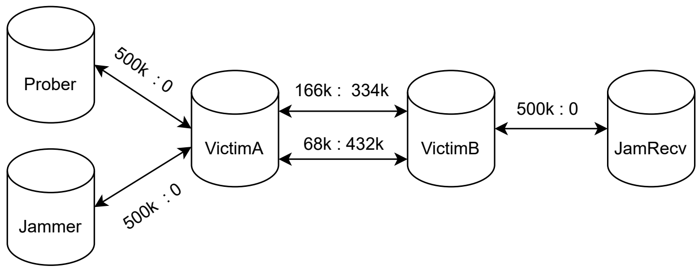

Fig. 15. Experimental setup in an isolated network

Jamming-Enhanced Probing To infer the balance of the smaller channel, the attacker sent $150k$ from Jammer to JamRecv, which held it for two minutes. The available balances were $68k$:$432k$ and $16k$:$334k$ (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 could improve balance estimates for 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. The attacker then configured the Prober node to send only zero-fee probes and successfully inferred the balance of the smaller channel.

D Experiments on Synthetic Hops

In this experiment, we demonstrate how the hop structure influences information gain. We consider two-channel hops with considerably different capacities: either $c_{big} = 2^{20}$ or $c_{small} = 2^{15}$ satoshis. 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 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 twelve configurations (Table 2). We do not consider channels disabled in both directions.

For each configuration, we generated synthetic hops and measured the information gain and probing speed with optimized and non-optimized amount selection methods (Table 3).

Hop configurations 2-2 and 1-1 were most vulnerable. The configuration least prone to probing was 2-0 (0.49 information gain). In general, asymmetric hop configurations were 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 went through the larger channel, while the smaller one remained "masked" (if it was not the only channel enabled in a given direction).

Practically speaking, we conclude that 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).

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

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

Configuration	**Inf. gain**	**Speed (non-opt.)**	**Speed (opt.)**	**Speedup (%)**
2-2	0.98	0.99	1.0	1
2-2-big-small	0.58	0.51	0.91	78
2-2-small-big	0.59	0.51	0.91	78
1-1	1.0	1.0	1.0	0
1-1-big-small	1.0	1.0	1.0	0
1-1-small-big	1.0	1.0	1.0	0
2-1	0.76	0.76	0.98	28
2-1-big-small	0.58	0.53	0.95	80
2-1-small-big	0.99	0.99	1.0	1
2-0	0.49	0.99	0.99	0
2-0-big-small	0.57	1.0	1.0	0
2-0-small-big	0.57	1.0	1.0	0

References

1. Multi-path payments in LND: Making channel balances add up. https://lightning.engineering/posts/2020-05-07-mpp/ (2020)
2. Androulaki, E., Karame, G., Roeschlin, M., Scherer, T., Capkun, S.: Evaluating user privacy in Bitcoin. In: Sadeghi, A. (ed.) Financial Cryptography and Data Security - 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers. Lecture Notes in Computer Science, vol. 7859, pp. 34–51. Springer (2013). https://doi.org/10.1007/978-3-642-39884-1_4, https://eprint.iacr.org/2012/596
3. Awathare, N., Suraj, Akash, Ribeiro, V.J., Bellur, U.: REBAL: channel balancing for payment channel networks. In: 29th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS 2021, Houston, TX, USA, November 3-5, 2021. pp. 1–8. IEEE (2021). https://doi.org/10.1109/MASCOTS53633.2021.9614304
4. BOLT: Lightning network specifications. https://github.com/lightningnetwork/lightning-rfc (2019)
5. Béres, F., Seres, I.A., Benczúr, A.A.: A cryptoeconomic traffic analysis of Bitcoin's Lightning network. Cryptoeconomic Systems (6 2020), https://cryptoeconomicsystems.pubpub.org/pub/b8rb0ywn
6. Conoscenti, M., Vetrò, A., Martin, J., Spini, F.: The CLoTH simulator for HTLC payment networks with introductory Lightning network performance results. Inf. **9**(9), 223 (2018)
7. van Dam, G., Kadir, R.A., Nohuddin, P.N.E., Zaman, H.B.: Improvements of the balance discovery attack on Lightning network payment channels. In: Hölbl, M., Rannenberg, K., Welzer, T. (eds.) ICT Systems Security and Privacy Protection - 35th IFIP TC 11 International Conference, SEC 2020, Maribor, Slovenia, September 21-23, 2020, Proceedings. IFIP Advances in Information and Communication Technology, vol. 580, pp. 313–323. Springer (2020). https://doi.org/10.1007/978-3-030-58201-2_21, https://eprint.iacr.org/2019/1385
8. EmelyanenkoK: Payment channel congestion via spam-attack. https://github.com/lightningnetwork/lightning-rfc/issues/182 (2017)
9. Engelmann, F., Kopp, H., Kargl, F., Glaser, F., Weinhardt, C.: 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). https://doi.org/10.1145/3152824.3152826, https://arxiv.org/abs/1711.02597
10. Gudgeon, L., Moreno-Sanchez, P., Roos, S., McCorry, P., Gervais, A.: SoK: Layer-two blockchain protocols. In: Bonneau, J., Heninger, N. (eds.) Financial Cryptography and Data Security - 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10-14, 2020. Lecture Notes in Computer Science, vol. 12059, pp. 201–226. Springer (2020). https://doi.org/10.1007/978-3-030-51280-4_12, https://eprint.iacr.org/2019/360
11. Harris, J., Zohar, A.: 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. pp. 202–213. ACM (2020). https://doi.org/10.1145/3419614.3423248, https://arxiv.org/abs/2006.08513
12. Hase, T., Wallace, V.: Smarter autopilot. https://blog.lightning.engineering/announcement/2019/04/23/mainnet-app.html (Apr 2019)

13. Hearn, M., Spilman, J.: Anti dos for tx replacement. https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2013-April/002417.html (2013)
14. Herrera-Joancomartí, J., Navarro-Arribas, G., Pedrosa, A.R., Pérez-Solà, C., García-Alfaro, J.: On the difficulty of hiding the balance of Lightning network channels. In: Galbraith, S.D., Russello, G., Susilo, W., Gollmann, D., Kirda, E., Liang, Z. (eds.) Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, AsiaCCS 2019, Auckland, New Zealand, July 09-12, 2019. pp. 602–612. ACM (2019). https://doi.org/10.1145/3321705.3329812, https://eprint.iacr.org/2019/328
15. Jager, J.: A proposal for up-front payments. https://lists.linuxfoundation.org/pipermail/lightning-dev/2020-March/002585.html (2020)
16. Jager, J.: Circuit breaker. https://github.com/lightningequipment/circuitbreaker (2021)
17. Kappos, G., Yousaf, H., Piotrowska, A.M., Kanjalkar, S., Delgado-Segura, S., Miller, A., Meiklejohn, S.: An empirical analysis of privacy in the Lightning network. In: Borisov, N., Díaz, C. (eds.) Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part I. Lecture Notes in Computer Science, vol. 12674, pp. 167–186. Springer (2021). https://doi.org/10.1007/978-3-662-64322-8_8, https://arxiv.org/abs/2003.12470
18. Khalil, R., Gervais, A.: Revive: Rebalancing off-blockchain payment networks. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017. pp. 439–453. ACM (2017). https://doi.org/10.1145/3133956.3134033, https://eprint.iacr.org/2017/823
19. Kumble, S.P., Roos, S.: Comparative analysis of lightning's routing clients. In: IEEE International Conference on Decentralized Applications and Infrastructures, DAPPS 2021, Online Event, August 23-26, 2021. pp. 79–84. IEEE (2021). https://doi.org/10.1109/DAPPS52256.2021.00014
20. Malavolta, G., Moreno-Sanchez, P., Schneidewind, C., Kate, A., Maffei, M.: 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), https://eprint.iacr.org/2018/472
21. Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G.M., Savage, S.: A fistful of bitcoins: Characterizing payments among men with no names. login Usenix Mag. **38**(6) (2013), https://www.usenix.org/publications/login/december-2013-volume-38-number-6/fistful-bitcoins-characterizing-payments-among
22. Mizrahi, A., Zohar, A.: Congestion attacks in payment channel networks. In: Borisov, N., Diaz, C. (eds.) Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part II. Lecture Notes in Computer Science, vol. 12675, pp. 170–188. Springer (2021). https://doi.org/10.1007/978-3-662-64331-0_9, https://arxiv.org/abs/2002.06564
23. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf (2008)
24. Naumenko, G.: Preventing channel jamming. https://blog.bitmex.com/preventing-channel-jamming/ (2021)

25. Nisslmueller, U., Foerster, K., Schmid, S., Decker, C.: Toward active and passive confidentiality attacks on cryptocurrency off-chain networks. In: Furnell, S., Mori, P., Weippl, E.R., Camp, O. (eds.) Proceedings of the 6th International Conference on Information Systems Security and Privacy, ICISSP 2020, Valletta, Malta, February 25-27, 2020. pp. 7–14. SCITEPRESS (2020). https://doi.org/10.5220/0009429200070014, https://arxiv.org/abs/2003.00003
26. Pérez-Solà, C., Ranchal-Pedrosa, A., Herrera-Joancomartí, J., Navarro-Arribas, G., García-Alfaro, J.: Lockdown: Balance availability attack against Lightning network channels. In: Bonneau, J., Heninger, N. (eds.) Financial Cryptography and Data Security - 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10-14, 2020 Revised Selected Papers. Lecture Notes in Computer Science, vol. 12059, pp. 245–263. Springer (2020). https://doi.org/10.1007/978-3-030-51280-4_14, https://eprint.iacr.org/2019/1149
27. Pickhardt, R.: 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)
28. Poelstra, A.: Lightning in scriptless scripts. https://lists.launchpad.net/mimblewimble/msg00086.html (Mar 2017)
29. Poon, J., Dryja, T.: The Bitcoin Lightning network: Scalable off-chain instant payments. Tech. rep. (2016)
30. Research, B.: Proportion of public vs private channels. https://blog.bitmex.com/lightning-network-part-7-proportion-of-public-vs-private-channels/ (2020)
31. Riard, A.: Route blinding. https://github.com/lightningnetwork/lightning-rfc/pull/765#pullrequestreview-511147029 (Oct 2020)
32. Riard, A., Naumenko, G.: Stake certificates. https://thelab31.xyz/stake-certificates (2020)
33. Riard, A., Naumenko, G.: Time-dilation attacks on the Lightning network. Cryptoeconomic Systems **1**(2) (10 2021). https://doi.org/10.21428/58320208.6ac6960a, https://cryptoeconomicsystems.pubpub.org/pub/riard-lightning-dilation
34. Rohrer, E., Malliaris, J., Tschorsch, F.: 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. pp. 347–356. IEEE (2019). https://doi.org/10.1109/EuroSPW.2019.00045, https://arxiv.org/abs/1904.10253
35. Rohrer, E., Tschorsch, F.: 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. pp. 214–227. ACM (2020). https://doi.org/10.1145/3419614.3423262, https://arxiv.org/abs/2006.12143
36. Romiti, M., Victor, F., Moreno-Sanchez, P., Nordholt, P.S., Haslhofer, B., Maffei, M.: Cross-layer deanonymization methods in the lightning protocol. In: Borisov, N., Díaz, C. (eds.) Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part I. Lecture Notes in Computer Science, vol. 12674, pp. 187–204. Springer (2021). https://doi.org/10.1007/978-3-662-64322-8_9, https://arxiv.org/abs/2007.00764
37. Russel, R.: A proposal for up-front payments. https://lists.linuxfoundation.org/pipermail/lightning-dev/2019-November/002275.html
38. Russel, R.: Loop attack with onion routing.. https://lists.linuxfoundation.org/pipermail/lightning-dev/2015-August/000135.html (Aug 2015)

39. Seres, I.A., Gulyás, L., Nagy, D.A., Burcsi, P.: Topological analysis of Bitcoin's Lightning network. In: MARBLE. pp. 1–12. Springer (2019), https://arxiv.org/abs/1901.04972
40. Tang, W., Wang, W., Fanti, G.C., Oh, S.: Privacy-utility tradeoffs in routing cryptocurrency over payment channel networks. In: Yeh, E., Markopoulou, A., Tay, Y.C. (eds.) Abstracts of the 2020 SIGMETRICS/Performance Joint International Conference on Measurement and Modeling of Computer Systems, Boston, MA, USA, June, 8-12, 2020. pp. 81–82. ACM (2020). https://doi.org/10.1145/3393691.3394213, https://arxiv.org/abs/1909.02717
41. Teinturier, B.: Trampoline onion format (feature 24/25). https://github.com/lightningnetwork/lightning-rfc/pull/836
42. Teinturier, B.: Spamming the Lightning network. https://github.com/t-bast/lightning-docs/blob/master/spam-prevention.md#costless-channel-probing (Nov 2020)
43. Tikhomirov, S., Moreno-Sanchez, P., Maffei, M.: 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), https://eprint.iacr.org/2020/303
44. Tikhomirov, S., Pickhardt, R., Biryukov, A., Nowostawski, M.: Probing channel balances in the Lightning network. CoRR **abs/2004.00333** (2020), https://arxiv.org/abs/2004.00333
45. Tochner, S., Schmid, S., Zohar, A.: Hijacking routes in payment channel networks: A predictability tradeoff. CoRR **abs/1909.06890** (2019), https://arxiv.org/abs/1909.06890
46. Tsabary, I., Yechieli, M., Manuskin, A., Eyal, I.: MAD-HTLC: because HTLC is crazy-cheap to attack. In: 42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021. pp. 1230–1248. IEEE (2021). https://doi.org/10.1109/SP40001.2021.00080, https://arxiv.org/abs/2006.12031
47. Zhang, Y., Yang, D., Xue, G.: Cheapay: An optimal algorithm for fee minimization in blockchain-based payment channel networks. In: ICC 2019 - 2019 IEEE International Conference on Communications (ICC). pp. 1–6 (2019). https://doi.org/10.1109/ICC.2019.8761804
48. ZmnSCPxj: Outsourcing route computation with trampoline payments. https://lists.linuxfoundation.org/pipermail/lightning-dev/2019-April/001950.html (2019)
49. ZmnSCPxj: A payment point feature family. https://lists.linuxfoundation.org/pipermail/lightning-dev/2019-October/002225.html (Oct 2019)