Improvements of the Balance Discovery Attack on Lightning Network Payment Channels

Gijs van Dam(p95677@siswa.ukm.edu.my)[0000−0002−6188−6859], Rabiah Abdul Kadir[0000−0003−3897−0873], Puteri N.E. Nohuddin[0000−0003−0627−5630], and Halimah Badioze Zaman[0000−0003−1553−3785]

Institute of Visual Informatics, The National University of Malaysia, 43600 UKM Bangi, Selangor http://www.ivi.ukm.my/en/

Abstract. The Lighting Network (LN) is a network of micropayment channels that runs on top of Bitcoin. The balances of payment channels are not broadcasted to the LN network to preserve the privacy of the nodes participating in the network. A balance disclosure attack (BDA) has been proven to be successful in determining the balance of large amount of channels in the network. In this paper we propose an improved algorithm for the BDA as well as a new type of attack that leverages the differences between LN client software implementations. Our improved algorithm extends the original BDA by performing payments from both sides of the channel. The new attack uses malformed payments to shutdown payment channels an adversary isn't part of.

Keywords: Bitcoin · Lightning Network · Network Security

1 Introduction

Bitcoin, the cryptocurrency with the largest market capitalization, has inherently limited scalability. Bitcoin generates 1 block of transactions every 10 minutes and the size of that block is limited to 1MB. With a basic transaction taking up 250 bytes and an average transaction size of 500 bytes the network has a maximum capacity of 4000 transactions per block and an average capacity of 2000 transactions per block. This boils down to 3-7 transactions per second. Increasing this capacity by either increasing the block size or the rate at which blocks are generated reduces the security of the Bitcoin network [1]. Increasing scalability of Bitcoin without abandoning security remains desirable. Firstly because if the amount of transactions being broadcasted

exceeds the capacity of the network, the law of supply and demand dictates that the transaction fees will increase [2] Secondly, if we want to achieve a viable alternative to current centralized payment networks we need to achieve comparable throughput which is in the order of magnitude of several thousand transactions per second.

It was Satoshi Nakamoto, the mysterious pseudonymous person or group of persons famous for developing Bitcoin, who suggested the use of transaction replacement for something he called high frequency trading [3]. In Nakamoto's proposal a group of parties could keep updating a transaction that had yet to be committed. The order of the updates was kept by a sequence number. Only the last agreed upon transaction needed to be broadcast. By doing so all the transactions prior to the final transaction were kept off-chain. The proposed solution depended on miners to commit the final transaction (the transaction with the highest sequence number). Since there is no incentive for miners to respect the sequence number [4], one of the involved parties could collude with a miner to have a non-final version committed, possibly stealing funds from the other parties. As such the solution couldn't operate in a trustless environment.

The scalability issues that face Bitcoin have renewed the interest in some form of transaction replacement by using them in a payment channel network (PCN). LN has emerged as the most prominent PCN to date [5] and is currently the only PCN in production.

LN is a peer-to-peer (P2P) network of connected nodes that uses Poon-Dryja [6] payment channels. Two connected nodes can open up a payment channel between them. A transaction from node A to node B can only happen if there is enough balance on the side of node A. Likewise, a transaction from node B to node A can only happen if there is enough balance on the side of node B. Both balances added together define the capacity of the channel. To create a transaction between two nodes that don't have a payment channel between them, multiple payment channels can be connected to form a route, as long as the balances along that route allow for the payment. This is known as a multi-hop payment. To participate in LN you have to run LN client software. Each LN client follows the LN specifications, set out in the Basis of Lightning Technology (BOLT) [7] documents.

Balances on the other hand are kept private and are never broadcasted on the network. The only balances known to a node are the balances of the channels that node participates in. Because of this it is impossible to know upfront whether a multi-hop payment will succeed, and there is only one way to find out: executing the payment. By executing multiple (fake) payments it is possible to probe the unknown balance of a payment channel. Disclosing balances this way has been dubbed the balance discovery attack (BDA) [8]. LN uses an onion routing [9] scheme called Sphinx [10] for the routing of payments.

In this study we analyzed potential BDA algorithm improvements and the role of LN client software in BDA. We propose an improvement to the basic algorithm for BDA that achieves a two-fold increase of the upper limit of balances that can be disclosed. Furthermore, we found that in certain situations LN client software can be leveraged to remove the upper limit of BDA completely. Finally we describe a specific situation where the interplay between different LN client software types leads to the permanent shutdown of a payment channel. This new attack, dubbed Payment of Death (POD), makes it possible to remotely shut down channels. We will show that POD is a threat to integrity of LN, as it has the potential for a malicious party to shutdown 17.5% of the network capacity.

2 Background

A formal analysis of privacy in the context of PCNs has been hindered by a lack of a rigorous definition of the PCN protocols, the absence of a threat model, and the ambivalent interpretations of the concept of anonymity [5].

A threat model is necessary to perform a formal analysis of privacy in the setting of trustless PCNs. Malavolta [5] describes a threat model with four notions of interest:

- Balance security: participants don't run the risk of losing coins to a malevolent adversary.
- Serializability: executions of a PCN are serializable as understood in concurrency control of transaction processing, i.e. for every

concurrent processing of payments there exists an equivalent sequential execution.
- (Off-path) value privacy: malicious participants in the network cannot learn information about payments they aren't part of.
- (On-path) relationship anonymity: given at least on honest intermediary, corrupted intermediaries cannot determine the sender and the receiver of a transaction better than just by guessing.

2.1 Balance Discovery Attack

In the basic scenario for channel balance discovery [8] it is assumed that there is an open payment channel AB between Alice, A, and Bob, B, with capacity C_{AB}. The goal of the adversary, Mallory, M, is to discover the balances of each node in channel AB: $balance_{AB}$ and $balance_{BA}$. To do so Mallory opens up a channel with Alice (see fig. 1).

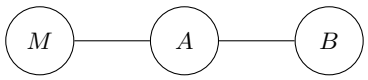

Fig. 1. Basic BDA were the adversary Mallory tries to disclose the balance between Alice and Bob

Mallory tries to to disclose $balance_{AB}$ by routing invalid payments through $M \leftrightarrow A \leftrightarrow B$, using the basic BDA algorithm. The inputs parameters for the algorithm are the target node B, the route to the target node, the value range to search in, given by 0 and C_{AB}, and the required accuracy for the algorithm. The algorithm creates invalid payments by using random, invalid payment hashes for each payment. The value for each payment follows a binary search pattern for which the initial lower and upper bounds are given by the value range input.

Bob, the recipient of the payment, is the only one who can determine that a payment from Mallory is invalid. Therefore, receiving an error stating the payment hash is invalid, means that $balance_{AB}$ was sufficient to route the payment, because if it was not, Alice would have returned an error stating insufficient funds and Bob would never have known about the payment. This fact is leveraged by

updating the lower bound of the binary search to the value of the last payment. If however the failure message states insufficient funds, the upper bound is updated with the value of the last payment. This process repeats itself recursively until the difference between the upper bound and the lower bound of the binary search is within the threshold set by the accuracy input. The algorithm returns a tuple that gives the range within which $balance_{AB}$ sits. Since the capacity of the channel C_{AB} is known, the $balance_{BA}$ can be calculated with $balance_{BA} = C_{BA} - balance_{AB}$.

By periodically executing a BDA, an adversary van monitor balances over time. This allows for tracing transactions. Therefore, this type of attack poses a threat for the value privacy as described in the threat model above.

3 Method

In order to research the role of LN client software in BDA we must first determine which LN clients are available. We used 1ML Lightning Network Search and Analysis Engine[1] to estimate respective proportions of each client in LN. 1ML is a website that publishes the current state of the LN graph and allows for node owners to self-report on a voluntary basis the type of client they use.

We chose the three LN clients with the largest network share to run a local cluster of LN nodes, each node running one of three supported clients. All LN nodes used Bitcoin Core's Bitcoind implementation as the Bitcoin backend. Bitcoind ran in regression testing mode, known as regtest mode. This is a local test mode, making it possible to almost instantly create blocks with no real world value. Using regtest mode, the different implementations could be tested without incurring transaction fees for the on-chain transactions and without having to wait for blocks to be mined.

On this cluster we analyzed the basic and improved algorithm having the LN nodes in each possible permutation of supported clients. This helped us determine if the new algorithm was to be considered an improvement and whether client differences could play a role in BDA.

[1] https://1ml.com/

3.1 Two-way channel probing

The original algorithm [8] is bound by an upper limit set by MAX_-PAYMENT_ALLOWED. This limit makes it impossible to probe balances that are higher than $2^{32} - 1\,msat$. This paper proposes an improved algorithm.

Consider a channel AB with capacity C_{AB}. Since $C_{AB} = C_{BA} = balance_{AB} + balance_{BA}$, the following holds

$$C_{AB} < 2^{33} \implies min\{balance_{AB}, balance_{BA}\} < 2^{32}$$

For all channels with a capacity $C_{AB} < 2^{33}$ there's always a balance lower than $\frac{2^{33}}{2} = 2^{32}$ on one end of the channel. With this knowledge we can extend the algorithm by letting it probe the channel from the other side, once we assess that the balance is higher than MAX_-PAYMENT_ALLOWED on the initial probing side. This setup requires an optional second channel from the adversary Node M to Node B, to be able to probe the channel between Node A and Node B from the side of Node B. (See fig. 2)

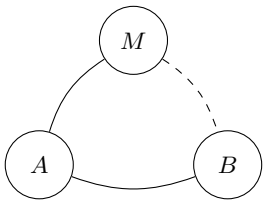

Fig. 2. Basic scenario with an optional second channel for two-way probing

Algorithm 1 describes BDA with optional two-way probing for channels with a capacity above MAX_PAYMENT_ALLOWED. Algorithm 1 takes the same input parameters as the basic algorithm and returns the same tuple.

If C_{AB} is higher than than MAX_PAYMENT_ALLOWED, the algorithm will try to send a fake payment with a size of exactly MAX_PAYMENT_ALLOWED. If that payment is possible, we have assessed that we are on the wrong end of the channel for probing the balance. The algorithm now calls itself with the target node and the final node of the route switched. The algorithm assumes that

there's a route from the adversary to this new target node. The return value of that call is $balance_{BA}$, for calculating $balance_{AB}$ we use the following formula:

$balance_{AB} = C_{BA} - balance_{BA}$

If $C_{AB} > MAX\_PAYMENT\_ALLOWED \wedge C_{AB} < 2 \times MAX\_PAYMENT\_ALLOWED$, the value of the first payment will not be exactly in the middle of the value range for the binary search, since it will use the fixed value of MAX_PAYMENT_ALLOWED for the first payment. That makes this algorithm slightly less computationally efficient then a perfect binary search, but it minimizes the use of the optional second channel.

4 Results

We confirmed the improvements provided by the two-way probing algorithm in two ways. Firstly we confirmed the feasibility of the algorithm in our local testing cluster. Secondly we have analyzed the LN running on top of Bitcoin mainnet, to estimate the number of channels that can have their balances disclosed by this algorithm and compare this to the earlier version of this attack.

4.1 Local Network Evaluation

We ran the Two-way Probing algorithm with every possible permutation of clients. By analyzing the responses from the clients, and analyzing the code of the respective clients on GitHub, we found that not every client implemented the MAX_PAYMENT_ALLOWED the same way.

On May 23rd, 2017 the BOLT specification was changed[2] by Paul "Rusty" Russel, who authored the majority of the BOLT documents. The variable containing the payment amount, amount_msat, was changed from a 32 bit unsigned integer to a 64 bit unsigned integer. This meant that before that change it was impossible to create a payment bigger than $2^{32} - 1$ whatsoever, but after that change in

[2] https://github.com/lightningnetwork/lightning-rfc/commit/068b0bccf94e8cdaf5f298dade0fcc8cc8421ef6#diff-3369c5aa1774fef2ff1e246979f223eaR590

Algorithm 1 Two-way Probing

Data: route, target, maxFlow, minFlow, accuracy_treshold
Result: bwidth, an array of tuples that gives the range of bandwidth discovered for each channel

$missingTests \leftarrow True$
$bwidth.max \leftarrow maxFlow$
$bwidth.min \leftarrow minFlow$
$channelCapacity \leftarrow getInfo(target).capacity$
while missingTests **do**
  if $bwidth.max - bwidth.min \leq accuracy\_threshold$ **then**
    $missingTests \leftarrow False$
  end if
  if $bwidth.max \geq 2^{32}$ **then**
    $flow \leftarrow 2^{32} - 1$
  else
    $flow \leftarrow (bwidth.min + bwidth.max)/2$
  end if
  $h(x) \leftarrow RandomValue$
  $response \leftarrow sendFakePayment(route = [route, target], h(x), flow)$
  if $response = UnknownPaymentHash$ **then**
    if $bwidth.min < flow$ **then**
      $bwidth.min \leftarrow flow$
    end if
  else if $response = InsufficientFunds$ **then**
    if $bwidth.max > flow$ **then**
      $bwidth.max \leftarrow flow$
    end if
  end if
  if $bwidth.min = 2^{32} - 1$ **then**
    $newTarget \leftarrow route.pop()$
    $route \leftarrow route.push(target)$
    $bwidthBA \leftarrow twowayProbing(route, newTarget, bwidth.min, 0, accuracy\_treshold)$
    $bwidth.min \leftarrow channelCapacity - bwidthBA.max$
    $bwidth.max \leftarrow channelCapacity - bwidthBA.min$
    $missingTests \leftarrow False$
  end if
end while
return *bwidth*

theory it was possible to create bigger payments. Additional specifications required the sending node to set the four most significant bytes of amount_msat to 0. But those additional requirements aren't implemented equally by the three main clients.

C-lightning is the only client that fully adheres to the requirements. Eclair has a limit of $5 \cdot 10^9 msat$. LND doesn't verify the for the each RPC. By using the unverified RPC in our algorithm we could send fake payments up to the maximal channel capacity. This meant that we can disclose any balance between two LND Nodes, even if the balance is above the upper limit of the two-way probing algorithm. In the scenario's where Alice is a LND node and Bob is an Eclair node or both are Eclair nodes, balances up to $5 \cdot 10^9 msat$ can be disclosed without making use of two-way probing.

In the case where Alice is a LND node and Bob is a C-lightning node, we saw interesting behavior of the C-lightning node which turned out to be a vulnerability of the current LN that can be exploited.

4.2 Payment of Death

If a C-lightning node is being requested to route a payment to another node, or is the receiver of a payment, with an amount that his higher than MAX_PAYMENT_ALLOWED, it decides to fail the channel with the requesting node and close down that channel. Since LN uses onion routing, the requesting node from the perspective of the c-lightning node, is the one that comes just before it in the route. But that isn't necessarily the node from which the payment originated.

Consider the basic scenario (see fig. 1), where Mallory and Alice run LND, and Bob runs c-lightning. Both channels between Mallory and Alice and between Alice and Bob have balances that allow for payments bigger than the MAX_PAYMENT_ALLOWED limit. If Mallory would create a fake payment with an amount above that limit, Bob would close down it's channel with Alice, without Alice being able to mitigate this in any way. We coined the term Payment of Death for this attack, after the infamous Ping of Death.

We have notified the developers of the LN implementations by means of a responsible disclosure.

4.3 Channels affected

The two-way probing algorithm works regardless of the client software. So we can look at the channel capacity of all public channels in the LN graph and determine the proportion of channels that are now susceptible to this type of attack based on a snapshot of the network taken on the 3rd of October, 2019 (see fig. 3).

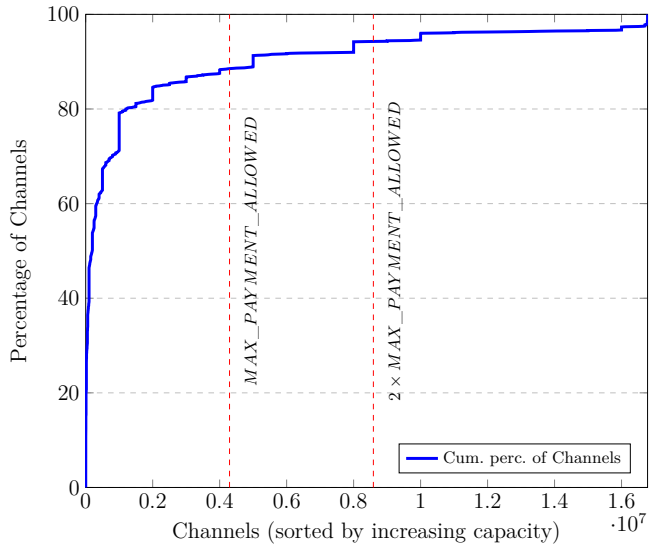

Fig. 3. Cumulative percentage graph of payment channels ordered by increasing capacity. $MAX\_PAYMENT\_ALLOWED$ shows the percentage of channels with disclosable balances using the basic BDA algorithm. $2 \times MAX\_PAYMENT\_ALLOWED$ shows the percentage of channels with disclosable balances using the two-way probing algorithm.

To estimate the number of channels susceptible for Balance Disclosure above the 2^{33} limit set by the Two-way algorithm, we need to know the type of client on either side of a channel. There's no known way of figuring out what kind of client is installed, but if you know the proportion of each client type in the LN, it is possible to estimate the amount of channels for each specific combination of clients.

We queried 1ML for each node in our snapshot of the LN. We identified the client type for 273 nodes out of 3608 and estimated the proportion of nodes running different clients based on that data. (See tbl. 1)

Table 1: Proportion of nodes running different Lightning clients

Client	n	Proportion (%)	CI[3] (%)
LND	220	80.59	(79.35-81.83)
c-lightning	40	14.65	(13.54-15.76)
Eclair	11	4.03	(3.41-4.65)
Other	2	0.73	(0.47-1.00)

The amount of channels that are susceptible to the Payment of Death, can now be estimated with the following analysis.

The LN is a graph G, with the number of vertices $n = |G(V)|$ and the number of edges $m = |G(E)|$. Our analysis yielded the following values for n and m:

$n = 3608$ $m = 9438$ with 1086 channels having a capacity greater than 2^{32} and 540 channels having a capacity greater than 2^{33}.

The client software defines the type of the vertex. $type_l$ for LND nodes, $type_c$ for c-lightning nodes and $type_e$ for Eclair nodes. An edge is said to be of $type_{(l,c)}$ if it connects a $type_l$ vertex and a $type_c$ vertex. The graph is without self-loops and undirected, so edge $type_{(l,c)} \equiv type_{(c,l)}$. Since we know the proportions of the different vertex types we can calculate the probability of an edge being of a specific type

- $P(type_{(l,l)}) = 0.8059^2$
- $P(type_{(c,c)}) = 0.1465^2$
- $P(type_{(e,e)}) = 0.0403^2$
- $P(type_{(l,c)}) = 2 \times 0.8059 \times 0.1465$
- $P(type_{(l,e)}) = 2 \times 0.8059 \times 0.0403$
- $P(type_{(c,e)}) = 2 \times 0.1465 \times 0.0403$

Assuming vertex type and channel capacity have a covariance of zero, the number of edges of each edge type, having a capacity greater than 2^{33} is calculated as follows: $P(type_{([c,e,l],[c,e,l])}) \times 540$. We are interested

[3] 95% Confidence interval

in the $type_{(l,l)}$ and $type_{(l,e)}$ channels, because the $type_{(l,c)}$ channels are susceptible to the Payment of Death, which doesn't allow for discovering the balance. So the amount of channels with a capacity above 2^{32} is $540 \times P(type_{(l,l)}) + 540 \times P(type_{(l,e)}) = 386$ channels. So a total of $9438 - 540 + 386 = 9284$ channels have balances that can be disclosed. This is 98.4% of all channels.

For the amount of channels affected by the POD we are interested in all $type_{(l,c)}$ channels, with a balance above MAX_PAYMENT_ALLOWED. This is $1086 \times P(type_{(l,c)}) = 256$ channels, meaning that 2.7% of all channels can be shutdown by using malformed payments.

5 Discussion

Herrera-Joancomarti [8] reported that 89.10% of all channels could have their balances exactly disclosed. Our research showed that we can improve this to 98.37% of all channels, a 9.27 percentage point increase (See tbl. 2). The basic BDA performed slightly less in our snapshot of the LN network because in the period between the two snapshots of the 8th of January, 2019 and the 3rd of October, 2019, the percentage of channels with a capacity C of $C > 2^{32}$ slightly increased.

Table 2: Percentage of channels susceptible for the basic BDA and the two-way probing BDA

Disclosable channels	basic BDA (%)	two-way probing BDA (%)
$C \leq 2^{32}$	89.10	88.49
$C > 2^{32} \wedge C \leq 2^{33}$	0	5.79
$C \leq 2^{33}$	0	4.09
TOTAL	89.10	98.37

5.1 Impact of Payment of Death

The properties of the vulnerability make it so that the highly capitalized nodes are more vulnerable, since it are these nodes that have channels with a balance above MAX_PAYMENT_ALLOWED limit. The average capacity of those 1086 channels is 10196116 $msat$. Using that average combined with the estimated proportions of affected channels, 17.5% of the total capacity of the network could

be taken down with an organized attack. These proportions align with proportions earlier found through alternative methods [11]. It's reasonable to assume that these channels are responsible for routing a disproportionate amount of the payments on the network. So such an attack could have substantial impact on the ability to route payments of the network as a whole.

The closing of channels comes at a cost to the victim nodes, since you have to broadcast on-chain transactions for closing a channel and again for reopening it. Those transactions have transaction fees attached to it. Furthermore, channel age is used as heuristic for determining the reliability of a node for routing payments, so routing nodes have an incentive to keep channels open as long as possible.

5.2 Countermeasures

Clients should adhere to the BOLT specification, making it impossible to create payments with a value higher than MAX_PAYMENT_ALLOWED and deny to consider payments with a value above the MAX_PAYMENT_ALLOWED for routing. The latter would make it impossible to disclose balances above $2^{33}\,msat$. Secondly, clients shouldn't consider a malformed payment a reason for permanently closing down a channel. This would make it impossible to mount a POD attack.

6 Conclusion

This paper presented an improvement to the algorithm of the original BDA. We showed that by approaching a payment channel from both sides instead of from one side, payment channels with a higher capacity than in the original BDA are now also susceptible to this attack. Since monitoring balances over time makes it possible to detect payments, it can be used to learn information about payments an adversary isn't part of [5]. We exposed differences in the implementation of the BOLT specification by the main three clients. These differences led us to develop new attack that closes down payment channels where the attacker isn't part of. We estimated the proportions of each client in LN, by using self-reported information. Based

on these proportions we estimated that this attack can be used to take down an important part of LN's entire network capacity.

7 References

1. Sompolinsky, Y., Zohar, A.: Secure High-Rate Transaction Processing in Bitcoin (full version). Financial Cryptography and Data Security - 19th International Conference, FC 2015. 507–527 (2015).

2. Easley, D., O'Hara, M., Basu, S.: From mining to markets: The evolution of bitcoin transaction fees. Journal of Financial Economics. 134, 91–109 (2019). https://doi.org/10.1016/j.jfineco.2019.03.004.

3. Hearn, M.: Anti DoS for tx replacement, https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2013-April/002417.html, (2013).

4. Harding, D.A., Todd, P.: BIP-0125: Opt-in Full Replace-by-Fee Signaling. (2015).

5. Malavolta, G., Moreno-Sanchez, P., Kate, A., Maffei, M., Ravi, S.: Concurrency and Privacy with Payment-Channel Networks. In: Proceedings of the 2017 acm sigsac conference on computer and communications security - ccs '17. pp. 455–471. ACM Press, New York, New York, USA (2017). https://doi.org/10.1145/3133956.3134096.

6. Poon, J., Dryja, T.: The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments, https://www.bitcoinlightning.com/wp-content/uploads/2018/03/lightning-network-paper.pdf, (2016).

7. Basis of Lightning Technology, https://github.com/lightningnetwork/lightning-rfc/blob/master/00-introduction.md.

8. Herrera-Joancomartí, J., Navarro-Arribas, G., Ranchal-Pedrosa, A., Pérez-Solà, C., Garcia-Alfaro, J.: On the Difficulty of Hiding the Balance of Lightning Network Channels. 602–612 (2019). https://doi.org/10.1145/3321705.3329812.

9. Reed, M.G., Syverson, P.F., Goldschlag, D.M.: Anonymous connections and onion routing. IEEE Journal on Selected Areas in Communications. 16, 482–493 (1998). https://doi.org/10.1109/49.668972.

10. Danezis, G., Goldberg, I.: Sphinx: A compact and provably secure mix format. Proceedings - IEEE Symposium on Security and Privacy. 269–282 (2009). https://doi.org/10.1109/SP.2009.15.

11. Pérez-Solà, C., Ranchal-Pedrosa, A., Herrera-Joancomartí, J., Navarro-Arribas, G., Garcia-Alfaro, J.: LockDown: Balance Availability Attack against Lightning Network Channels, https://eprint.iacr.org/2019/1149, (2019).