Bank run Payment Channel Networks

No Author Given

No Institute Given

Abstract. Payment Channel Networks (PCNs) have been a promising approach to scale blockchains. However, PCNs lack liquidity, as large-amount or multi-hop payments may fail. Payment griefing is one of the attacks on PCNs' liquidity, where the payee withholds the preimage in Hash Time Locked Contract. Before a payment expires, coins involved in this payment cannot be used in other payments. We introduce Bankrun attack, which exploits payment griefing to "bank run" PCNs. Bankrun in finance means numerous clients withdraw their money from a bank, which makes the bank insolvent and even bankrupted. In our Bankrun attack, the adversary generates sybil nodes, establishes channels with hubs, makes payments between his nodes and griefs payments simultaneously. Consequently, the adversary "withdraws" capacity of nodes in the PCN, and nodes cannot make or route normal payments. We propose concrete steps of launching Bankrun attacks, and develop various strategies attacking different aspects of the liquidity. We propose a framework for quantifying the PCN's liquidity and the effectiveness of Bankrun attacks. We then evaluate bankrun attacks on Bitcoin's Lightning Network, the first and most well-known PCN. Our evaluation results show that, with an acceptable budget, an adversary can almost paralyse the entire Lightning Network, including reducing the success rate of payments; increasing the fee of payments; raising the attempt times of payments; and increasing the number of bankrupt nodes. The cost for launching bankrun attacks is small (∼187 USD), as it only consists of transaction fees for establishing channels.

1 Introduction

Public blockchains suffer from limited throughput. Payment Channel Network (PCN) – introduced by the Lightning Network (LN) [18] – is one of the promising ways to scale blockchains. Payment channels enable *off-chain* payments, i.e. payments that do not need to be recorded on the blockchain. To open a payment channel, two nodes collateralise some coins in a joint address. Two nodes make a payment by signing a new transaction that updates their balances. To close the channel, one of the two nodes commits the transaction recording the latest balance allocation to the blockchain. If two nodes do not have a direct channel, they can make payments to each other using *multi-hop payments*, i.e., payments going through one or more intermediate channels. In a multi-hop payment, the payer should find a path, i.e., a group of channels, that directs him to the payee. The payment is made by updating balances of these channels in an atomic way.

The atomic update can be achieved by Hash Time Locked Contracts (HTLCs). The HTLC enables a payer to lock a payment in a way such that the payee should reveal a hash value's preimage before a timeout to redeem the payment from the payer, otherwise the payment will expire. In a HTLC-based multi-hop payment, the payee chooses a preimage, and nodes make HTLC payments on all involved channels with this preimage's hash value. Revealing this preimage activates these HTLC payments simultaneously.

PCNs are suspected to lack *liquidity*, i.e., the ability of processing payments. Payment griefing [5] is an attack on PCNs' *liquidity*. In payment griefing, the adversary makes a payment and withholds the preimage, so that coins involved in this payment are locked and cannot be used in other payments before the payment expires. Payment griefing is free, as the payer does not need to pay anything for failed payments. Payment griefing is also unaccountable, as the intermediate nodes can not know who the payer and payee really are.

1.1 Bankrun attacks on PCNs

In this paper, we introduce the **Bankrun attack**, which exploits payment griefing to "bank run" the entire PCN. Bank run [9] is a concept in finance that, numerous clients withdraw their money from a bank simultaneously as they lose confidence in this bank. If the bank does not have enough balance, the bank will be short of money and eventually go bankrupt. Bank run usually happens when people lose confidence in this bank, e.g., during a financial crisis. In the Bankrun attack, the adversary generates some sybil nodes, establishes channels with existing nodes in the PCN, initiates numerous multi-hop payments between his nodes and griefs these payments simultaneously. Such concurrent griefing attacks greatly occupy nodes' capacity, and nodes can become insolvent towards normal payments. With sufficient coins, the adversary can lock a large portion of the PCN's capacity and thus paralyse the entire PCN.

We introduce a framework for launching Bankrun attacks with four concrete steps, namely 1) node selection, 2) payment enumeration, 3) payment ranking, and 4) launching attack. To evaluate Bankrun attacks, we introduce a framework to quantify the liquidity in PCN. We evaluate the Bankrun attack on Bitcoin's Lightning Network (LN) – the first and most well-known PCN. Our results show that Bankrun attacks can significantly damage the liquidity of PCNs. In particular, with direct channels to 1.5% richest nodes, the adversary can launch a Bankrun attack that locks 45% ($\sim$267 BTC) coins in the network; reduces success rate of payments by 21.4%$\sim$59.0%; increases fee of payments by 4.0%$\sim$15.0%; increases average attempts of payments by 42.0%$\sim$104.2%; and increase the number of bankruptcy nodes by 19.4%$\sim$131.7%, where the amounts of payments range from 100,000 to 1,900,000 satoshi [4].

While being destructive, Bankrun attacks are cheap to launch. Launching a Bankrun attack needs some transaction fee for establishing channels and some balance for griefing payments. The transaction fee is negligible: our evaluation shows that the total transaction fee used for attacking LN is approximately 187 USD. The adversary does not lose balance during the attack, as payments for

griefing will eventually expire. This means the adversary can use the balance to launch Bankrun attacks for unlimited times.

1.2 Roadmap

Section 2 provides the background of PCNs and griefing. Section 3 describes the security model and the Bankrun attack. Section 4 describes the evaluation framework of PCNs' liquidity. Section 5 evaluates Bankrun attacks on Lightning Network. Section 6 discusses the cost of Bankrun and strategy to utilise it for making a profit. Section 7 reviews relevant literature and Section 8 concludes this paper. Appendix A outlines detailed evaluation results. Appendix B provides some detailed analysis of the evaluation results. Appendix C discusses rank-by-fee strategy.

2 Background

2.1 Payment Channel Networks

Lightning Network [18] introduces the idea of Payment Channel Networks. A payment channel allows two parties to pay each other without the need to publish every payment to the blockchain. Instead, two parties collateralise their coins into a single on-chain transaction, and jointly sign it using a 2-2 multi-signature. This creates a payment channel, where each party controls his coins. They can make payments with each other by mutually signing new transactions with updated amounts of their collateralised coins. Before closing the payment channel, two parties will not commit these subsequent payments to the blockchain. To close the channel, one party commits the latest state of channel balance to the blockchain, and coins in this channel will be allocated to both parties accordingly.

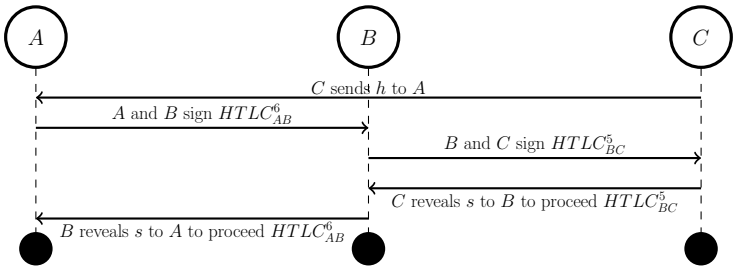


Fig. 1: A multi-hop payment from A to C via an intermediate node B.

The payment channel can be further extended to support offline payments that go through multiple channels. Most multi-hop payment protocols are based on Hash Time Locked Contracts (HTLCs). HTLC is a contract between two parties that, a payment will be made if the payee shows the preimage of a hash

value within a timeout (represented as a block height on the blockchain). If the payee does not show the preimage and the time has reached the timeout, the payment will expire and deemed invalid.

Figure 1 describes a multi-hop payment where A pays 5 BTC to C via an intermediate node B in Bitcoin's Lightning Network. First, C chooses a random string s as preimage and send its hash value $h = H(s)$ to A, where $H(\cdot)$ is a cryptographic hash function. A then signs a HTLC contract $HTLC^6_{AB}$ with B stating "A will pay 6 BTC to B if B can show the value of s within a given timeout (say 144 blocks)". B also signs a HTLC contract $HTLC^5_{BC}$ with C saying that "B will pay 5 BTC to C if C can show the value of s within a given timeout (say 138 blocks)". Then C shows s to B to redeem 5 BTC in $HTLC^5_{BC}$ from B. Meanwhile, B can redeem 6 BTC in $HTLC^6_{AB}$ from A by revealing s to A. B is incentivised to reveal s, as B does not want to lose money. The timelock of AB is set to be longer than BC, so B always has sufficient time to reveal s to A.

By routing this payment, B gets 1 BTC from A. This is known as "fee", which is paid by the payer and is used for encouraging nodes to route multi-hop payments. In LN, fee consists of a fixed base fee and proportional fee that fluctuates according to the congestion level of the network. To minimise the cost, payers usually search for a path with the least fee when making payments.

2.2 Payment Griefing

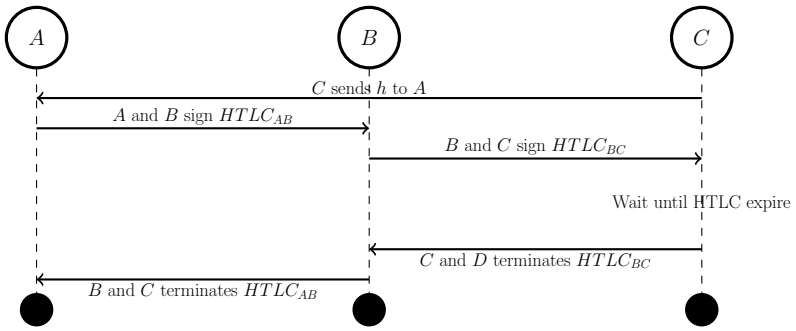


Fig. 2: Payment griefing attack.

If the payee C reveals the preimage on time and the intermediate node B is rational, the multi-hop payment will eventually happen. However, there exists an attack called **payment griefing**, where the payee withholds the preimage until HTLCs expire. Before HTLC expire, coins involved in all channels of this payment are locked and cannot route other payments.

Payment griefing is a threat to PCNs' liquidity. If a big portion of coins in a PCN are locked, the PCN will no longer be able to route payments. Payment

griefing is cheap, as the payment does not really happen and the payer does not pay for the fee to intermediate nodes. Identifying payment griefing can be hard, as nodes cannot distinguish whether the withholding is due to network delay, on purpose, or by accident. If the PCN's routing protocol is privacy-preserving, payment griefing can even be launched anonymously. For example, Bitcoin's Lightning Network adopts onion routing [11], where each intermediate node only has the knowledge of nodes who directly connect with him.

3 Bankrun attack

3.1 Security model

We consider nodes in the PCN are rational. They publish their routing fee standards, and accept all affordable routing requests. Each non-malicious node in a multi-hop payment will reveal the preimage of the hashlock to the upstream node once he knows it.

At the beginning, the Byzantine adversary does not control any node, but has the knowledge of all nodes in the PCN, including the network topology, the capacity and the fee standard of each channel. This can be achieved, taking lighting network as an example, by accessing all publically available data on the Bitcoin blockchain. When establishing a channel with a node, the node is willing to provide sufficient capacity. According to liquidity providers such as Bitrefill[3], purchasing capacity from existing nodes is easy and cost-effective.

For simplicity, we do not consider the impact of timelocks on our attack. Besides the timelock, a multi-hop payment has two parameters, namely length and amount. We define the length of a multi-hop payment l as the number of channels the payment involves, the amount X as the amount of coins that the payer wants to pay to the payee, and the size of a multi-hop payment θ as

$$\theta = X \cdot l \tag{1}$$

3.2 Overview of the Bankrun attack

Figure 3 shows the intuition of our Bankrun attack. In a Banrun attack, the adversary exploits the payment griefing to "bank run" PCNs. First, the adversary establishes payment channels with existing nodes in the PCN, and make numerous multi-hop payments between his nodes simultaneously. Then, the adversary withholds preimages until these payments expire. Before that, coins locked in these payments cannot be used in other payments. If the adversary has sufficient budget, he can lock a great portion of coins in the PCN so that the PCN may be paralysed.

This attack is similar to bank-run [9] in finance, where numerous clients withdraw their money from a bank simultaneously, so that the bank will run out of money and eventually bankrupt. In our Bankrun attack, an adversary pretends to be multiple payers and payees and launch payment griefing attacks

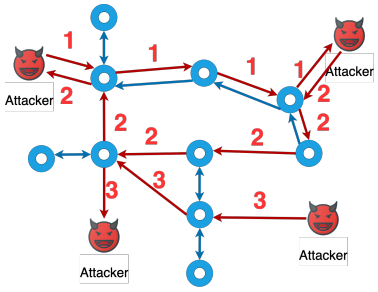


Fig. 3: Bankrun attack. The adversary generates sybil nodes (red), establishes payment channels with existing nodes (blue), and makes payments between his nodes and griefs these payments simultaneously.

simultaneously. Bank-run on both banks and PCNs leads to liquidity risk [9], where the system is insolvent and can no longer process payments. However, Bankrun attacks on PCNs can be much more dangerous than on banks. PCNs inherently lack liquidity, as PCNs are decentralised and coins are distributed among channels. Meanwhile, a bank can be treated as the only hub who routes all payments in a PCN. In addition, Bankrun attacks on PCNs can be cheap. The adversary only needs to spend negligible coins on opening channels, and coins for payment griefing will be refunded eventually.

We introduce a framework for launching Bankrun attacks. The framework consists of four critical components, namely 1) Node selection, the adversary chooses a set of nodes and establishes channels with them. 2) Payment enumeration, the adversary enumerates all payments between his nodes. 3) Path ranking, the adversary orders these payments. And 4) Launching attack, the adversary starts to make and abort payments with this order.

Section 3.3-3.6 will describe how to conduct these steps in detail.

3.3 Node selection

The adversary's first step is to join the PCN by establishing payment channels with existing nodes. Here come two questions that, which and how many nodes the adversary should establish channels with.

To answer the first question, we suggest establishing channels with rich nodes (aka. hubs), as a hub is likely to route more payments than a poor node. The answer of the second question depends on how the adversary enumerates payments for griefing (i.e., the next step). More specifically, we should establish channels with sufficient nodes so that the sum of sizes of enumerated payments takes the majority of the network capacity and starts to converge. When this sum starts to converge, the adversary should be able to lock sufficient coins, and cannot lock more by establishing channels with more nodes. Later in Section 5.2, we will show that the adversary only needs to establish channels with top 1.5% (42) richest nodes if attacking Bitcoin's Lightning Network.

3.4 Enumerating payments

Algorithm 1 Enumerating payments.

Input:
The entire network $\mathcal{G}$
The adversary's node list $\mathcal{N}$
The amount of payment α
Output:
The list of payments $\mathcal{P}$

$T \leftarrow []$
for (v_1, v_2) in $\mathcal{V}$ **do** ▷ Start from richest nodes
  $path\_list \leftarrow \mathsf{BFS}(\mathcal{G}, v_1, v_2)$
  for $path \in path\_list$ **do**
    $P \leftarrow \{path : [], amount : 0\}$ ▷ Initialise an empty payment
    $P[path] \leftarrow path$
    $capacity\_list \leftarrow$ [c.capacity for c in path.channels]
    $\alpha \leftarrow min(capacity\_list)$ ▷ Get the most viable amount of *path*
    if $\alpha = 0$ **then continue** ▷ This path is not viable
    $P[amount] \leftarrow \alpha$ ▷ P is a viable payment
    Append P to $\mathcal{P}$
    Consume P in $\mathcal{G}$ ▷ In simulated environment
  end for
end for
return $\mathcal{P}$

After establishing payment channels, the adversary enumerates all possible payments between his nodes. To this end, he should find all paths between each pair of his nodes, and calculate the maximum amount that each path can afford.

We first model the PCN as a weighted directed graph, where each channel consists of two edges with opposite directions, and each edge is weighted by its balance. As we cannot know real-time balances of channels, we use the initial channel capacity for now, which can be retrieved from transactions opening channels on the blockchain. When starting to attack (in Section 3.6), the adversary will try to make these payments by gradually decreasing the payment amounts.

Our payment enumerating algorithm builds upon the Ford-Fulkersons algorithm [10] - a maximum flow algorithm in graph theory. Maximum flow is a classic problem in graph theory, which aims at finding the maximum amount of flow that the network allows from a source to a sink. Ford-Fulkersons algorithm is one of the most effective algorithms to solve the maximum flow problem. Given a weighted directed graph and two vertices, Ford-Fulkersons algorithm first uses Breadth-First Search (BFS) to find all paths between these two vertices. For each path, the maximum viable amount is the minimum weight of edges.

Algorithm 1 describes the process of enumerating payments. Similar to Ford-Fulkersons, we employ BFS to find all paths between each two adversary nodes. Then, we derive the most viable amount using the least channel capacity for each path. Each path together with its most viable amount is a viable payment. Then, we consume this payment from the graph and add this payment to our payment list. Eventually, the payment list contains all viable payments.

3.5 Ranking payments

Griefing different payments have different impacts on the PCNs' liquidity. As the adversary's balance is limited, he should start from griefing important payments for maximising the attack's effect. Thus, the adversary should have a way of ranking payments in terms of their importance.

We first consider three ranking criteria according to Equation 1, namely the payment's length, amount and size. *Rank-by-length* aims at maximising the effect while minimising the cost, as long payments lock most capacity with the least amount. *Rank-by-amount* aims at attacking the network backbone. Since real-time balance in LN cannot be seen, payer tends to prefer to go through channels with large capacities to reduce attempt times. *Rank-by-size* aims at maximising the attack effect without considering the budget, as payments with large sizes cost most collateral.

Inspired by existing works, we propose two extra ranking criteria. The first – inspired by B'eres et al. [7] – is *rank-by-fee*, where the adversary first attacks channels with lower fees. This aims at maximising the average channel fee of normal payments after the attack. The other – inspired by Dandekar et al. [8] – is *rank-by-bankrupt*. Dandekar et al. [8] formalise credit networks, which can be utilised for modelling PCNs [19]. Their approach quantifies liquidity by evaluating the probability that nodes become *bankruptcy*. A node is bankrupt if its balance is lower than a given amount in our case. Dandekar et al. [8] proves that, the probability that a node v goes bankrupt is upper-bounded by $\frac{1}{\Gamma_v+1}$, where Γ_v is the total capacity of all channels connecting to v. In *rank-by-bankrupt*, the adversary first attacks payments that reduce the mathematical expectation of nodes becoming bankruptcy most. Formally, for payment P from node v_0 to node v_t with amount α, the criteria $Score(P)$ is

$$\begin{aligned} Score(P) = &\frac{1}{\Gamma_{v_0}-\alpha+1} - \frac{1}{\Gamma_{v_0}+1} + \sum_{i=1}^{t-1}\left[\frac{1}{\Gamma_{v_i}-2\alpha+1} - \frac{1}{\Gamma_{v_i}+1}\right] \\ &+ \frac{1}{\Gamma_{v_t}-\alpha+1} - \frac{1}{\Gamma_{v_t}+1} \end{aligned} \tag{2}$$

3.6 Launching attack

When enumerating payments, we use channels' capacities rather than their real-time balances for determining the amounts of payments. Thus, some of our

Algorithm 2 Launching attack.

Input:
The list of ranked payments $\mathcal{P}_{ranked}$
The dropping step of amount *step*
The budget of the adversary B

for P in $\mathcal{P}_{ranked}$ **do** ▷ Can be concurrent using multiple threads
  if $B \leq 0$ **then**
    return ;
  end if
  while $True$ **do**
    $response \leftarrow make\_payment(P)$
    if $response$ = InsufficientFunds **then**
      $P[amount] = P[amount] - step$
      if $P[amount] \leq 0$ **then break**
      continue
    end if
    $B = B - P[amount]$
    break
  end while
end for

enumerated payments may not succeed. In real-world PCNs, if a node cannot route a payment, the node will reply to the payer with an error message. For example, Bitcoin's Lightning Network calls this error *InsufficientFunds*.

We introduce a retry mechanism similar to Joancomarti et al. [14] for making payments. Algorithm 2 describes the attack process. If the payer receives *InsufficientFunds*, he will reduce the payment's amount by a parameter *step*, and retries until it is successful or the amount reaches zero. Unlike Joancomarti et al. [14] using binary search, we search for feasible amounts, from the largest possible to zero. This is because we aim at making payments successfully with the largest amounts, rather than revealing channels' balances with fewest attempts.

4 Evaluation framework of PCNs' liquidity

To evaluate the impact of Bankrun attacks, we introduce a framework for quantifying the PCNs' liquidity. To this end, we first generate a batch of payments, simulate them on the PCN, and observe their execution results. The metrics include the success rate, the average cost and the number of attempts of payments, and the number of bankruptcy nodes. Based on the framework, we can quantify the effectiveness of a Bankrun attack by comparing the liquidity before and after the attack.

4.1 Testing PCNs' liquidity

We follow the approach of Béres et al. [7] to test PCNs' liquidity. Specifically, we generate a batch of n payments, of which payers and payees are random and the amount x_t is fixed. Note that the focus of the attack may be different: some adversaries aim at discouraging small payments, while other adversaries aim at discouraging large payments. To this end, we test multiple batches of payments with different amounts, which will be discussed in Section 5.

We then simulate these payments in the PCN. We allow each payment to try r times for finding a viable path. If it finds a path within r tries, we consider it successful, otherwise failed. So the payments can be categorized into three states according to the status before and after the attack, namely *added*, *survived*, or *removed*. *Added* means the payment is failed before the attack but is successful after the attack. *Survived* means the payment is successful both before and after the attack. *Removed* means the payment is successful before the attack but is failed after the attack. As a result, we can measure pre-attack and post-attack scores based on the four metrics presented to quantify the impact of Bankrun.

4.2 Quantifying PCNs' liquidity

We then derive the PCNs' liquidity from the execution results of the batch of tested payments. We consider four metrics, including 1) the success rate of payments, 2) the average fee of payments paid to intermediate nodes, 3) the average attempt time of payments, and 4) the number of bankruptcy nodes.

Success rate p. The success rate of payments is a metric quantifying the liquidity. High success rate means the PCN has sufficient liquidity. We define the success rate p of payments as

$$p = \frac{n_{succ}}{n}$$

where n_{succ} is the number of successful payments, and n is the number of payments used for testing.

Average fee $\bar{f}$. In addition, the average fee of making payments is a metric. We only consider survived payments when evaluating the average fee. Survived payments might be forced to choose channels with a higher fee or go through more intermediate nodes, leading to higher fee. We define the average fee $\bar{f}$ as

$$\bar{f} = \frac{\sum_{i=1}^{n_{sur}} f_i}{n_{sur}}$$

where n_{sur} is the number of survived payments, and f_i is the fee of the i-th survived payment.

Average attempts $\bar{a}$. Moreover, the average number of attempts of making a payment is a metric. After the attack, making a payment will need more attempts for finding a viable path. We define the average attempts $\bar{a}$ as

$$\bar{a} = \frac{\sum_{i=1}^{n_{succ}} a_i}{n_{succ}}$$

where n_{succ} is the number of successful payments, and a_i is the attempt time of the i-th payment.

Number of bankruptcy nodes b**.** After processing the batch of payments, there may be some new bankruptcy nodes. We consider the number of bankruptcy nodes after these payments as another liquidity metric.

With this framework quantifying PCNs' liquidity, we can evaluate the effectiveness of Bankrun attacks by comparing the liquidity of PCNs before and after the attack.

5 Evaluation of Bankrun attacks

In this section, we evaluate Bankrun attacks on Bitcoin's Lightning Network (LN), the first and most well-known PCN. First, we show the impact of the most effective rank-by-length in naive strategies and demonstrate the cheapness of Bankrun. Second, we verify the effectiveness of rank-by-bankrupt experimentally in aspects of micropayment and bankruptcy. The remaining strategies will be discussed in the appendix due to their inferior performance and ineffectiveness. Our results show that the adversary who adopts Bankrun can greatly paralyse the entire PCN.

5.1 Experimental setting

We simulate and implement our attack using python 3.7.4 and NetworkX [12] - a Python library for complex networks. Similar to Béres et al. [7], we use the snapshot [6] of the Bitcoin Lightning Network as the dataset. It contains the network topology, capacities (but not balances) of channels, and fee standards of nodes. We randomly generate the balance for each channel, which is same as in [7]. We test attacks with all ranking criteria in 3.5, and $step = 0.1 * amount$ in Algorithm 2. We test attacks with different levels of budgets of the adversary, i.e., $\{7.7, 15.4, \ldots, 77\}$ BTC.

When testing LN's liquidity, we pick $n = 7,000$, $x_t = \{100,000, 700,000, 1,300,000, 1,900,000\}$ (satoshi), and $r = 10$. The range of amounts covers most scenarios using PCNs: 100,000 satoshi is approximately 10 USD, and 1,900,000 is approximately 196 USD. Each batch consists of 7,000 payments, which is similar to Béres et al. [7]. We allow a payment to try 10 times for finding a viable path. If it finds a path within 10 tries, we consider it successful, otherwise, we consider it failed. We consider the threshold of bankruptcy is 60,000 satoshi, which is the average amount of payments in LN [7].

5.2 Number of nodes for establishing channels

We test the percentage of the capacity of the entire network that the adversary can lock by establishing channels with different numbers of nodes on Bitcoin's Lightning Network. Figure 4 shows that, by establishing channels with top 1.5% (42) richest nodes, the enumerated payments take $\sim 83\%$ of the capacity of the

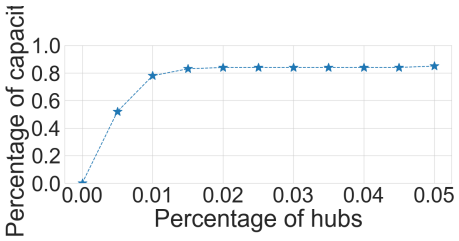
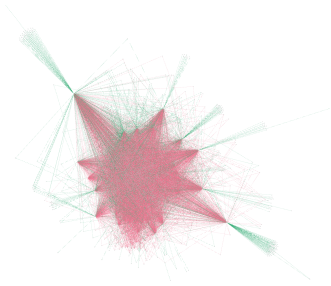


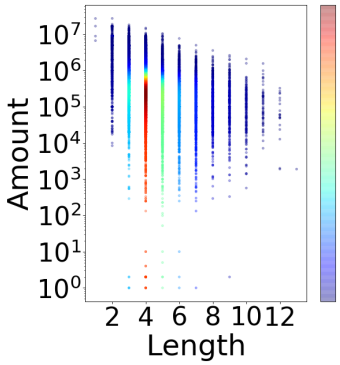
Fig. 4: The relationship between the percentage of hubs that the adversary establishes channels with (x axis) and the percentage of the capacity of the entire network that the enumerated payments take (y axis).

entire network. In addition, the sum of sizes of enumerated payments converges with the percentage of hubs increasing. Thus, if attacking Bitcoin's Lightning Network, we suggest the adversary establishing channels only with top 1.5% richest nodes.

5.3 Characterisation of enumerated payments



(a) Visualisation of enumerated payments. Red indicates edges influenced by our attack, and green indicates unaffected edges.



(b) Distribution of enumerated payments. Red indicates many overlapped points, and blue indicates few overlapped points.

Fig. 5: Characterisation of enumerated payments.

As aforementioned, we establish channels with 42 richest nodes in the network. Figure 5a visualises our enumerated payments. It shows that our attack can influence most channels in the network. Figure5b further visualises the distribution of amounts and lengths of enumerated payments. The amount ranges

from zero to 10^7 satoshi, and the length ranges from 1 to 13. Besides, most payments are with lengths of 3∼6 and with the amount of $10^3 \sim 10^6$ satoshi.

5.4 Impact of Bankrun attacks

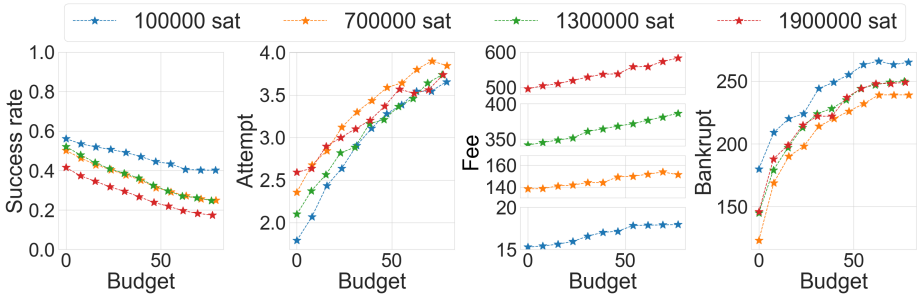
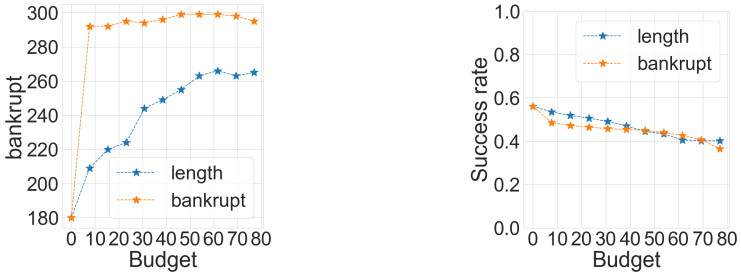


Fig. 6: Overview of impacts.

Figure 6 shows the impact of Bankrun attacks on LN. Here we only provide the results with the rank-by-length strategy, which achieves the best overall performance. Appendix A outlines the detailed evaluation results of all attack strategies. Our result shows that Bankrun attack follows this strategy can reduce the payments' success rates by 28.4%∼58.1%, increase the fee by 9.4%∼17.5%, increase payments' attempt times by 44.3%∼104.2%, and increase the number of bankruptcy nodes by 47.2%∼94.1%.



(a) Comparison of the bankrupt nodes.

(b) Comparison of success rate.

In addition, the rank-by-bankrupt strategy is particularly effective in the number of bankruptcy nodes and the success rate of small payments. Figure 7a and 7b compare rank-by-bankrupt and rank-by-length with payment amount

100,000 satoshi. The results show that when the attack budget is limited, rank-by-bankrupt outperforms rank-by-length in terms of the number of bankruptcy nodes and the success rate of small payments. In particular, the adversary can bankrupt 120 nodes and reduce the success rate of payments with 100,000 satoshi by 10% using rank-by-bankrupt and only 7.7 BTC.

6 Discussion

6.1 Budget analysis

To bank run PCNs, the adversary should have some coins as the budget. The attack's impact depends on the adversary's budget. For example, in LN, the adversary can paralyse small payments with 7.7 BTC (1.3% of the network capacity), and paralyse the entire LN with 77 BTC (13% of the network capacity). The required budget can be affordable, especially when the PCN is much smaller than the blockchain that supports it. In Bitcoin, there are more than 10,000 addresses with more than 157 BTC[2]. All of them can launch Bankrun attacks to paralyse the entire LN easily.

The adversary can also reuse the budget to launch Bankrun attacks for unlimited times. As griefing payments will eventually expire, the adversary neither loses money nor pays channel fees to intermediate nodes.

6.2 Cost analysis

Launching Bankrun attacks takes negligible cost. The cost consists of two parts, namely the transaction fee for establishing channels and the opportunity cost for locking coins in PCNs. The transaction fee is cheap: for example, to attack LN, the fee to establish 42 channels is about 187 USD [1]. The opportunity cost is the money that the adversary can earn by using the budget of coins to route payments. Existing research [7] shows that in LN, the opportunity cost – i.e., the return of investment (RoI) of routing payments – is only 0.002% per year.

6.3 Profit from Bankrun attacks

In PCNs, nodes can profit by charging fees for routing payments. The more payments a node routes, the more profit it makes. To maximise the profit, an adversary can bank run channels with large capacity and force more payments to go through its own channels. This can be seen as an inverse version of the routing hijacking attack [22], where the adversary sets nodes with low channel fees to attract payments. To obtain optimal profit, the adversary should maximise the probability that payments go through its nodes by draining the most important channels. Existing research [22] shows that the probability of payments going through a node is proportional to the node's *betweenness centrality*, and maximising the *betweenness centrality* by removing channels can be formalised as an NP-hard problem called *destructive betweenness improvement* [15]. To our knowledge, there exists no good approximation algorithm to solve this problem, and we consider this as our future work.

7 Related work

Dan Robinson [20] first discussed the griefing problem of HTLCs, and Interledger RFCs [5] discussed payment griefing attacks in PCNs. Our Bankrun attack exploits payment griefing attacks to reduce the entire PCN's liquidity. A closely related work is the Congestion attack [16], where the adversary floods a PCN with numerous small payments to prevent nodes from routing normal payments. However, each node can parametrise the number of concurrent payments by adjusting a parameter *max_concurrent_htlcs*. The Congestion attack can be mitigated simply by increasing *max_concurrent_htlcs*.

There have been emerging attacks on PCNs that exploit other vulnerabilities or focus on different aspects of PCNs. Harris et al. [13] propose the flood-and-loot attack, where the adversary congests the blockchain, triggers disputes over multi-hop payments on PCNs, and steals money as nodes cannot commit transactions for disputes. Saar et al. [22] propose the hijacking attack, where the adversary publishes channels with a cheap fee to attract payments and eventually withhold them. While the Bankrun attack exploits griefing, the flood-and-loot attack exploits blockchains' limited throughput, and the hijacking attack exploits nodes' rationality. Rohrer et al. [21] discuss two attacks, namely channel exhaustion and Node isolation. Pérez-Sola et al. [17] formalise Node isolation as Lockdown attack. While the Bankrun attack aims at paralysing the entire PCN, these three attacks aim at exhausting individual channels or isolating individual nodes.

8 Conclusion

In this paper, we propose Bankrun attack, which exploits payment griefing to paralyse PCNs. We develop concrete steps for launching Bankrun attacks with various strategies. Based on our framework on quantifying PCNs' liquidity, we evaluate Bankrun attacks on Bitcoin's Lightning Network – the first and most well-known PCN, and show that Bankrun attack is cheap to launch and can greatly reduce the Lightning Network's liquidity.

References

1. Bitcoin Average Transaction Fee. `https://ycharts.com/indicators/bitcoin_average_transaction_fee` (2020), [Online; accessed 20-September-2020]
2. BitInfoCharts. `https://bitinfocharts.com/top-100-richest-bitcoin-addresses-1.html` (2020), [Online; accessed 20-September-2020]
3. Bitrefill. `https://www.bitrefill.com/` (2020), [Online; accessed 20-September-2020]
4. Coinbase.com. 2020. Bitcoin Price Chart (BTC). `https://www.coinbase.com/price/bitcoin` (2020), [Online; accessed 20-September-2020]
5. Akash Khosla, E.S., Hope-Bailie, A.: Interledger rfcs, 0018 draft 3, connector risk mitigations. `http://j.mp/2m2OvfP` (Github, 2019)
6. Beres, F.: Lightning Network data. `https://dms.sztaki.hu/~fberes/ln/ln_data_2019-10-29.zip` (2020), [Online; accessed 20-September-2020]

7. Béres, F., Seres, I.A., Benczúr, A.A.: A cryptoeconomic traffic analysis of bitcoins lightning network. arXiv preprint arXiv:1911.09432 (2019)
8. Dandekar, P., Goel, A., Govindan, R., Post, I.: Liquidity in credit networks: A little trust goes a long way. In: Proceedings of the 12th ACM conference on Electronic commerce. pp. 147–156 (2011)
9. Diamond, D.W., Dybvig, P.H.: Bank runs, deposit insurance, and liquidity. Journal of political economy **91**(3), 401–419 (1983)
10. Ford Jr, L.R., Fulkerson, D.R.: Flows in networks, vol. 54. Princeton university press (2015)
11. Goldschlag, D., Reed, M., Syverson, P.: Onion routing. Communications of the ACM **42**(2), 39–41 (1999)
12. Hagberg, A., Swart, P., S Chult, D.: Exploring network structure, dynamics, and function using networkx. Tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States) (2008)
13. Harris, J., Zohar, A.: Flood & loot: A systemic attack on the lightning network. arXiv preprint arXiv:2006.08513 (2020)
14. 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. In: Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security. pp. 602–612 (2019)
15. Hoffmann, C.: Algorithms and complexity for centrality improvement in networks (2017)
16. Mizrahi, A., Zohar, A.: Congestion attacks in payment channel networks. arXiv preprint arXiv:2002.06564 (2020)
17. Pérez-Sola, C., Ranchal-Pedrosa, A., Herrera-Joancomartí, J., Navarro-Arribas, G., Garcia-Alfaro, J.: Lockdown: Balance availability attack against lightning network channels (2019)
18. Poon, J., Dryja, T.: The bitcoin lightning network: Scalable off-chain instant payments (2016)
19. Ramseyer, G., Goel, A., Mazières, D.: Liquidity in credit networks with constrained agents. In: Proceedings of The Web Conference 2020. pp. 2099–2108 (2020)
20. Robinson, D.: Htlcs considered harmful. In: Stanford Blockchain Conference (2019)
21. 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&PW). pp. 347–356. IEEE (2019)
22. Tochner, S., Schmid, S., Zohar, A.: Hijacking routes in payment channel networks: A predictability tradeoff. arXiv preprint arXiv:1909.06890 (2019)

A Detailed evaluation results and analysis

Larger payments are less likely to succeed both before and after the attack. This is because if a payment's amount is bigger, fewer channels can route this payment. Also, the success rate of large payments drops more than small payments after the attack. This is because our attack starts from hubs and mostly influences channels with large capacity.

In addition, rank-by-length is more effective than the other four strategies in terms of the success rate. Specifically, rank-by-length can help underfunded

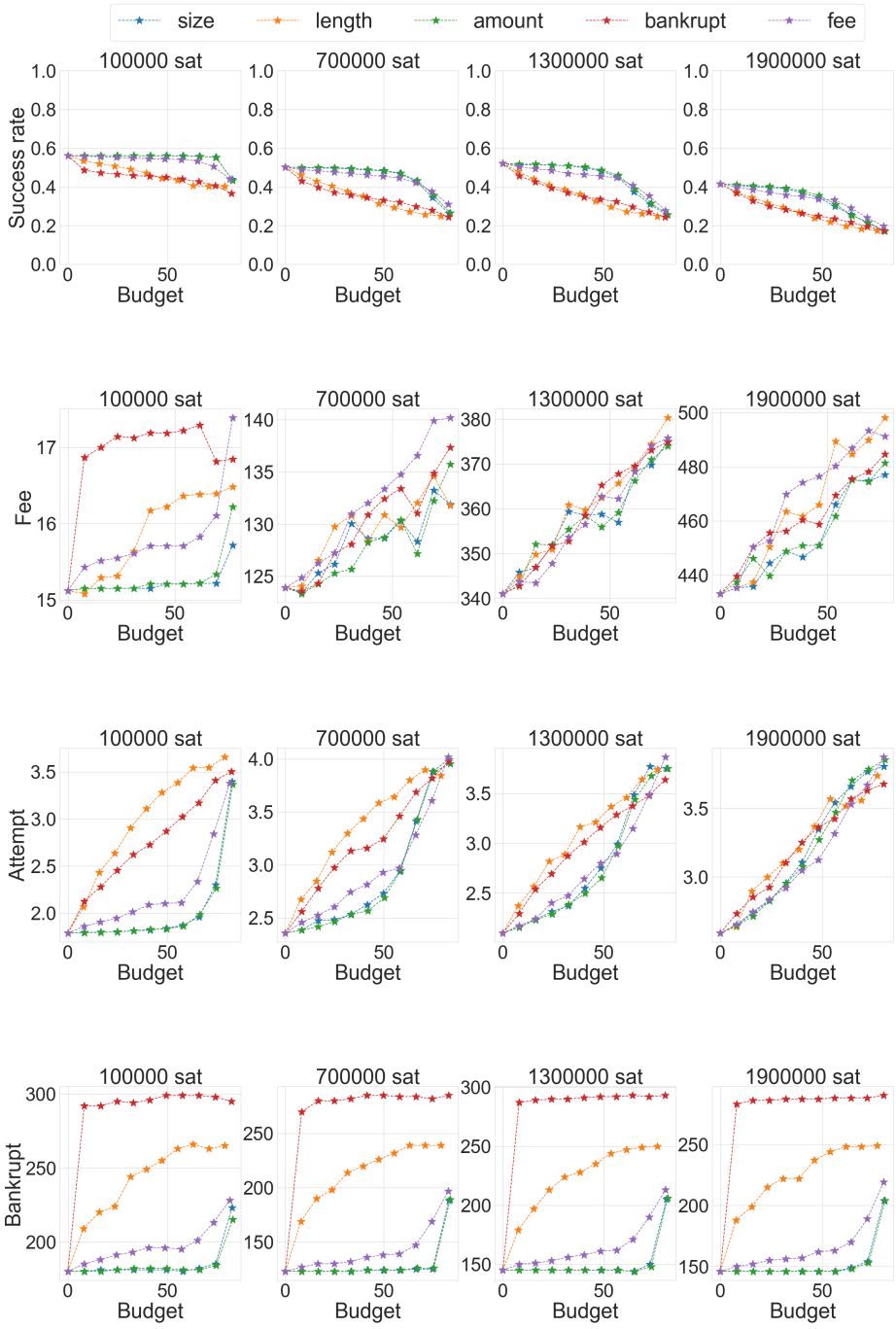


Fig. 8: Overview impact of all strategies.

adversaries get an additional 2%-19% net reduction in the success rate. This is because the adversary's balance is limited in our setting. As discussed in Section 3.5, when the adversary's balance is limited, he can lock most coins in the network by starting from griefing long payments.

The second row shows the average fee of survived payments after the attack. Note that for each figure, a payment is treated as survived if it is successful after the attack with any budget and any strategy. The result shows that, the fee slightly rises with the budget of the attack increases.

Overall, the attack using with the rank-by-length strategy results in most fee rise, and rank-by-bankrupt has a bigger impact on micropayments. For payments with 1,900,000 satoshi, five strategies result in similar fee rise. For payments 1,300,000 satoshi, the rank-by-length strategy results in most fee rise. For payments with 100,000 and 700,000 satoshi, rank-by-bankrupt and rank-by-fee strategy result in most fee rise.

The reason that the rank-by-length strategy achieves most fee rise is that griefing long payments influences most channels and is likely to force many payments to change paths. This can be more obvious on small payments, as small payments are more sensitive to changes of channels' balances.

Similar to success rate and fee, rank-by-length is more effective than the other four strategies in terms of the average attempt time. The reason is also that the rank-by-length strategy is most effective when the budget is limited. The advantage of rank-by-length is more obvious on small payments, as small payments are more sensitive to changes of channels' balances.

In addition, the average attempt time of payments with 100,000 satoshi is always much fewer than payments with a bigger amount. We suspect this is because payments with 100,000 satoshi have much fewer "deceptive" channels. We say a channel is deceptive for a payment with amount x when the capacity of the channel is greater than x, but the balance is less than x.

Table 1: The proportion μ of deceptive channels out of channels with capacities greater than the payment amount. $\bar{a}$ is the average attempt time before the attack.

	100,000	700,000	1,300,000	1,900,000
μ	15.0%	31.8%	31.2%	39.2%
$\bar{a}$	1.76	2.59	2.66	3.11

Due to deceptive channels, the payer usually needs to attempt multiple paths for making a payment. The attempt time depends on the ratio between the number of deceptive channels and the number of channels with capacities greater than the payment amount. As aforementioned, we assume the balance of each channel is uniformly distributed. Under this assumption, given the payment's amount x, this proportion μ is calculated as

$$\mu = \frac{\sum_{i=1}^{n} \frac{x}{c_i}}{n}$$

where n is the number of channels with capacities greater than x, and c_i is the i-th channel's capacity. Table 1 shows the trend of μ and $\bar{a}$ is consistent, which confirms our suspicions.

About bankruptcy, rank-by-bankrupt has a significant advantage over other strategies.

Rank-by-amount and rank-by-size are the worst performers, because the former essentially attacking nodes that are rich in capacity, which is contrary to Equation2. And rank-by-size exhibits similar behaviour to rank-by-amount since the variance of the length of the griefing paths we enumerate is small.

B Analysis of impact on fee of payments

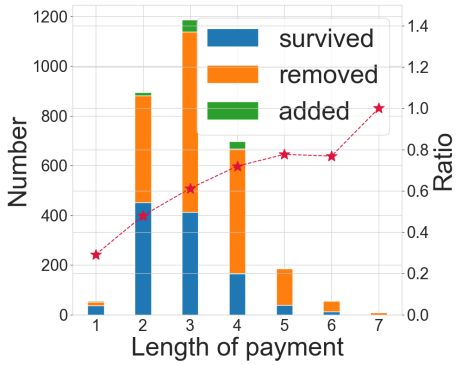


Fig. 9: Status of payments (1,300,000 satoshi) after the attack with rank-by-length strategy. Red line indicates the percentage of removed payments.

We use the scenario where the payment amount is 1,300,000 satoshi as an example to show why we only count fee of *survived*. The result (in Figure 9) shows that, long payments are more likely to be influenced by our attack. Specifically, all 7-hop payments are killed by the attack, while only about 30% of 1-hop payments are affected. This is because a long payment indicates that there are more channels in the path, so the payments are more likely to be attacked.

Interestingly, some payments are *added*. Two scenarios that cause *added* payments. The first scenario is that, consider two payments $P_1 = A \to B \to C \to D$ and $P_2 = F \to C \to D$. Before attacking, P_1 will exhaust channel CD, and P_2 will fail. The attack will drain AB, and P_1 will fail. In this way, CD can still route P_2, and P_2 will be successful. The second scenario is that, the payment

$A \to B \to C$ is forced to find another path $A \to E \to C$ as the attack drains AB, so that BC can route other payments.

By increasing the fee in the lightning network, we can provide a favourable environment for some attacks. For example, Saar et al. [22] propose to attract and hijack payments in the network by setting the fee of his channels to 0. The premise of the attack is that the victim believes that the path through the adversary is more economical. The Bankrun attack can therefore be used to increase the fee across the network, making the adversary's channel more competitive.

C Analysis of rank-by-fee

To increase average fees in PCN, an intuitive idea is to prioritize attacking those channels with cheap fees. Based on the idea, we can obtain weight as follows

$$Score_P = \frac{\sum_{i=0}^{len(P)-1} F_{P_i}}{len(P) - 1} \tag{3}$$

where P is the path of griefing payment, P_i represents the fee charged by the channel between the i-th and i+1th nodes of the path. Then, we experiment with rank-by-length and rank-by-fee strategies under the payment amount of 100,000 satoshi.

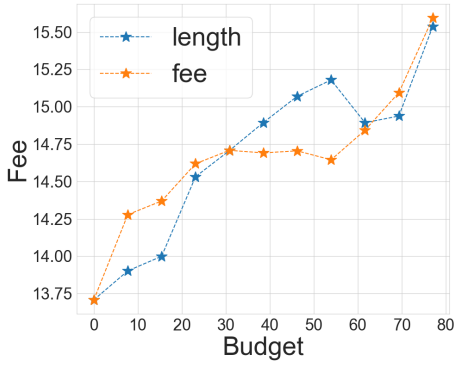


Fig. 10: Rank-by-fee and rank-by-length comparison.

We find that the rank-by-fee strategy does not show a significant advantage when compared to rank-by-length. This is because the fee for a payment is influenced by two factors, the length of the payment path and the average fee of the channels on the payment path. Equation 3 consider the latter. Specifically, the average length of a survived payment under the rank-by-length strategy

is 4.5, while each channel charges 3.5 satoshi. Rank-by-length has an average length of only 4.3, but the average channel charges a fee of 3.7 satoshi. To get a significant increase in fees, we need to design a metric that takes into account both length and fee, which will be our future work.