Bank run Payment Channel Networks

Zhichun Lu[1], Runchao Han[2,3], and Jiangshan Yu[2⋆]

[1] Zhejiang Gongshang University, China
[2] Monash University, Australia
[3] CSIRO-Data61, Australia

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 identified attacks on PCNs' liquidity, where the payee withholds the preimage in Hash Time Locked Contract. Before this 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 attacker generates sybil nodes, establishes channels with hubs in the network, makes payments between his nodes and griefs them simultaneously. If the adversary has sufficient coins, he can lock a high percentage of coins in the PCN, so that the PCN may no longer handle normal payments.
We introduce a framework for launching Bankrun attacks, and develop three strategies with a focus on minimising the cost, draining important channels, and locking most amount of coins, respectively. We evaluate the effectiveness of Bankrun attacks on Bitcoin's Lightning Network, the first and most well-known PCN. Our evaluation results show that, using channels with 1.5% richest nodes, the attacker can lock 83% of the capacity in the entire network. With connections to these nodes, an adversary with 13% (∼77 BTC) of coins in the network can lock up to 45% (∼ 267 BTC) of coins in the entire network until time out (e.g. for an entire day); reduces the success rate of payments by 23.8%∼62.7%; increases fee of payments by 3.5%∼14.0%; and increases average attempts of payments by 26.4%∼113.7%, where payments range from 100,000 to 1,900,000 satoshi (7∼135 USD).

1 Introduction

Since Bitcoin, blockchains have achieved great success. However, they suffer from limited throughput. Introduced by Lightning network (LN) [11], Payment Channel Network (PCN) is one of the promising approaches to scale blockchains.

Payment channels allow nodes to make offchain payments where transactions do not need to be recorded on the blockchain. To open a payment channel, two parties collaterise some of their coins in a 2-2 multi-signature address. To make

⋆ corresponding author

a payment, two parties only need to sign a new transaction which updates their balance in this channel. To close the channel, one of the two parties commits the latest transaction to the blockchain to commit their final balance on the chain. Without a direct channel, two parties can make off-chain payments through multiple channels, and such payments are known as *multi-hop payments*. If a party wants to make a payment, he should find a group of channels (i.e., a path) that can direct him to the payee. The network of channels is called a Payment Network Channel (PCN). A multi-hop payment should update involved channels in an atomic way, which usually relies on Hash Time Locked Contracts (HTLCs). HTLC is a contract that, the payee should reveal the preimage of a hash value before timeout to redeem the payment, otherwise the payment will expire. In a multi-hop payment, the payee chooses a preimage and computes its hash value, and HTLC payments of all involved channels share this hash value. By revealing this preimage, all HTLC payments are activated simultaneously.

Payment griefing [2] is an attack that exploits HTLC to reduce PCNs' liquidity i.e., the ability of a PCN to route payments. In payment griefing, the attacker withholds the preimage of HTLCs. Before the payment expires, coins involved in this payment are locked and cannot be used in other payments. The attack is free, as for an unsuccessful payment the payer does not need to pay for anything. In addition, this attack is not accountable as intermediate nodes does not know who the payer is, due to the anonymity nature of PCNs.

In this paper, we introduce **Bankrun attack**, which exploits payment griefing to bank run the entire PCN. Bank run [4] is a concept in finance, where numerous clients withdraw their money from a bank simultaneously, as they lose confidence to this bank due to events such as financial crisis. If the bank does not have enough money in hand (i.e., is insolvent), the bank will be short of liquidity and eventually go bankrupt. The key of bank run is initiate a large number of withdraw requests to paralyse the bank, which is similar to our attack. In a Bankrun attack in LN, the adversary generates a number of sybil nodes, establishes channels with existing nodes in the PCN, initiates numerous multi-hop payments between his nodes and griefs them simultaneously. If the adversary has sufficient coins, he can lock a high percentage of coins in the PCN, so that the PCN may no longer handle normal payments.

We introduce a framework for launching Bankrun attacks, which consists of four critical components, namely 1) node selection, 2) payment enumeration, 3) payment ranking, and 4) attack. For node selection, the adversary selects a subset of nodes in the PCN and establishes payment channels with them. We choose richest nodes (aka. hubs) in the PCN, as they have more capacity in relying payments, have connections to large number of nodes, and are capable to route more payments. For payment enumeration, the adversary enumerates all viable payments between his nodes. We employ Breadth First Search (BFS) for finding viable paths, and Ford-Fulkersons Algorithm [5] for calculating the maximum amount that each path can afford. For payment ranking, the adversary orders the enumerated payments in three ranking strategies according to the attack objectives, namely 1) order-by-amount for breaking the network "backbone"

channels (i.e., channels with most capacities), 2) order-by-length for maximising the locked coins while minimising the attack budget, and 3) order-by-size (the multiplication of amount and length) for maximising the locked coins regardless of the budget. During an attack, the adversary griefs all payments simultaneously. As we cannot know the balance of each channel, a payment may fail. In this case we retry the payment by gradually decreasing its amount.

We then evaluate the effectiveness of Bankrun attacks on Bitcoin's Lightning Network [11] as an example. The effectiveness consists of three aspects, namely the dropped success rate, the increased fee, and the increased attempts of payments. For ethical concern, the experiment is conducted in a simulated network. Our results show that, with channels established with 1.5% richest nodes, the attacker can occupy 83% capacity of the entire network. The payment griefing attack is cheap to launch, and can significantly damage the liquidity of PCNs. In particular, with 13% (∼77 BTC) coins in the network for making payments (which will be refunded as payments will not be successful) and negligible coins for opening payment channels, the attacker can launch a Bankrun attack that locks 45% (∼267 BTC) coins in the network; reduces success rate of payments by 33.3%∼71.4%; increases fee of payments by 3.5%∼14.0%; and increases average attempts of payments by 26.4%∼113.7%, where the amounts of payments range from 100,000 to 1,900,000 satoshi (7∼135 USD[4]).

The main contributions are summarized as follows:

- We introduce Bankrun attack on PCNs.
- We provide a framework for launching Bankrun attacks.
- We evaluate Bankrun attacks on Bitcoin's Lightning Network as an example.
- We show that Bankrun attacks are cheap and can significantly damage the liquidity of PCNs.

Roadmap. §2 provides the background of the proposed attack. §3 describes the security model. §4 describes Bankrun attack step by step. §5 describes how we evaluate the attack's effectiveness and provides the evaluation results and analysis. §6 reviews relevant literatures, and §7 concludes this paper.

2 Background

2.1 Payment Channel Networks

Lightning Network [11] 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 collaterise 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 collaterised coins. Before closing the payment channel,

[4] Bitcoin price was fetched from `https://www.coinbase.com/price/bitcoin` at 09/04/2020.

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.

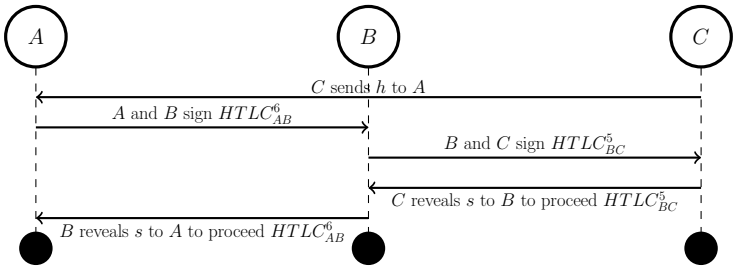


Fig. 1: A multi-hop payment from A to C via an intermediate node B.

The payment channel can be further extended to support offline payments that go through multiple channels (aka. multi-hop payments). 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 in order to redeem 5 BTC in $HTLC^5_{BC}$ from B. In the meantime, B is able to 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. In our example, the timelock of AB is 24 hours, and is one hour longer than BC.

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 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 least fee when making payments.

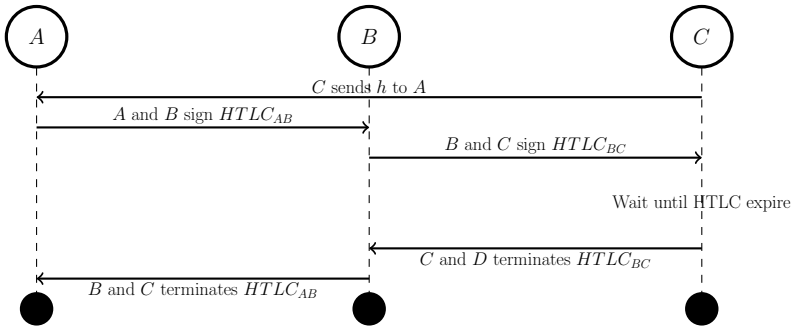


Fig. 2: Payment griefing attack.

2.2 Payment Griefing

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 on 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 [6], where each intermediate node only has the knowledge of nodes who directly connect with him.

3 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 its value.

At the beginning, the 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[5], purchasing capacity from existing nodes is easy and cost-effective.

[5] `https://www.bitrefill.com/`

For simplicity, we does 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 as the number of channels the payment involves, the amount of a multi-hop payment as the amount of coins that the payer wants to pay to the payee, and the size of a multi-hop payment θ as

$$\theta = X \cdot l$$

where X and l are the amount and the length of the payment. Given a payment of size θ, there are θ coins locked in total. Before this payment finishes or expires, these θ coins cannot be used for routing other payments.

4 Bankrun attack

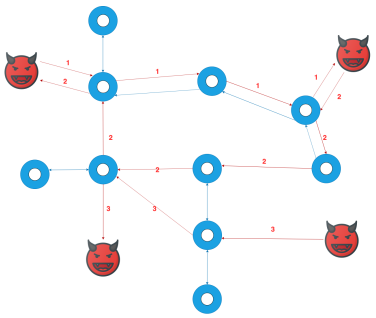


Fig. 3: Bankrun attack. The adversary generates some sybil nodes (red), establishes payment channels with existing nodes (blue) in the network, and makes payments between his own nodes and griefs these payments.

We introduce **Bankrun attack** (shown in Figure 3), an attack that 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 with bank-run [4] 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 simultaneously. Bank-run on both banks and PCNs leads to liquidity risk [4], 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 of 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 four critical components, namely 1) Node selection, 2) Path finding, 3) Path ranking, and 4) Launching attack.

1. **Node selection**: The adversary chooses a set of nodes and establishes channels with them.
2. **Enumerating payments**: The adversary enumerates all payments between his nodes.
3. **Ranking payments**: The adversary orders these payments.
4. **Launching attack**: The adversary starts to make and abort payments with this order.

§4.1-4.4 will describe how to conduct these steps in detail.

4.1 Node selection

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

To answer the first question, we suggest to establish 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 §5.1 we will show that the adversary only needs to establish channels with top 1.5% (42) richest nodes if attacking Bitcoin's Lightning Network.

4.2 Enumerating payments

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 §4.4), the adversary will try to make these payments by gradually decreasing the payment amounts.

Algorithm 1 Enumerating payments.

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

$T \leftarrow []$
for (n_1, n_2) in $\mathcal{N}$ **do** ▷ Start from richest nodes
  $path\_list \leftarrow \mathsf{BFS}(\mathcal{G}, n_1, n_2)$
  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]
    $amount \leftarrow min(capacity\_list)$ ▷ Get the most viable amount of *path*
    if $amount = 0$ **then continue** ▷ This path is not viable
    $P[amount] \leftarrow amount$ ▷ P is a viable payment
    Append P to $\mathcal{P}$
    Consume P in $\mathcal{G}$ ▷ In simulated environment
  end for
end for
return $\mathcal{P}$

Our payment enumerating algorithm borrows the idea of Ford-Fulkersons Algorithm [5] - 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 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 with 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.

4.3 Ranking payments

Griefing different payments may have different impacts on a PCN's liquidity. When the adversary's balance is limited, he should start from griefing critical payments in order to maximise the attack's effectiveness. Thus, the adversary should rank payments by their importance. We consider three ranking criteria, namely 1) the length l, 2) the amount X, and 3) the size θ. These three criteria have different preferences on the attack effectiveness.

Order by size θ. This strategy is for maximising the amount of coins locked in the PCN, without considering the balance of the adversary. For each payment, the adversary can lock as many coins as the size θ of this payment.

Order by amount X. This strategy is for attacking most important channels in the PCN. Payments with large amounts involve channels with large amounts. These channels can be important and treated as the "backbone" of the entire PCN, as they may route numerous payments. Thus, attacking these important channels can make all payments relying on these channels to fail.

Order by length l. This strategy is for maximising the amount of coins locked in the PCN while minimising the cost. Given a fixed payment size, the amount will be smaller if the payment involves more channels. Thus, when the expected cost of the adversary is limited, starting from griefing long payments can maximise the amount of coins locked.

4.4 Launching attack

Algorithm 2 Launching attack.

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

When enumerating payments, we use channels' capacities rather than their real-time balances for determining the amounts of payments. Thus, some of our 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 with Joancomarti et al. [8] 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. [8] using binary search, we search for feasible amounts, from largest possible to zero. This is because we aim at making payments successfully with largest amounts, rather than revealing channels' balances with fewest attempts.

5 Evaluation

In this section, we evaluate Bankrun attack with three strategies in §4.3 on Bitcoin's Lightning Network. Our results show that, the adversary can greatly paralyse the entire PCN by launching a Bankrun attack. In particular, to launch the attack, the adversary needs to use ∼77 BTC to establish payment channels with 42 richest nodes in the network. The attack will lock 45% (∼ 267 BTC) of balance of the entire network; reduce the success rate of most payments by 22% - 62% (compared to their success rate without the attack); increase average attempt times of making payments by 26.4%∼113.7%; and slightly increase fees of payments. According to [1], there are more than 10,000 addresses that have more than 77 BTC and are able to launch payment griefing attacks.

5.1 Evaluating the attack's effectiveness

Similar to Béres et al. [3], we use a batch of n payments, of which payers and payees are random and the amount x_t is fixed, to test the attack's effectiveness.

Note that the focus of the attack may be different: some attackers aim at discouraging small payments, while other attackers aim at discouraging large payments. To this end, we test multiple batches of payments with different amounts (which will be discussed later).

We then simulate these payments in the PCN, both before and after the attack. 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.

There are three situations for payments, namely *added*, *survived*, or *removed*. *Added* means the payment is failed before attack but is successful after attack. *Survived* means the payment is successful both before and after attack. *Removed* means the payment is successful before attack but is failed after attack.

Bankrun attacks aim at reducing the liquidity of PCNs. The liquidity of PCNs consists of three aspects, namely 1) the success rate of payments, 2) the average fee of payments paid to intermediate nodes, and 3) the average attempt time of payments.

Success rate p. The drop of success rate of payments is a metric quantifying the attack's effectiveness. After the attack, payments will be less likely to succeed. We define the success rate p of payments with that amount as

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

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

Experimental setting We simulate and implement our attack using python 3.7.4 and NetworkX [7] - a Python library for complex networks. All experiments run on a macOS with Catalina 10.15.3 operating system, a Intel Core i5-CPU 2.4G Hz CPU, and 8GB of memory. Similar to Béres et al. [3], 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 [3].

We test attacks with three ranking criteria in §4.3, and $step = 0.1 * amount$ in Algorithm 2. We test attacks with different levels of budgets of the adversary, including $7.7, 15.4, \ldots, 77$ BTC. To evaluate the effectiveness using the method in §5.1, we pick $n = 7,000$, $x_t = \{100,000, 700,000, 1,300,000, 1,900,000\}$ (satoshi), and $r = 10$. In particular, we use four batches of payments with different amounts (100,000, 700,000, 1,300,000, and 1,900,000 satoshi). The range of amounts covers most scenarios using PCNs: 100,000 satoshi is approximately 7 USD, and 1,900,000 is approximately 131 USD. Each batch consists of 7,000 payments, which is similar to Béres et al. [3]. 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.

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 number 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 entire network. In addition, the sum of sizes of enumerated payments converges with the percentage

[6] `https://dms.sztaki.hu/~fberes/ln/ln_data_2019-10-29.zip`

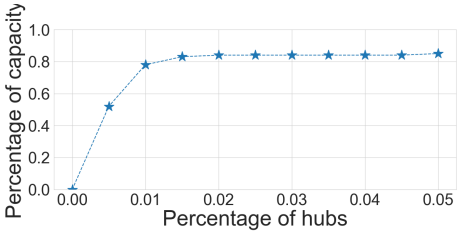
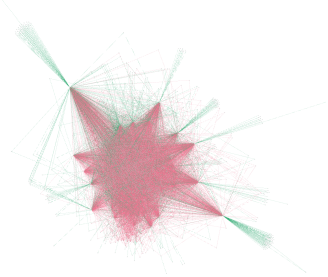
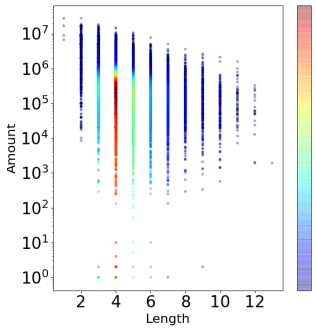


Fig. 4: The relationship between the percentage of hubs that the adversary establishes channels with (x axis) and the percentage of the capacity of the entire network that the enumerated payments take (y axis).

of hubs increasing. Thus, if attacking Bitcoin's Lightning Network, we suggest the adversary establishing channels only with top 1.5% richest nodes.



(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.

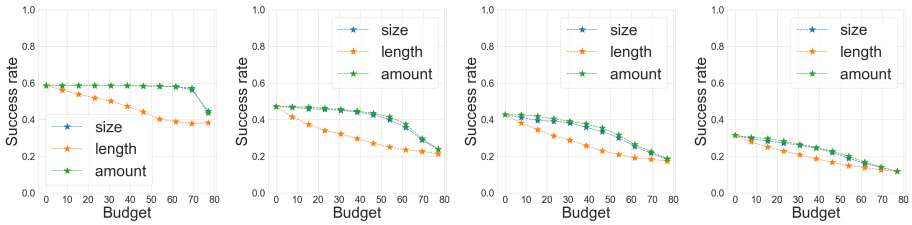
Fig. 5: Characterisation of enumerated payments.

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 entire 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. In addition, most payments are with the length of 3∼6 and with the amount of $10^3 \sim 10^6$ satoshi. This confirms that our payment enumeration algorithm (Algorithm 1) is effective.

5.2 Simulation results

Payment success rate Figure 6 shows the impact of our payment griefing attacks on the success rate of payments with different amounts. It shows that, the success rate of payments significantly drops due to the attack. In particular,

- Success rate of payments with 100,000 satoshi ($\sim$ 7 USD) drops approximately from 58% to 40% (by 31.0%).
- Success rate of payments with 700,000 satoshi ($\sim$ 50 USD) drops approximately from 47% to 22% (by 51.1%).
- Success rate of payments with 1,300,000 satoshi ($\sim$ 93 USD) drops approximately from 42% to 18% (by 57.1%).
- Success rate of payments with 1,900,000 satoshi ($\sim$ 135 USD) drops approximately from 31% to 12% (by 61.3%).



(a) 100,000 satoshi (b) 700,000 satoshi (c) 1300,000 satoshi (d) 1,900,000 satoshi

Fig. 6: Success rate of payments after attacks.

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, order-by-length is more effective than the other two strategies in terms of the success rate. Specifically, length priority can help underfunded attackers get an additional 2%-19% net reduction in the success rate. This is because the attacker's balance is limited in our setting. As discussed in §4.3, when the attacker's balance is limited, he can lock most coins in the network by starting from griefing long payments.

Moreover, the success rate of payments is 100,000 satoshi remains stable when the budget is less than 69 BTC, but drops dramatically after that. Since starting from griefing payments with big sizes or amounts, the attacker might spend a great amount of coins, but only affect few channels.

We also evaluate the success rate of payments with different lengths after the attack. We use the scenario where the payment amount is 1,300,000 satoshi

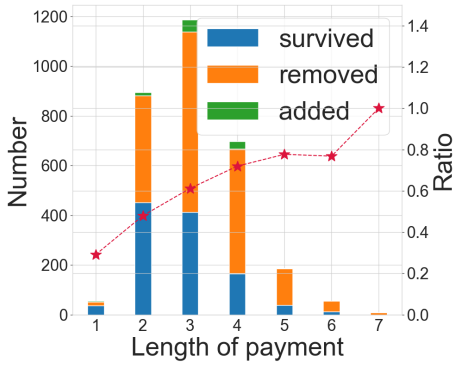


Fig. 7: Status of payments (1,300,000 satoshi) after the attack with 77 BTC and order-by-length strategy. Red line indicates the percentage of removed payments.

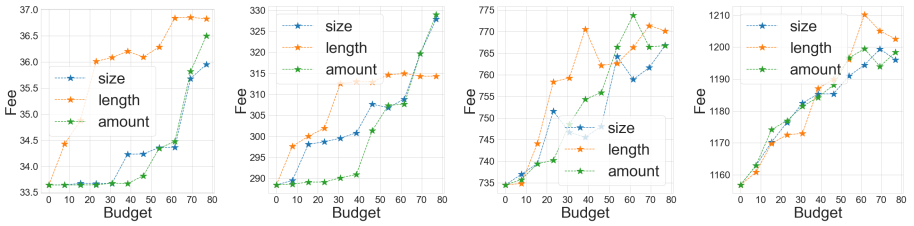
and the attack uses order-by-length strategy as an example. The result (in Figure 7) shows that, long payments are more likely to be influenced by our attack. Specifically, all 7-hop payments are killed by the attack, while only about 30% of 1-hop payment are affected. This is because a long payment indicates that the payer and the payee are far from each other, and draining a single channel between them may fail this payment.

Interestingly, some payments are *added*. There are 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.

Average fee of survived payments Figure 8 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 order-by-length strategy results in most fee rise, and its advantage is more significant on small payments. For payments with 1,900,000 satoshi, three strategies results in similar fee rise. For payments 1,300,000 satoshi, the order-by-length strategy results in most fee rise, while the other two strategies achieve less fee rise. For payments with 100,000 and 700,000 satoshi, the order-by-length strategy results in most fee rise, and the order-by-size strategy results in least fee rise.

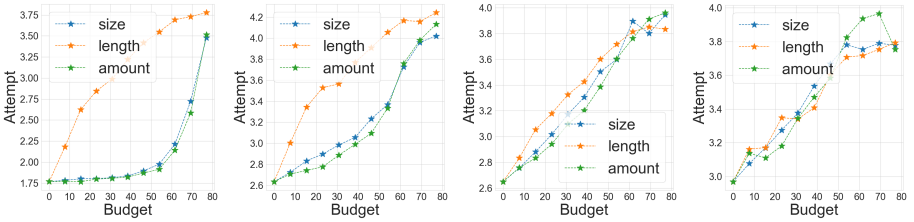
The reason that the order-by-length strategy achieves most fee rise is because griefing long payments influences most channels and is likely to force many payments to change paths. This can be more obvious on small payments, as small payments are more sensitive to changes of channels' balances.



(a) 100,000 satoshi (b) 700,000 satoshi (c) 1,300,000 satoshi (d) 1,900,000 satoshi

Fig. 8: Fee of survived payments after the attack.

Average attempt time of successful payments Figure 9 shows the impact of out attack on the average attempt time of successful payments. For payments with 100,000, 700,000, 1,300,000 and 1,900,000 satoshi, our attack forces the average attempt time to increase by 2, 1.5, 1, and 0.8, respectively.



(a) 100,000 satoshi (b) 700,000 satoshi (c) 1300,000 satoshi (d) 1,900,000 satoshi

Fig. 9: Time of attempts of successful payments after the attack.

Similar with success rate and fee, order-by-length is more effective than the other two strategies in terms of the average attempt time. The reason is also because the order-by-length strategy is most effective when the budget is limited. The advantage of order-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 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.

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

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 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

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

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 suspect.

6 Related work

To the best of our knowledge, Bankrun attack is the first practical attack on PCNs' liquidity. Dan Robinson [12] first discussed the griefing problem introduced by HTLCs, and Interledger RFCs [2] discussed payment griefing attacks in PCNs. Our Bankrun attack exploits payment griefing attacks to reduce the entire PCN's liquidity.

Mizrahi et al. [9] propose Congestion attack, where the adversary floods a PCN with numerous small payments so that nodes reach the maximum number of payments they can routes simultaneously. However, this maximum number is controlled by *max_concurrent_htlcs*, a parameter determined by the node himself. This can be mitigated simply by increasing the value of *max_concurrent_htlcs*. If a PCN's liquidity is bottlenecked by *max_concurrent_htlcs* rather than the collaterised coins, the adversary can also launch Bankrun attacks that exhaust *max_concurrent_htlcs* rather than channel capacities.

Rohrer et al. [13] discuss two attacks, namely Channel exhaustion and Node isolation. Channel exhaustion aims at exhausting channels' capacities by making payments through them. Node isolation aims at exhausting all channels of a node so that the node can no longer route payments. Pérez-Sola et al. [10] formalised Node isolation as Lockdown attack, and evaluated it on Bitcoin's Lightning Network. Our Bankrun attack aims at paralysing the entire PCN, while Node isolation and Lockdown attack aim at isolating individual nodes in the PCN.

7 Conclusion

In this paper, we propose Bankrun attack, which exploits payment griefing to bank run PCNs. We provide a framework with three different strategies for launching Bankrun attacks. We evaluate the effectiveness of Bankrun attacks on Bitcoin's Lightning Network as an example, and show that Bankrun attack is cheap to launch and can greatly reduce the Lightning Network's liquidity.

References

1. Top 100 richest bitcoin addresses. https://bitinfocharts.com/top-100-richest-bitcoin-addresses.html, bitinfocharts, 2020
2. Akash Khosla, E.S., Hope-Bailie, A.: Interledger rfcs, 0018 draft 3, connector risk mitigations. http://j.mp/2m2OvfP (Github, 2019)
3. Béres, F., Seres, I.A., Benczúr, A.A.: A cryptoeconomic traffic analysis of bitcoins lightning network. arXiv preprint arXiv:1911.09432 (2019)
4. Diamond, D.W., Dybvig, P.H.: Bank runs, deposit insurance, and liquidity. Journal of political economy **91**(3), 401–419 (1983)
5. Ford Jr, L.R., Fulkerson, D.R.: Flows in networks, vol. 54. Princeton university press (2015)
6. Goldschlag, D., Reed, M., Syverson, P.: Onion routing. Communications of the ACM **42**(2), 39–41 (1999)
7. 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)
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. In: Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security. pp. 602–612 (2019)
9. Mizrahi, A., Zohar, A.: Congestion attacks in payment channel networks. arXiv preprint arXiv:2002.06564 (2020)
10. Pérez-Sola, C., Ranchal-Pedrosa, A., Herrera-Joancomartí, J., Navarro-Arribas, G., Garcia-Alfaro, J.: Lockdown: Balance availability attack against lightning network channels (2019)
11. Poon, J., Dryja, T.: The bitcoin lightning network: Scalable off-chain instant payments (2016)
12. Robinson, D.: Htlcs considered harmful. In: Stanford Blockchain Conference (2019)
13. 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)