SCOOP: CoSt-effective COngestiOn Attacks in Payment Channel Networks*

Mohammed Ababneh, Kartick Kolachala, Roopa Vishwanathan

New Mexico State University, Las Cruces, NM, USA

mababneh@nmsu.edu, kart1712@nmsu.edu, roopav@nmsu.edu

June 24, 2025

Abstract

Payment channel networks (PCNs) are a promising solution to address blockchain scalability and throughput challenges, However, the security of PCNs and their vulnerability to attacks are not sufficiently studied. In this paper, we introduce SCOOP, a framework that includes two novel congestion attacks on PCNs. These attacks consider the minimum transferable amount along a path (path capacity) and the number of channels involved (path length), formulated as linear optimization problems. The first attack allocates the attacker's budget to achieve a specific congestion threshold, while the second maximizes congestion under budget constraints. Simulation results show the effectiveness of the proposed attack formulations in comparison to other attack strategies. Specifically, the results indicate that the first attack provides around a 40% improvement in congestion performance, while the second attack offers approximately a 50% improvement in comparison to the state-of-the-art. Moreover, in terms of payment to congestion efficiency, the first attack is about 60% more efficient, and the second attack is around 90% more efficient in comparison to state-of-the-art.

*A preliminary version of this work appears in Proceedings of the 6th Workshop on Coordination of Decentralized Finance (CoDecFin) 2025.

1 Introduction

Cryptocurrencies, the most popular blockchain-based application, enable users to transfer money securely and efficiently without relying on centralized authorities such as banks or governments. However, despite their increasing popularity, scalability remains a significant challenge. For instance, Bitcoin blockchain generates a 1MB block every 10 minutes, processing only 7 transactions per second, while users have to wait around one hour (i.e., six blocks) for the transaction to be confirmed [7]. In contrast, Visa handles 24,000 transactions per second [18].

Payment channel networks (PCN) [16] have emerged as an off-chain solution to the scalability challenge, such as Bitcoin's Lightning Network (LN) [14] and Ethereum's Raiden Network [15]. The main component of a PCN is a payment channel, where two nodes establish a channel to conduct transactions. Nodes not directly connected with a channel can route transactions through intermediate nodes using multiple hops, extending the network and enabling transactions across the entire PCN. One of the main advantages is that PCNs do not require access to the blockchain every time a transaction occurs (unless there is a dispute). Additionally, PCNs do not require any additional consensus mechanism other than the one employed by the underlying blockchain, and do not require complex cryptographic machinery to process transactions. This offers the advantage of enabling transactions to be executed over the PCN with relatively low latency and high throughput.

Despite the advantages mentioned above, PCNs are vulnerable to attacks. Several types of attacks have been proposed such as griefing [4], LockDown [13], congestion [9, 11], wormhole attack [10], Flood and Loot attack [3] and more. The common goal of these attacks is to exploit the inherent limitations and characteristics of the PCN to either gain financial advantage, collect information about the network, or disrupt its operation. Thus, these attacks can harm the network's efficiency, reliability, and user experience. Nonetheless, it is important to study and propose new attacks in PCNs as they are useful in drawing attention to network weaknesses and limitations. It will prompt researchers to design corresponding attack mitigation techniques, thus improving overall network security.

This paper focuses on congestion-based attacks, in which the attacker node issues a payment contract with a specific payment amount to another attacker node over paths involving intermediate nodes.[1] As the attacker

[1] As a part of ethical considerations for vulnerability disclosure, we have informed Lightning Labs [6] about the attacks described in this paper. Lightning Labs is currently reviewing our attacks and the corresponding results. We also present a few mitigation strategies for the attacks in the paper.

manages both the sender and receiver of the payments, they can intentionally postpone executing the payments, intensifying the congestion in PCNs. These attacks aim to congest payment paths (constituting channels), reducing the network's ability to process legitimate further payments. While previous studies have explored various aspects of congestion attacks, an important consideration that needs to be considered is congestion attack efficiency. In this work, we aim to address this gap by examining how the attackers can optimize their strategies to congest the network more effectively, given limited resources. In particular, we investigate how, in PCN, multiple attacker nodes with predefined budgets can allocate path payments to congest the network efficiently.

To address this problem, several factors must be considered, including the attacker's resources, i.e., attack budget, payment path information, e.g., lengths, capacity, and congestion performance. We propose several metrics to quantify the congestion performance more effectively. Using these metrics, we formulate congestion attacks as linear optimization problems. In the first formulation, our goal is to minimize the total amount of payments allocated by the attacker over the different available paths to achieve a threshold congestion performance. In the second formulation, given an attack budget, the attacker aims to maximize the congestion performance over the different payment paths. In essence, both problems aim to provide a mathematical formulation of efficient resource-limited congestion attacks. To summarize, the contributions of our work are as follows:

1. We introduce novel metrics that accurately quantify the impact and effectiveness of congestion attacks on PCNs. These metrics provide a more precise evaluation of congestion severity across multiple dimensions, including channel capacity and path length.

2. We present SCOOP, a comprehensive framework that includes two innovative congestion attack strategies on PCNs. These attacks are formulated as linear optimization problems, providing a mathematical framework for designing efficient, resource-limited congestion attacks. Attackers can either minimize budget expenditure while achieving a desired congestion threshold or maximize the overall congestion impact across the network under predefined budget constraints.

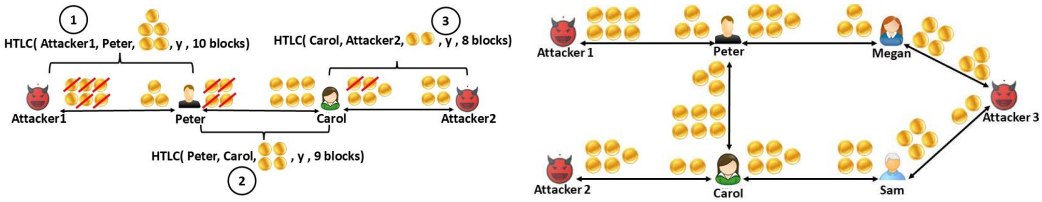
3. We evaluate the performance of SCOOP through extensive simulations of a PCN. The simulations validate the effectiveness of the proposed congestion attacks compared to existing strategies, demonstrating superior resource efficiency and congestion performance.

Outline: In Section 2, we discuss necessary preliminary information. In Section 3, we discuss the relevant related work. In Section 4, we present our congestion attacks. In Section 5, we present our experimental evaluation. In Section 7 we conclude the paper.

2 Background

2.1 Congestion Attack

The LN is susceptible to congestion attacks [9, 11]. In a congestion attack, the attacker adds Sybil nodes to PCN by establishing a payment channel with existing nodes. Subsequently, The attacker initiates numerous simultaneous payments across multiple paths to either lock capacity along all paths or initiates tiny payments to occupy available HTLC slots along a path and withholds the preimages of the payments between the Sybil nodes. This locks coins across paths, preventing intermediate nodes from earning fees and disrupting their operations. Congestion attacks are cost-effective for the attacker. The attacker incurs on-chain fees for opening and closing channels but avoids routing fees as payments remain incomplete. The congestion attacker aims to reduce LN throughput, cause transaction failures, and eliminate competition by blocking competing nodes and redirecting traffic and fees to their own nodes.



(a) (b)

Figure 1: (a) Channel Capacity Limitation. (b) Congestion Attack

In the congestion attack shown in Figure 1b, we have three Sybil nodes (Attacker1, Attacker2, and Attacker3). For simplicity, we identified a subset of paths between these Sybil nodes are identified as follows: Attacker1 → Peter → Megan → Attacker3, Attacker1 → Peter → Carol → Attacker2, Attacker2 → Carol → Sam → Attacker3, and Attacker3 → Sam → Carol → Attacker2. Each attacker generates a payment and sends it along its respective path, with the receiving attacker intentionally withholding the payment execution.

In the case where two paths share a common payment channel, such as between Attacker1 → Peter → Carol → Attacker2, and Attacker3 → Sam → Carol → Attacker2, which both share the payment channel Carol → Attacker2, accurate channel usage is critical. To address this, each path is evaluated sequentially, and the capacities of the shared channels are updated after each evaluation. For example, if Attacker1 initiates payments over its paths, the capacity of shared channels, such as Carol → Attacker2, is updated to reflect the locked amounts before subsequent paths are considered. This approach ensures that no two paths simultaneously attempt to use the full capacity of a shared channel, thereby avoiding resource conflicts. The Hashed Time Locked Contracts (HTLCs) and their Limitations are described in in Appendeix D. The implications of congestion attacks on PCNs are described in the Appendix. C.

3 Related Work

In this section, we analyze various studies examining attacks on PCNs. Our review focuses on congestion and other related attacks, exploring how attackers can exploit network limitations, such as channel capacity and maximum accepted HTLCs, to disrupt payment flows and compromise network security. Drawing on our insights and observations from prior works, SCOOP proposes new congestion attacks and introduces new congestion attack metrics.

Mizrahi and Zohar [11] proposed the first congestion attack where the attacker uses a greedy algorithm to construct the paths by consecutively picking high-weighted edges (using channel capacity weight or betweenness centrality). Betweenness centrality measures the number of shortest paths passing through the edge, indicating its importance in the PCN. The attacker generates dust payments and sends them over a given path to hold all the available HTLCs, preventing the payment channel along the path from processing more payment transactions. The attacker targets one path at a time, but by repeating the attack across different paths, multiple paths eventually can be congested. In another work, Lu *et al.* [9] proposed a general congestion attack, where the attacker generates Sybil nodes and connects them to strategically chosen set of nodes, typically, nodes with high channel capacities to route more griefing payments. The attacker finds paths between all pairs of Sybil nodes, and employs the Ford–Fulkerson algorithm to find the maximum flow over each path. The attacker generates numerous payments and sends them over all paths simultaneously. This causes network congestion and disruption of network performance. In [4], the authors proposed a griefing attack, where the attacker refuses to respond to an HTLC, which

locks all funds along a path until the HTLC time expires. Pérez-Sola *et al.* [13] presented a lockdown attack that blocks an intermediate node in the multiple-path payment by depleting the capacity in all its channels, which affects its ability to participate in payment routing. Malavolta *et al.* [10] proposed a wormhole attack in which two intermediate nodes in a payment path collude to exclude other intermediate nodes from participating in the payment, thereby stealing the fees intended for those nodes. Harris and Zohar [3] proposed an attack called Flood & Loot, where the attacker controls source and receiver nodes. The source node initiates payments but denies HTLC fulfillment, forcing channel closures and on-chain claims. Jourenko *et al.* proposed Payment Trees [5], a payment protocol for PCNs. In this work, the authors construct a virtual channel between the sender and the receiver (on top of already existing payment channels), such that the intermediate node(s) along the payment path lock collateral. Any malicious behavior by the intermediate nodes, such as causing congestion or denial of service is punishable since the malicious nodes lose their collateral. This protocol requires modifications to the current mechanism in which off-chain payments are processed in LN and hence is not compatible with LN. In SCOOP, the payment mechanism we use for demonstrating congestion does not require any modifications for payment processing in LN.

In contrast to other methods, our approach, SCOOP, frames congestion as linear optimization problems, providing a more precise and resource-efficient attack strategy. This sets SCOOP apart from existing approaches by optimally allocating resources and targeting congestion at the path level rather than relying on heuristic or greedy algorithms [9, 11]. SCOOP introduces new metrics that consider channel capacity and path length, allowing for a broader attack surface encompassing multiple dimensions of the network's structure.

4 System Construction

In this section, we describe the assumptions and provide the necessary notations and definitions for the PCN and the attacker. We then present the metrics used to evaluate the effectiveness of congestion attacks. Additionally, we examine the implications of congestion attacks on PCNs and propose mitigation techniques to address the congestion attacks.

4.1 Network Model and Assumptions

A PCN can be modeled as a directed graph $G = (V, E)$ in which the set of vertices V represent network nodes (i.e., users) and E represents the set of directed edges (i.e., payment channels) between nodes. Assume nodes v_x, v_y $\in V$ have established a payment channel $e_{x,y} \in E$ between them. Let $b_{x,y}$ represent the balance from node x to y (i.e., how many coins can node x forward in the direction of node y). Note that the balances $b_{x,y}$ and $b_{y,x}$ are not necessarily equal. The capacity of the channel $e_{x,y}$ is denoted as $c_{x,y}$, where $c_{x,y} = b_{x,y} + b_{y,x}$.

The attacker deploys a set of N Sybil nodes denoted as $\mathcal{A} = \{A_1, A_2, \ldots, A_N\}$. The attacker nodes can attach to the PCN nodes using a number of different attachment strategies (e.g., random, highest degree, etc.). The attacks in SCOOP can be carried out regardless of the strategy. We now outline our key assumptions that enable the attacks. First, the attacker communicates with the Sybil nodes out-of-band, and any information gathered by the Sybil nodes is accessible to the attacker. Second, the attacker is aware of the entire network topology, including all of the payment channels (i.e., network graph) and their initial capacities and balances. This knowledge can be attained using probing protocols such as [11]. The probing path does not necessarily involve the network; it is sufficient to prob a number of paths between the attacker pairs. This enables performing a timely attack in which the channel capacities do not change significantly. Assuming that channel capacities undergo a significant change, attack payment will fail. The attacker node (sender) knows immediately that the change happened and thus can either start another prob or reduce the attack payment by updating values of the constraint in the optimization problem.

4.2 Congestion Attack Metrics

We now propose and describe metrics used to evaluate the effectiveness of SCOOP's congestion attacks. In particular, these metrics quantify the impact of the attack on the PCN. Our metrics are: i) Channel Congestion Ratio (CCR), ii) Path Congestion Ratio (PCR), iii) Scale Path Congestion Ratio (SPCR).

To define the metrics, consider two arbitrary Sybil nodes A_n and A_m and let P_j denote the j^{th} path (whose length is measured in terms of number of constituent channels) between them with the attack originating from node A_n. let $\mathcal{P} = \{P_1, P_2, \ldots, P_J\}$ represent all of the paths between all of the pairs of attackers A_n and A_m. Furthermore, let e_j^i denote the i^{th} channel along the path P_j. We assume A_n runs a probing algorithm. Probing is a technique

used by attackers to determine the balance of a payment channel. This is accomplished by sending probes, essentially fake payments whose HTLC digest(s) do not correspond to a valid preimage [1, 17]. Let us denote the corresponding channel capacity of channel i by c_j^i, and the estimated channel balance along the path P_j as b_j^i. Finally, let $\alpha_j^{n,m}$ denote the congestion payment made along path P_j from attacker A_n towards attaKer A_m. Having introduced the above quantities, we now define our performance metrics.

Definition 4.1. ***Channel Congestion Ratio (CCR)*** *is the ratio between the payment (originating from attacker A_n to attacker A_m) $\alpha_j^{n,m}$ along the path P_j and the balance of the i-th channel along that path (i.e., b_j^i)*

$$CCR_{\alpha_j^{n,m}}^{n,m} = \begin{cases} \alpha_j^{n,m}/b_j^i & \text{if } \alpha_j^{n,m} < b_j^i \\ 1 & \text{if } \alpha_j^{n,m} = b_j^i \end{cases} \tag{1}$$

The CCR quantifies the congestion effect of assigning a payment $\alpha_j^{n,m}$ to the channel e_j^i to process further payments. Naturally, the larger the locked payment $\alpha_j^{n,m}$ relative to the balance b_j^i, the more congested a channel is and the larger is the channel's CCR. A channel is fully congested when $\alpha_j^{n,m}$ equals its balance and therefore is assigned a value of 1.

Note that the CCR quantifies congestion over a single channel along a path. A more comprehensive view of the congestion attack can be provided by considering payment paths instead of individual channels. When considering congestion along a path, the congestion bottleneck is determined by the channel with the minimum balance b_j^i. Regardless of the balances of other channels along a path, the amount of payment along that path cannot exceed the minimum channel's balance. Thus, the path congestion ratio (PCR) metric is defined as follows.

Definition 4.2. ***Path Congestion Ratio (PCR)*** *is defined as the ratio between the payment $\alpha_j^{n,m}$ and $\min(b_j^i)$ for a path P_j and is given as.*

$$PCR_{\alpha_j^{n,m}}^{n,m} = \begin{cases} \alpha_j^{n,m}/\min(b_j^i) & \text{if } \alpha_j^{n,m} < \min(b_j^i) \\ 1 & \text{if } \alpha_j^{n,m} = \min(b_j^i) \end{cases} \tag{2}$$

Note that the PCR definition above incorporates the bottleneck, which is defined by the minimum balance along the path (i.e., $\min b_j^i$). Similar to the CCR metric, once the payment $\alpha_j^{n,m}$ equals the $\min(b_j^i)$, the PCR takes a value of 1 and the path becomes fully congested. Finally, note that the PCR can be expressed in terms of the CCR as follows:

$$PCR_{\alpha_j^{n,m}}^{n,m} = \begin{cases} \max(CCR_{\alpha_j^{n,m}}^{n,m}) & \text{if } CCR_{\alpha_j^{n,m}}^{n,m} < 1 \\ 1 & \text{if } CCR_{\alpha_j^{n,m}}^{n,m} = 1 \end{cases} \tag{3}$$

One limitation of the PCR definition is that it does not take the path's length into consideration. Paths differ in their lengths where a path length is defined as the number of channels (i.e., edges) along the path. In addition to the PCR metric, it is informative to consider path lengths as well. Intuitively, a congested path with some PCR value and of length M has a more disastrous effect on the network than a different path with the same PCR but with a length of N where $M > N$. This is due to the fact that when a payment amount α_j is locked along a path P_j, it is locked on all channels along that path. Thus, in order to incorporate the path length, the scale path congestion ratio (SPCR) is defined next.

Definition 4.3. ***Scale Path Congestion Ratio (SPCR).*** *Let l_j denote the length of the j^{th} path P_j and L_{max} refer to the maximum length of a network's path. Then, a path's SPCR is defined as the product of the path's length ratio and its PCR and is given as*

$$SPCR^{n,m}_{\alpha^{n,m}_j} = (\frac{l_j}{L_{max}})PCR^{n,m}_{\alpha^{n,m}_j} \tag{4}$$

We note that SPCR incorporates the number of channels congested along a path (i.e., path length) with the path's PCR. Thus, the larger a path's PCR and length, the greater the number of congested channels, resulting in a higher SPCR.The SPCR provides insight into the relative congestion effect of a path compared to others in the network. A higher SPCR corresponds to a longer path, meaning more intermediate nodes along the path are required to lock their coins for the duration of the payment. This introduces the cumulative congestion effect: as coins are restricted across multiple nodes, the network's ability to handle other transactions decreases. Importantly, it does not matter which specific hop along the path is the bottleneck; the critical point is that all nodes along the long path must reserve the current maximum amount required for the payment. Additionally, SPCR is a directional value that depends on the sender and receiver attack pairs and the origin of the payment. The value of L_{max} determined by the PCN sets the maximum allowable path length. For example, in LN, this value is set at 20 hops [8]. Additionally, the SPCR varies between 0 and 1, as discussed earlier, a higher SPCR value indicates a more effective congestion attack.

4.3 Overview Of Proposed Congestion Attacks

In this section, we formulate two congestion attacks as linear optimization problems based on the earlier metrics. The first formulation aims to minimize the attacker's total payment while achieving a desired congestion threshold

quantified by the SPCR metric presented earlier. The second formulation aims to maximize congestion effectiveness by maximizing the SPCR while not exceeding the attacker's allowed budget. The SPCR metric plays a critical role in both formulations. It provides a precise measure of congestion attacks by considering both path length and bottleneck capacity. This combination allows for evaluating the detrimental effects that SCOOP's congestion attacks have on the PCN, which makes SPCR a valuable metric for optimizing the effectiveness of congestion attacks.

It is noteworthy that the proposed congestion attacks are linear in both the objective/cost functions as well the imposed constraints. As such, the problems can be solved either analytically or using linear solvers. However, due to the linear and deterministic nature of the problems they lends themselves easily to a linear solver approach. Next, the congestion attacks are presented in detail.

4.3.1 SCOOP's *Payment Minimization* Attack

(MinPay) In this attack, an attacker is to minimize budget expenditure while achieving a desired congestion threshold. This threshold is determined by setting a target SPCR value, which can be viewed as a quality of service (QoS) metric from the attacker's point of view. The larger the SPCR threshold is, the larger the congestion payments are, and the longer the congested paths, thus, the more severe the attack is. SCOOP's Payment Minimization Attack can be stated as follows.

4.3.2 SCOOP's *Payment Minimization* Attack Formulation

In this formulation, an attacker $A_n \in \mathcal{A}$ is to determine the partial payments on the available paths. In particular, given the set of path capacities and the budget B_n of the attacker A_n on each path P_j, how should the attacker A_n send payments over these paths such that the total forwarding payment is minimized. The problem can be mathematically stated as follows;

$$\min \sum_{\substack{\forall n,m \\ n \neq m}}^{|\mathcal{A}|} \sum_{j=1}^{|\mathcal{P}|} \alpha_j^{n,m} \tag{5}$$

subject to

$$\sum_{j=1}^{|\mathcal{P}|} \alpha_j^{n,m} \leq B_n \quad \forall n, m, j \tag{6}$$

$$0 \leq \alpha_j^{n,m} \leq \min(b_j^i) \quad \forall n, m, i, j \tag{7}$$

$$\text{ThresholdValue} \leq \text{SPCR}^{n,m}_{\alpha_j^{n,m}} \leq 1 \quad \forall n, m, j \tag{8}$$

Where $\alpha_j^{n,m}$ denotes the payment allocated by attacker A_n over path P_j towards attacker A_m, and $SPCR^{n,m}_{\alpha_j^{n,m}}$ denotes the SPCR value of P_j resulting from payment allocations by all attacker nodes. Note that the cost function and constraints in the above problem are linear in the payment $\alpha_j^{n,m}$. The cost function combines the total payments $\alpha_j^{n,m}$ overall congestion paths P_j. The constraint in Eqn. 6 is to enforce that an attacker does not exceed its total budget B_n. The constraint in Eqn. 7 ensures that a path payment does not exceed a path's minimum (i.e., bottleneck) capacity since any larger payment can not be supported over the path. The last constraint in Eqn. 8 ensures that the attacker achieves a certain congestion threshold over all paths. Naturally, the higher the threshold value, the more severe the attack is and the more resources (i.e., payments) that need to be used by the attacker.

It is important to note that the primary goal of the above problem is to minimize the total payment made by the attacker while achieving a target SPCR value, quantified by the obtained value of SPCR in Eqn. 4. Next, we describe the SCOOP's *Congestion Maximization* attack that seeks to maximize congestion in the network.

4.3.3 SCOOP's *Congestion Maximization* Attack

(SPCR Max) In the previous attack, our goal was to minimize the attack budget to meet, as best as possible, a desired congestion performance in terms of the SPCR. One can envision a second problem of interest as follows; How can we maximize the SPCR congestion performance given a certain attacks budget. This is the attack strategy we develop next

4.3.4 SCOOP's *Congestion Maximization* Attack Formulation

In this formulation, an attacker $A_n \in \mathcal{A}$ is to determine the partial payments on the available paths. In particular, given the set of path capacities and the budget of B_n of the attacker A_n on each path P_j, how should the attacker A_n send payments over these paths s such that the congestion is maximized. The problem can be mathematically stated as follows;

$$\max \sum_{\substack{\forall n,m \\ n \neq m}}^{|A|} \sum_{j=1}^{|\mathcal{P}|} \text{SPCR}^{n,m}_{\alpha_j^{n,m}} \tag{9}$$

subject to

$$\sum_{j=1}^{|\mathcal{P}|} \alpha_j^{n,m} \leq B_n \quad \forall n, m, j \tag{10}$$

$$0 \leq \alpha_j^{n,m} \leq \min(b_j^i) \quad \forall n, m, i, j \tag{11}$$

$$0 \leq \mathrm{SPCR}_{\alpha_j^{n,m}}^{n,m} \leq 1 \quad \forall n, m, j \tag{12}$$

The goal of the objective function in Eqn. 9 is to maximize the total of $SPCR_{\alpha_j^{n,m}}^{n,m}$ along all paths across every pair of attackers. This objective seeks to enhance the overall efficiency or performance across all paths, taking into account both their relative lengths and how well resources are allocated relative to their capacities. The constraint in Eqn. 10 is the budget constraint which ensures that the total payments $\alpha_j^{n,m}$ do not exceed the budget B_n. The second constraint in Eqn.11 ensures that congested payment $\alpha_j^{n,m}$ cannot be negative, and it also ensures that a payment of $\alpha_j^{n,m}$ can be routed via path P_j. The third constraint in Eqn.12 ensures that the $SPCR_{\alpha_j^{n,m}}^{n,m}$ is between zero and one. It can not be negative or should not be greater than one, indicating over-resource utilization. The protocols that describe the setup phase and the launching of SCOOP's congestion attacks are described in the Appendix A .

5 Empirical Evaluation

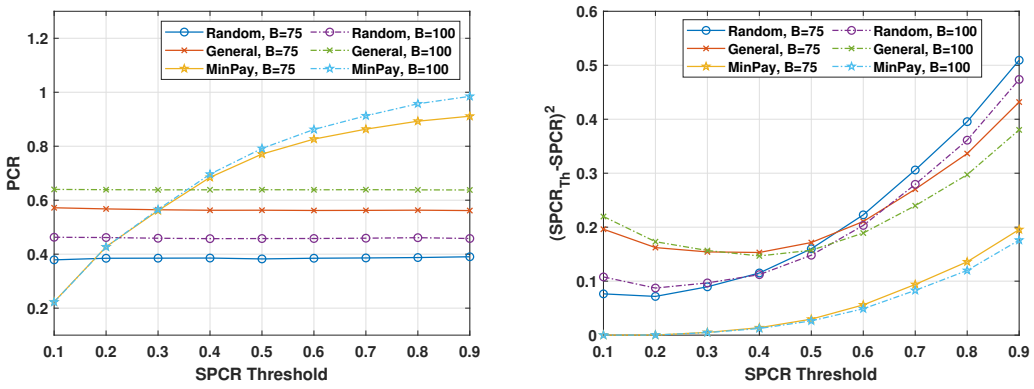
This section investigates the performance of the proposed payment minimization attack (MinPay) and congestion maximization attack (SPCR Max). For comparison, we employ two congestion attacks. These congestion attacks are random congestion attack and general congestion attacks [9]. In the random congestion attack, both the attacker's budget and path payment are randomly selected. On the other hand, the general congestion attack is based on a greedy strategy, where the payments are allocated according to path capacities. Paths with larger capacities are allocated more than the ones with smaller capacities. Moreover, payments are allocated sequentially until the attacker's entire budget is consumed.

For comparison purposes, we use the metrics defined in Section 4.2. The first is the PCR metric used to quantify congestion effect over the lowest capacity channels (i.e., bottleneck) in the different paths. The second is the SPCR metric which in addition to channel congestion incorporates path length and is indicative of the congestion effect over a path. The higher the PCR and SPCR values, the more effective an attack is in terms of congestion. Another metric is LockedPayment, which represents the amount of

locked payment allocated by the attacker. As a metric, the locked payment should be taken in conjunction with the achieved congestion performance. Thus, a merely high or low locked payment by itself is not indicative of the overall performance. To address this, we introduce another metric, the cost-congestion efficiency ratio, denoted as γ. γ is used to quantify how much payment was allocated to provide a given congestion performance. The lower the γ value, the more efficient an attack strategy is. Lastly, we use the distribution of PCR values over the attack paths as information to provide an in-depth view and better visualization of congestion distribution among the different attack paths.

5.1 MinPay Attack Performance Evaluation

In the first experiment, the performance of the proposed MinPay attack is measured. The number of attackers is set at $N = 6$ attacker pairs (i.e., a total of 12 Sybil nodes), and we experiment with two budgets: $B = 75 \times 10^6$ satoshi and $B = 100 \times 10^6$ satoshi. Channel capacities are random with a mean of 4×10^6 satoshi. The mean of the path lengths between attacker nodes is 6 with a maximum length of 8 channels. Experiment results are averaged over 200 iterations. Herein, the performance of the attack is evaluated using different metrics (e.g., PCR and SPCR) as we vary the required SPCR threshold for different B values. Results in Fig. 2a show the resulting PCR for the different



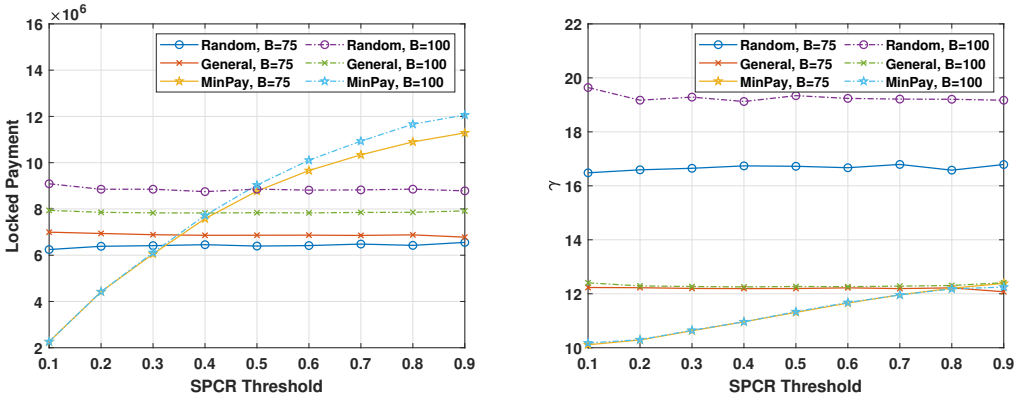
(a) (b)

Figure 2: (a) PCR vs. SPCR Threshold for different attacker's budget (B) values. (b) SPCR deviation for different attacker's budget (B) values.

algorithms as a function of SPCR threshold and B values. Note that the PCR of the MinPay attack increases with the SPCR threshold. This is true since as the SPCR requirement increases, it requires more payment to be allocated

and thus higher resulting PCR values. We note that for the other algorithms there is no such dependence and thus no change in PCR values. It is also noted that a budget increase from $B = 75 \times 10^6$ to $B = 100 \times 10^6$ results in a noticeable increase in PCR values for MinPay attacks since more payments are available for congestion purposes.

The SPCR results are shown in Fig . 2b . We note that instead of plotting the achieved SPCR vs. SPCR threshold value (denoted as $SPCR_{Th}$, the deviation between achieved and required SPCR is depicted instead. This is because path conditions (e.g., capacity and length) might make it infeasible to achieve the required SPCR precisely. In this case, and as discussed earlier, the MinPay attack attempts at meeting the SPCR threshold as close as possible with minimum payment. We note that the initial SPCR deviation of the MinPay is close to zero as the threshold is small and thus can be achieved with the given budget. However, as the threshold increases it becomes more difficult to satisfy it and thus the deviation increases. However, we note that the MinPay achieves the smallest deviation in comparison to other attack strategies regardless of the threshold. We also note that the deviation decreases as the budget is increased, which validates our argument.



(a) (b)

Figure 3: (a) LockedPayment vs. SPCR Threshold for different attacker's budget (B) values. (b) γ vs. SPCR Threshold for different attacker's budget (B) values.

The total payment allocated by the different attacks is summarized in Fig. 3a. However, we note that since the different attacks have varying PCR performances, the locked congestion payment alone is not sufficient for comparison of performances. To this end, we introduce the average cost-to-congestion ratio γ defined as:

$$\gamma = \frac{1}{k} \sum_{\substack{\forall n,m \\ n \neq m}} \sum_{j} \frac{\alpha_j^{n,m}}{SPCR_{\alpha_j^{n,m}}^{n,m}}$$

It is an average operation (i.e., arithmetic or sum of value $\alpha_j^{n,m}/SPCR_{\alpha_j^{n,m}}^{n,m}$ divided over the number of iterations, k). A small γ value implies a small overall payment and a large PCR value. The more efficient an attack is, the smaller is the γ value is. The γ corresponding to Figs. 2a and 3a is plotted in Fig. 3b. It can be see that the MinPay attack has the lowest γ value and is thus the most efficient. As we can see, The random congestion attack has more variability due to a lack of strategic payment allocation, which makes it sensitive to budget changes from $B = 75 \times 10^6$ satoshi to $B = 100 \times 10^6$ satoshi.

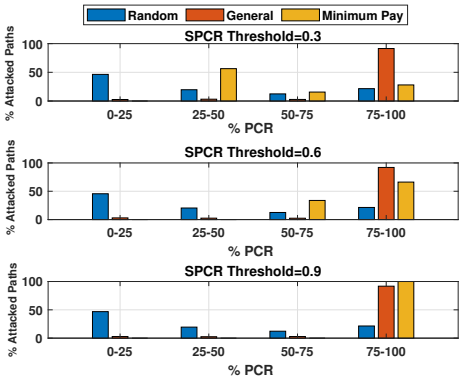


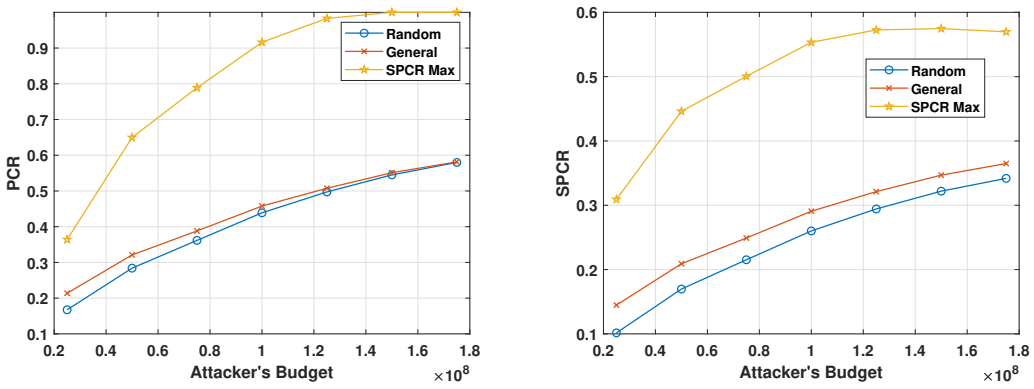
Figure 4: $\%PCR$ Distribution

Results in Fig. 4 show the distribution of PCR values for $B = 100 \times 10^6$ as the SPCR threshold is varied. In particular, the x-axis represents PCR values (i.e., congestion ratios) which range from $0 - 100\%$ divided into 4 equal intervals. While the y-axis represents percentage of paths that fall within each PCR interval. We note that achieved PCR distribution for all algorithms, except for the MinPay, are similar for all thresholds. However, for the MinPay, note that the distribution changes with every SPCR threshold. Note that, with unlimited resources, it is advantageous to have all attack paths be in the 75-100% interval since this reflects the best congestion performance. However, since in the MinPay it is only required to satisfy the required SPCR threshold value, not all channels fall in the 75-100% interval. For example, when the threshold is set at 0.3, we note that with the MinPay around 50% of the paths fall in the 25-50% interval with the remaining paths having a better congestion (i.e., in the 50-75 and 75-100%) which is expected.

However, as the threshold is increased to 0.9, then 100% of the paths have a congestion ratio of 75-100%. This reflects the effectiveness of the MinPay in payment allocation. This is in contrast to other attacks. For example, the random algorithm allocates payments such that most of the channels have a 0-25% congestion which reflects a poor payment allocation. Moreover, the general attack allocates payments such that most channels are in the 75-100% range regardless of the threshold. This indicates that inefficient allocation is being made when compared to MinPay.

5.2 SPCR Max Attack Performance Evaluation

In the second experiment, the performance of the proposed SPCR Max attack is investigated. Here, the goal is to solely maximize the SPCR and thus the resulting PCR as the total allocated budget B is varied. Experiment parameters are the same as the first experiment. The PCR and SPCR performance of the different attacks are shown in Figs. 5a and 5b. It is evident that the SPCR Max attack outperforms the other attacks. Both the PCR and SPCR increase with the attack budget B . As more budget is available, the PCR reaches the maximum value of 1 and this results in the saturation of the SPCR as well. In contrast to the PCR, the SPCR value reaches a maximum around 0.57. This is due to the scaling down of the PCR value by the channel relative length variable of the SPCR (i.e., l_{P_j}/L_{max}). Additional simulations show that Lockedpayment vs. Attacker's Budget (B) values and γ vs. Attacker's Budget (B) values (see Appendix B).



(a) (b)

Figure 5: (a) PCR vs. Attacker's budget (B) values. (b) SPCR vs. Attacker's budget (B) values.

6 Mitigation Techniques

Reputation system. A reputation-based payment system enhances the security of decentralized networks by imposing initial restrictions on new nodes, limiting payment size and transaction volume to prevent malicious activities like DoS attacks. As nodes build a positive transaction history, these restrictions are gradually relaxed, incentivizing trustworthy behavior. **Updating system parameters.** Adjusting system parameters, such as reducing the maximum payment path length to less than 20 (the max. path length set by LN [12]), is one potential strategy to address the issue. However, this approach only postpones the problem instead of solving it, serving as a temporary fix. Additionally, this is a significant change and is unlikely to be implemented in practice. **Selectively throttling or rate-limiting of neighbors based on either 1) number of HTLCs, or 2) liquidity.** It can mitigate attacks in the Lightning Network. Parameters such as $htlc\_minimum\_msat$, $max\_accepted\_htlcs$, and $max\_htlc\_value\_in\_flight\_msat$ help nodes limit their exposure by capping the smallest HTLC value, the number of HTLCs, and the total value of outstanding HTLCs, respectively. **Alternate between elephant/mice payments.** This technique refers to how the nodes in PCNs can adjust their willingness to route payments based on current PCN conditions. During periods of high congestion, the nodes may prefer to process mice payments. Otherwise, the nodes will process more elephant payments. This technique ensures consistent throughput in PCNs.

7 Conclusion

In this paper, we propose SCOOP consisting of two novel congestion attacks, *Congestion Maximization* attack, and *Payment Minimization* attack, which are presented as linear optimization problems. The goal of the *Payment Minimization* attack is to minimize budget expenditure while achieving a desired congestion threshold. The *Congestion Maximization* attack aims to maximize congestion along the paths in PCN (i.e., making optimal use of an attacker's budget and achieving as much congestion as possible within that budget). The performance of the proposed attacks is compared against random congestion attack and general congestion attack [9].Results show that the proposed attacks outperform existing congestion attacks across multiple metrics. Future work will explore new metrics and attack strategies in PCNs.

8 Acknowledgments

This material is based upon work supported by the National Science Foundation under Award No 2148358, 1914635, 2417062,and the Department of Energy under Award No. DESC0023392. Any opinions, findings and conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation and the Department of Energy.

References

[1] Biryukov, A., Naumenko, G., Tikhomirov, S.: Analysis and probing of parallel channels in the lightning network. In: Financial Cryptography and Data Security: 26th International Conference, FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers. p. 337–357. Springer-Verlag, Berlin, Heidelberg (2022)

[2] Decker: Trampoline slides. `https://residency.chaincode.com/presentations/lightning/Trampoline.pdf` (2023)

[3] Harris, J., Zohar, A.: Flood & loot: A systemic attack on the lightning network. In: Proceedings of the 2nd ACM Conference on Advances in Financial Technologies. pp. 202–213 (2020)

[4] Htlcs are harmful. `https://diyhpl.us/wiki/transcripts/stanford-blockchain-conference/2019/htlcs-considered-harmful/`, in: Stanford Blockchain Conference, 2019

[5] Jourenko, M., Larangeira, M., Tanaka, K.: Payment trees: Low collateral payments for payment channel networks. In: Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part II 25. pp. 189–208. Springer (2021)

[6] Labs, L.: Lightning labs. `https://lightning.engineering/` (2024)

[7] Li, C., Li, P., Zhou, D., Yang, Z., Wu, M., Yang, G., Xu, W., Long, F., Yao, A.C.C.: A decentralized blockchain with high throughput and fast confirmation. In: 2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20). pp. 515–528 (2020)

[8] Lnd/bolts/onion routing. https://github.com/lightning/bolts/blob/6e1bea0d4868cb55fae7590975ae593a228ad3fe/04-onion-routing.md#requirements-2, accessed: 2024-04-11

[9] Lu, Z., Han, R., Yu, J.: General congestion attack on htlc-based payment channel networks. Cryptology ePrint Archive (2020)

[10] Malavolta, G., Sanchez, P.M., Schneidewind, C., Kate, A., Maffei, M.: Anonymous multi-hop locks for blockchain scalability and interoperability. In: 26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019. The Internet Society (2019)

[11] Mizrahi, A., Zohar, A.: Congestion attacks in payment channel networks. In: International Conference on Financial Cryptography and Data Security. pp. 170–188. Springer (2021)

[12] Network, L.: Bolt/onion routing protocol. https://github.com/lightning/bolts/blob/6e1bea0d4868cb55fae7590975ae593a228ad3fe/04-onion-routing.md#requirements-2 (2019)

[13] Pérez-Sola, C., Ranchal-Pedrosa, A., Herrera-Joancomartí, J., Navarro-Arribas, G., Garcia-Alfaro, J.: Lockdown: Balance availability attack against lightning network channels. In: Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers 24. pp. 245–263. Springer (2020)

[14] Poon, J., Dryja, T.: The bitcoin lightning network: Scalable off-chain instant payments (2016)

[15] Raiden network. https://raiden.network/, accessed: 2024-04-11

[16] Payment channels. https://happypeter.github.io/binfo/payment-channels.html, accessed: 2024-04-11

[17] Tikhomirov, S., Pickhardt, R., Biryukov, A., Nowostawski, M.: Probing channel balances in the lightning network (2020)

[18] Visa. https://usa.visa.com/dam/VCOM/download/corporate/media/visanet-technology/visa-net-booklet.pdf, accessed: 2023-09-18

A Algorithms

In this section, we describe the algorithms used in the construction of SCOOP. Algorithm 1 describes how the Sybil nodes are generated and how they attach with honest nodes by opening payment channels. First, a set of the honest nodes, V', is initialized, where each node generates a key pair using a key generation function for identification. Next, a number of Sybil nodes, N, is created, each generating its key pair. A set of targeted nodes, $\mathbb{T}$, is selected from the honest nodes set, and channels are established between the Sybil nodes and these target nodes.

Algorithm 1: Setup And Sybil Node Creation

```
   /* V' is the set of honest nodes in the PCN                    */
1  for i = j; j ≤ |V'|; j++ do
2  |  node i performs KeyGen(1^λ) → (sk_i, vk_i)
   /* Sybil nodes being created                                   */
3  An number of Sybil nodes N is selected
4  A = ∅
5  for n = 1; n ≤ |A|; n++ do
6  |  Sybil node A_n is created: CreateSybil(A_n) → (sk_{A_n}, vk_{A_n})
7  |  A = A ∪ {vk_{A_n}}
   /* Channel creation between Sybil nodes and honest nodes
      */
8  A set T of target nodes from V' is selected using
     SelectNode(V') → T, where |T| ≥ 1
9  For each A_n ∈ A, a channel is created between A_n and members of
     T, t ∈ T using PC.Open(vk_{A_n}, vk_v, B_{A_n}) → success.
```

Algorithm 2 implements congestion attacks SCOOP's *Payment Minimization* Attack and SCOOP's *Congestion Maximization* Attack, respectively, between pairs of Sybil nodes in the PCN. In both attacks, The attacks begin by iterating over each attacker $A_n \in \mathcal{A}$. For a given attacker A_n, the algorithms find all available paths $\mathcal{P}$ that originate from A_n and connect to all other Sybil nodes $A_{n'} \in \mathcal{A}$ using using $\mathsf{FindAllPaths}(A_n, A_{n'})$ function. For each identified path $P_j \in \mathcal{P}$, the algorithms use $\mathsf{Probing}(P_j)$ function to determine the minimum channel capacity, $\min(b^i_{P_j})$, which represents the bottleneck capacity of the path P_j, and computes the hop counts for each path P_j using the $\mathsf{HopCount}(P_j)$ function, which calculates the number of intermediate nodes along the path P_j.These values are stored in the $\mathbb{C}$ and $\mathcal{L}$, respectively. In SCOOP's *Payment Minimization* Attack, a threshold value is introduced to minimize the payment costs, ensuring that the allocated payments $\alpha_{P_j}^{A_n,A_{n'}}$ remain efficient. In contrast, the SCOOP's *Congestion Maximization* Attack

maximizes congestion across the network without applying a threshold value. After computing the payments $\alpha_{P_j}^{A_n,A_{n'}}$, the budgets B_{A_n}, and the channel capacities $b_{P_j}^i$ along all of the paths P_j are updated to reflect the effect of the attack on the network's channels.

Once the updates are applied, the next attacker A_n begins their iteration. Importantly, this sequential process ensures that each attacker operates on a network state that reflects the cumulative impact of the updates from all previous attackers. As a result, no two attackers see the same network conditions, and the computation of payments $\alpha_{P_j}^{A_n,A_{n'}}$ adapts dynamically to the updated channel capacities and states. This sequential dependency is critical for effectively simulating and evaluating the evolving impact of the attacks on the network. After all attackers $A_n \in \mathcal{A}$ have completed their iterations, Step 9 onward executes the congestion attack. This involves creating Hashed Time-Locked Contracts (HTLCs) along each path P_j using the allocated payments $\alpha_{P_j}^{A_n,A_{n'}}$, propagating requests sequentially between nodes with a timeout t_{P_j} and hash commitment Y. The attackers $A_{n'}$ do not execute the payments but use the HTLC setup to congest the network and exploit its resources.

B SPCR Max Attack Performance Evaluation

The locked payment results are shown in Fig. 6a where it is shown that payments allocated by both the random and general attacks increase with B. However, the SPCR Max attack does not show the same increase pattern. This is due to the fact that the SPCR Max is able to attain the maximum PCR/SPCR with less allocated funds which results in the plateauing of its curve. The relative efficiency of the different attacks can be quantified using the γ parameter which is depicted in Fig. 6b The results indicate that the SPCR Max becomes more efficient as more attack budget is available. In fact, at large B values, it can be more than two times more efficient than other algorithms.

C Implications of Congestion Attacks on PCNs

1. Targeting High Liquidity Nodes: The adversary in SCOOP can target the nodes with high liquidity and/or those with a large number of payment channels (high degree nodes). These high liquidity/degree nodes are often referred to as routing helpers/trampolines/trampoline nodes [2] and play a pivotal role in facilitating transactions across the LN.

Algorithm 2: Unified Algorithm for SCOOP's *Payment Minimization* and *Congestion Maximization* Attacks

/* This algorithm is run for every pair of Sybil nodes in $\mathcal{A}$. L_{max} is a system parameter. ThresholdValue is optional and used only in the *Payment Minimization* Attack. */

Initialize set $\mathcal{P} = \emptyset$

Maintain lists $\mathbb{C} = \emptyset$, $\mathcal{L} = \emptyset$, $\mathcal{M} = \emptyset$, $\mathcal{B} = \emptyset$

for *each attacker* $A_n \in \mathcal{A}$ **do**

  runs $\mathsf{FindAllPaths}(A_n, A'_n)$ to find the set of paths $\mathbb{P}$ originating from A_n to all other attackers $A'_n \in \mathcal{A}$

  $\forall\ P_j \in \mathcal{P}$, compute $\mathsf{Probing}(P_j) \rightarrow \min(b^i_{P_j})$, where i is the index of each channel along path P_j, and append each $\min(b^i_{P_j})$ to $\mathbb{C}$

  $\forall$ paths $P_j \in \mathcal{P}$, compute $\mathsf{HopCount}(P_j) \rightarrow l_{P_j}$, and append l_{P_j} to $\mathcal{L}$

  /* B_{A_n} is the budget allocated for attacker A_n */

  if *Attack is Payment Minimization* **then**

    Compute *Payment Minimization* $(\mathbb{C}, \mathcal{L}, B_{A_n}, L_{max}, \text{ThresholdValue}) \rightarrow \mathcal{M}$; /* Uses ThresholdValue */

  else if *Attack is Congestion Maximization* **then**

    Compute *Congestion Maximization* $(\mathbb{C}, \mathcal{L}, B_{A_n}, L_{max}) \rightarrow \mathcal{M}$; /* No ThresholdValue */

  /* $\mathcal{M}$ includes the attack payments $\alpha_{P_j}^{A_n, A_{n'}}$ for attacker A_n */

  After finding all $\alpha_{P_j}^{A_n, A_{n'}}$, update B_{A_n} and $b^i_{P_j}$ along each path P_j

for *each path* $P_j \in \mathcal{P}$ **do**

  A'_n generates $X \leftarrow\$ \{0,1\}^\lambda$, computes $H(X) \rightarrow Y$, and sends Y to A_n

  /* t_j is the number of blocks. */

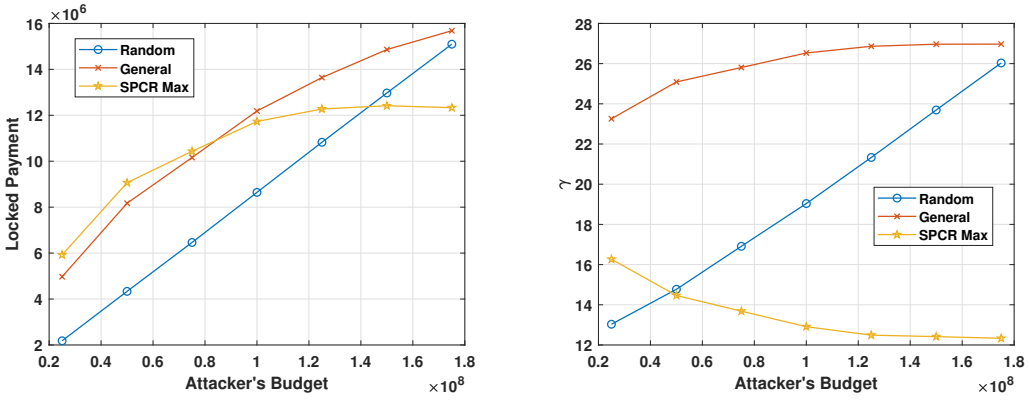
  for *every pair of consecutive nodes* $k, k+1$ *along each* P_j **do**

    k sends the tuple $\mathsf{CreateHTLC}(\mathsf{vk}_k, \mathsf{vk}_{k+1}, \alpha_{P_j}^{A_n, A_{n'}}, Y, t_j)$ to $k+1$

  if $curr_{time} == t_j$ **then**

    $k+1$ sends the tuple $\mathsf{failHTLC}(\mathsf{vk}_k, \mathsf{vk}_{k+1}, \mathsf{Cannot\ Fulfill})$ to k

By establishing payment channels with these nodes, the adversary can cause significant disruptions by locking their liquidity, preventing them



(a) (b)

Figure 6: (a) Lockedpayment vs. Attacker's Budget (B) values. (b) γ vs. Attacker's Budget (B) values

from routing transactions, thereby reducing the overall throughput of the LN.

2. Disconnecting LN: Payment paths in LN can use payment channels that act as critical bridges, linking independent clusters of nodes and enabling the flow of payments between them. An attacker using SCOOP could could intentionally congest these payment channels, disrupting the payments that depend on them.

3. Node's reputation: Nodes in LN facilitate payment forwarding and generate routing fees. However, they can be vulnerable to congestion attacks, which may damage their reputation if they are unable to process requests, resulting in "temporary channel failure" errors. These errors signify insufficient funds, leading senders to lose trust in the node and opt against future transactions.

D Hashed Time Locked Contracts (HTLCs) and their Limitations

The LN uses Hashed Time Locked Contracts (HTLCs) to enable secure, trustless, multi-hop payments. HTLCs ensure staggered payments, where each node forwards payments while being protected from loss through the blockchain's dispute resolution mechanism. However, HTLCs have limitations that can be exploited for congestion attacks:

1) **Maximum Accepted HTLCs Limitation**: Nodes have a limit on the number of HTLCs they can handle (e.g., 483 in LN[11]). Attackers can fill these slots with numerous small payments (e.g., dust payments of 5.46×10^{-6} BTC [11]), blocking legitimate transactions. 2) **HTLC Expiration Time Limitation**: HTLCs use a Check Lock Time Verify (CLTV), with 14, 40, or 144 blocks expiring in LN implementations. Attackers can exploit this linear accumulation (up to 2016 blocks [11]) to lock coins for extended periods, delaying transactions. 3) **Channel Capacity Limitation**: Channel capacities are fixed at opening, limiting transaction amounts. Attackers can lock coins along paths by withholding preimages, as in Figure 1a, Attacker1 locks five coins with Peter, who forwards four coins to Carol after deducting routing fees. Attacker2 then withholds the preimage, locking a total of twelve coins, congesting the path, and reducing path capacity.