Optimizing Virtual Payment Channel Establishment in the Face of On-Path Adversaries

Lukas Aumayr[1,5] Esra Ceylan[2] Yannik Kopyciok[3] Matteo Maffei[1,5]
Pedro Moreno-Sanchez[4] Iosif Salem[3] Stefan Schmid[3]
[1]TU Wien [2]University of Vienna [3]TU Berlin [4]IMDEA Software Institute
[5]Christian Doppler Laboratory Blockchain Technologies for the Internet of Things

***Abstract*—Payment channel networks (PCNs) are among the most promising solutions to the scalability issues in permissionless blockchains, by allowing parties to pay each other off-chain through a path of payment channels (PCs). However, routing transactions comes at a cost which is proportional to the number of intermediaries, since each charges a fee for the routing service. Furthermore, analogous to other networks, malicious intermediaries in the payment path can lead to security and privacy threats. Virtual channels (VCs), i.e., bridges over PC paths, mitigate the above PCN issues, as after the VC creation, intermediaries are excluded from every VC transaction. However, creating a VC has a cost that must be paid out of the bridged PCs' balance. Currently, we are missing guidelines on how and where to best set up VCs.**

In this work, we address this VC-setup problem, formalizing it as an optimization problem. We present an integer linear program (ILP) to compute the globally optimal VC-setup strategy in terms of fees, security, and privacy. We then accompany the computationally heavy ILP with a fast local greedy algorithm. Our solutions can be used with any on-path adversary, given that its strategy can be expressed as a set of corrupted nodes that is estimated by the honest nodes. We evaluate our greedy algorithm over a Lightning Network (LN) snapshot, the largest PCN. Our findings confirm that our greedy strategy minimizes costs while protecting against threats of on-path adversaries and may serve the LN community as guidelines for the deployment of VCs.

I. Introduction

Permissionless cryptocurrencies face severe scalability challenges, as they rely on a set of mutually untrusted users located across the world to maintain a distributed and publicly verifiable transaction ledger. The transaction throughput today is limited to tens of transactions per second at best, while transactions can take up to 60 minutes to be confirmed.

Payment channels (PC) have emerged as one of the most promising scalability solutions, and instances such as the Lightning Network [1] are gaining traction. In this approach, Alice and Bob can create a PC between them with a single on-chain transaction that transfers their coins into an escrow (or multi-signature) controlled by both of them with the additional guarantee that they can get refunded at a mutually agreed-upon time. After that, Alice and Bob can pay each other off-chain by exchanging authenticated copies of the updated balances in the escrow. Finally, the PC is closed with an on-chain transaction representing the last authenticated distribution of coins.

ISBN 978-3-903176-63-8 © 2024 IFIP

PCs can be linked to form a network, also called a payment channel network (PCN), where any two users can perform a payment if they are connected by a path of PCs. The payment in the PC between Alice and the first intermediary is forwarded along the intermediary PCs until it reaches Bob. A key challenge in this approach is then to ensure that the balance updates of all PCs in the path are atomic to prevent any intermediary (i.e., Ingrid) from trivially stealing the money by denying forwarding it.

State-of-the-art techniques to construct atomic multi-hop payments [2]–[8] require that intermediaries are involved in every single payment. This approach brings several disadvantages: (i) reduction of the payment reliability (e.g., Ingrid may simply be offline or crash); (ii) increase in the payment latency since additional PCs are required; (iii) high payment costs as each intermediary charges a fee per transaction for providing the routing service; and (iv) possible leakage of sensitive information to the intermediaries, which enables a number of security and privacy issues, such as route hijacking [9], wormhole attacks [2] or user anonymity [4], to name a few.

Recently, the concept of virtual channels (VCs) [10]–[13] has been proposed to improve upon the aforementioned drawbacks of PCs. A VC can be seen as a bridge over two PCs. For instance, assume that Alice and Bob have a PC with an intermediary, Ingrid. In order to set up a VC, Ingrid must collaborate and coordinate with Alice and Bob to lock coins in their corresponding PCs in order to use those coins to build a VC directly between Alice and Bob. This approach brings the following benefits: (i) Alice and Bob can pay each other "as if they had a PC between them", that is, without the involvement of Ingrid; (ii) payment latency is reduced to one hop; (iii) payment fees charged by the intermediaries for their routing service are avoided; and (iv) the details of every single payment are not revealed to possibly malicious or curious intermediaries. Note that intermediaries still charge fees for the coordination service when establishing the VC.

A crucial question, not yet addressed in the literature, is *what strategy should users follow to open VCs while optimizing the cost-effectiveness, as well as on-path security and privacy benefits provided by VC networks?* This is an optimization problem, given that the funding to be locked, and thus the number of VCs a party can establish, is limited by the number of underlying PCs and the amount of coins that are locked on them. To ensure *on-path* security and privacy we provide a modular framework for preventing attacks. Our algorithms use as input a set of nodes that the honest nodes estimate to be corrupted. The honest nodes derive the set of potentially corrupted nodes based on their assumptions on the adversary. We demonstrate how our framework functions through three exemplary well-studied attacks in the literature. Note that, while several existing works have studied from a game theoretic perspective how a PCN should evolve based on the fee optimization goal of the users [14]–[16], none of them considered VCs, nor on-path security and privacy goals.

We make the following ***contributions***. First, we address the VC-setup problem, formalizing it as an optimization problem of three distinct goals: (i) cost-effectiveness of the transactions (i.e., fees) while providing (ii) security and (iii) privacy guarantees against on-path adversaries, and prove that the optimization problem is NP-hard. On-path adversaries account for a significant share of attacks in the PCN-related literature: e.g., they may aim to perform denial-of-service and wormhole attacks, or to harm value privacy and relationship anonymity properties, among many others [2], [4], [9], [17]–[20]. Such attacks have been shown to potentially have a severe impact in practice [21].

On-path adversaries can do damage depending on the attack that they are carrying out. In this work, we provide a general framework for mitigating attacks of on-path adversaries. Our goal is to prevent the attacks of an adversary, parameterized by an adversarial budget, a strategy, and different types of on-path attacks. The adversary corrupts nodes according to the strategy and attack types until they run out of budget. We study three exemplary types of attacks: value privacy attacks, relationship anonymity attacks, and wormhole attacks. Our adversarial strategy consists of corrupting nodes, such that the largest fraction of payments is affected by the given attack types, cf. [21]–[24]. In practice, the adversarial budget can be estimated based on the value of the coins and information sent in these transactions: The higher the budget is set, the more cost-efficient nodes are denied to the adversary for corruption, making an attack ever more expensive or even completely impossible, if all intermediary nodes are bypassed. In our solution, the adversarial strategy and the studied attack types can be replaced in a modular way. Thus, aside from our results, we provide a generic approach for users and developers to analyze this problem.

Second, we analytically show a synergy between the different VC optimization objectives. In particular, we prove that minimizing transaction fees by the appropriate use of VCs also prevents attacks from on-path adversaries, such as those against value privacy and relationship anonymity, or wormhole attacks. In practice, this implies that users can set up their VCs following a single strategy to minimize their transaction costs, and as a side benefit, they will be secure against on-path adversaries. We demonstrate the latter for the three exemplary on-path attacks on security and privacy in study.

Third, and motivated by the uncovered synergy between the objectives, we describe concrete approaches to devise fee optimization strategies which mitigate on-path security and privacy attacks (and specifically value privacy, relationship anonymity, and wormhole attacks). In particular, we present both an efficient approach (based on a greedy routing algorithm) to optimize the cost-effectiveness, security, and privacy of PCNs using VCs, and a rigorous and exact approach based on integer linear programming (ILP), which is computationally intractable (we also propose how to reduce the running time of the ILP). The network topology of PCNs such as the Lightning Network is known publicly. In our exact ILP-based approach, we additionally assume that all transactions we want to route are known globally, in order to find the globally optimal solution. Our greedy algorithm on the other hand, can be applied locally, using only the information of individual nodes.

Finally, we evaluate our greedy optimization approach on a recent snapshot of the Lightning Network (LN). We show that our transaction cost minimization strategy is efficient and effective, and indeed subsumes the strategies to optimize for on-path security and privacy. We find that depending on how many payments two endpoints plan to conduct via the virtual channel, the routing cost can be reduced significantly, for example, to about half compared to a normal payment for two consecutive payments, or to about 3% for 50 consecutive payments. In addition to this cost reduction, other users can utilize these virtual channels to route their payments through a potentially cheaper path.

To summarize, for the first time, we present both an analytical and an empirical study of the impact of using VCs in (current) PCNs in terms of cost-effectiveness of the transactions as well as security and privacy guarantees. The results of this work motivate the deployment of VCs, and we hope that they can encourage the PCN community and developers to include VCs within current PCN software and make them accessible to PCN users.

Related work. Virtual channels were introduced by Dziembowski et al. [10] to overcome the requirement that intermediaries along a channel route need to be online (a concern also considered in [25], [26]) and explicitly confirm all mediated transactions. Recent work has extended the deployment scope of virtual channels, introducing efficient protocols that are compatible with Bitcoin and other popular cryptocurrencies [11]–[13]. While existing literature on virtual channels revolved around protocol design aspects, to the best of our knowledge, our paper is the first to investigate the problem of optimizing the allocation of virtual channels in order to improve the security and efficiency of PCNs. In parallel work [27], Khamis and Rottenstreich studied how to amortize the creation of new channels through reduced routing costs, however, without accounting for security aspects. The extended version of the related work appears in [28].

Paper organization. We introduce background on PCNs in Section II; our model and problem formulation in Section III. We present an efficient greedy algorithm in Section IV, its evaluation in Section V, and an exact algorithm in Section VI. We defer additional material, such as additional algorithms and discussions, to an extended technical report [28].

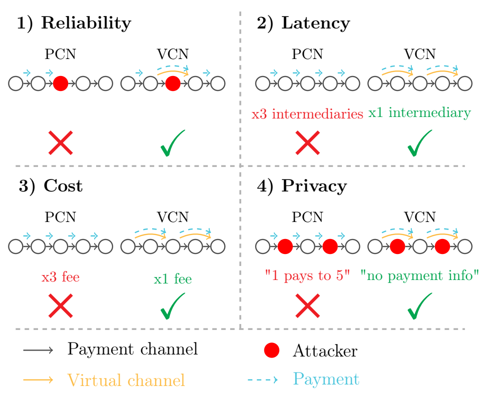

Fig. 1. Comparison between PCN and VCN.

II. Background and Problem Overview

Payment channel networks (PCNs). A PCN [4] is a directed graph $\mathcal{G} := (\mathcal{V}, \mathcal{E})$. Nodes $\mathcal{V}$ represent users and edges $\{e_{i,j}, e_{j,i}\} \subset \mathcal{E}$ represent PCs between users. The weight on a directed edge denotes the amount of remaining coins that can be forwarded on that direction. For every pair of edges $\{e_{i,j}, e_{j,i}\}$, users v_i and v_j can exchange any part of their balance freely. Moreover, each directed edge $e_{i,j}$ between users v_i and v_j is associated with two non-negative numbers, the base fee f_i, and the proportional fee p_i, that together determine the fees that each user charges for forwarding the payments. For a forwarded amount α via $e_{i,j}$, v_i charges $fee(e_{i,j}, v_i) = f_i + p_i \cdot \alpha$. We denote a PC $\{e_{i,j}, e_{j,i}\}$ with the tuple $(pc_{\langle v_i, v_j \rangle}, \beta_i, \beta_j, f_i, f_j, p_i, p_j)$, where $\beta_{k \in \{i,j\}}$ is the initial balance of each node upon channel creation.

The success of a payment between two users depends on the capacity available in the path connecting the sender s to the receiver r. Assume that s wants to pay α coins to r and that they are connected through a path $s \to u_1 \to \ldots \to u_n \to r$. The fees charged for every node in the path depend on the forwarded amount. That is, v_n charges $fee_n = f_{n,n+1} + p_{n,n+1} \cdot \alpha$ and in general v_j charges $fee_j = f_{j,j+1} + p_{j,j+1} \cdot (\alpha + \sum_{k=j+1}^{n} fee_k)$, for $j = 1, \ldots, n$ (each node forwards α and the forwarding fees of the remaining nodes in the path). Such a payment is successful if (i) s starts the payment with a value $\alpha^* := \alpha + \sum_{j=1}^{n} fee_j$ and (ii) every edge on the path has a balance of at least α'_i, where $\alpha'_i := \alpha^* - \sum_{j=1}^{i-1} fee_j$ (the initial payment value α minus the fees charged by the previous users in the path), $e_{j,j+1} = (u_j, u_{j+1})$, and $u_{n+1} = r$. If the payment is successful, the balance of every edge $e_{j,j+1}$ on the path from s to r is decreased by α'_i, while the balance of every edge $e_{j+1,j}$ is increased by α'_i.

PCN challenges. For successful payments, intermediaries must actively participate and must not disturb them, either actively (e.g., dropping it) or passively (e.g., being offline). Thus, PCN payments suffer from the following drawbacks:

Reliability: If intermediaries are offline or do not forward the payment (e.g., the red user in Figure 1), the payment fails.

Latency: The time to process a payment is directly proportional to the number of intermediate users. E.g., the latency of the payment shown in Figure 1 (latency section) could be reduced if a shorter path between nodes 1 and 5 existed.

Cost: The payment cost is proportional to the number of intermediate users, since each charges a routing fee.

Privacy: Each intermediary learns sensitive information. Recent work [4], [29] has shown that intermediaries can learn details about who pays what to whom in the currently deployed Lightning Network. While alternative payment mechanisms that hide (some of) the information required in such payment exist, e.g., [2], [3], they have not been adopted yet and still protect only some sensitive information but not other (e.g., the payment amount) and also do not decrease routing fees.

Virtual channels (VCs). Bypassing intermediaries can mitigate these drawbacks. One could build a new PC, but this requires an expensive on-chain transaction and additional funds. Instead, a VC can be created off-chain between two users, say Alice and Bob, who have a PC with a common intermediary, say Ingrid. Using a 3-party protocol, the users can block coins in the underlying PCs and move them into the VC between Alice and Bob. After that, Alice and Bob can perform arbitrarily many payments without involving Ingrid. The amount of VCs that can be created are thus limited by the balances of the underlying PCs. Yet, it is interesting to deploy VCs as they provide several advantages over PCNs.

Reliability: Payments are carried out without involving the intermediary user, who cannot thus disturb it either actively (e.g., dropping it) or passively (e.g., being offline). In Figure 1, the malicious node 3 does not participate in the payment between 1 and 5 as it is omitted by the VC between 2 and 4.

Latency: VCs lead to shorter paths. Since there are fewer intermediate users, the latency of the overall payment is reduced. In the running example, the latency is reduced from 3 to 1 intermediaries, assuming that two VCs have been created.

Cost: Assume, for simplicity, that users charge the same fees for forwarding a payment through a PC and a VC. In such a case, as with latency, the fact that VCs lead to shorter paths, can also help to reduce the overall payment cost in terms of fees. In Figure 1, the transaction cost using VCs is reduced to the fee charged by the only intermediary that is involved, avoiding thus the fees charged by nodes 2 and 4.

Privacy: The fact that fewer intermediaries are participating in the payment improves the privacy of the overall payment. And although intermediaries are part of the 3-party creation of the VC and thus learn who are the two VC endpoints, they no longer see the amounts of the individual payments routed through the VC. For instance, in Figure 1, the malicious node 2 would learn that there exists a VC between nodes 1 and 3 as it needs to help them to set the channel up, but afterwards the node 2 does not learn when a VC is used.

VCs in practice. Despite the advantages provided by VCs, we currently lack a comprehensive analysis leading to a set of guidelines to help the users decide when to open VCs, with what neighbors, and under what circumstances. Ideally, a user would like to open a VC with every other user in the network. Unfortunately, this is not possible since each user has a limited budget, i.e., the amount of coins available on her PCs which need to be locked to create a VC. In this state of affairs, the following questions arise: how should a user choose which neighbor to open a VC with? how many payments are required to amortize the cost of opening a VC? what strategy should a user follow to maximize the security and privacy gains against on-path adversaries when opening VCs?

III. Modelling virtual payment channel networks

We introduce a more formal model of virtual payment channel networks (VPCNs). We will then discuss the security and privacy threats by on-path adversaries, define the studied optimization goal on VPCNs, and show its NP-hardness.

Definition 1 (VPCN)**.** A virtual payment channel network, VPCN, is defined as a graph $\mathcal{G} := (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ denotes the set of users in the network and $\mathcal{E} := \mathcal{E}_p \cup \mathcal{E}_v$ denotes the set of channels. In particular, $\mathcal{E}_p$ denotes the set of payment channels, and $\mathcal{E}_v$ denotes the set of VCs. Each payment channel is defined by a tuple $(pc_{\langle v_1,v_2\rangle}, \beta_1, \beta_2, f_1, f_2, p_1, p_2)$, where $pc_{\langle v_1,v_2\rangle}$ denotes a payment channel identifier, $\beta_{i\in\{1,2\}}$ denotes the current balance of the node $v_{i\in\{1,2\}}$, $f_{i\in\{1,2\}}$ is the base fee and $p_{i\in\{1,2\}}$ the fee rate (proportional to the amount paid) charged to use this channel in each direction, respectively. Analogously, a VC is defined by a tuple $(vc_{\langle v_1,v_2\rangle}, \beta_1, \beta_2, f_1, f_2, p_1, p_2, f_e)$, where f_e denotes the VC establishment fee.

A VPCN is defined with respect to a blockchain $\mathbb{B}$ that stores publicly accessible entries of the form $(v, \beta^{\textit{on-chain}})$ where v denotes an address of the underlying blockchain and $\beta^{\textit{on-chain}}$ denotes its on-chain balance. For readability, we hereby use $\mathbb{B}[v]$ to denote the on-chain balance of v in $\mathbb{B}$. A VPCN exposes the operations expressed in Figure 2.

openPC $(v_1, v_2, \beta_1, \beta_2, f_1, f_2, p_1, p_2)$

$v_{i\in\{1,2\}}$:	Nodes
$\beta_{i\in\{1,2\}}$:	PC initial capacity
$f_{i\in\{1,2\}}$:	Base routing fees
$p_{i\in\{1,2\}}$:	Proportional routing fee

- If $\mathbb{B}[v_1] < \beta_1$ or $\mathbb{B}[v_2] < \beta_2$, abort. Else, create a new payment channel $(pc_{\langle v_1,v_2\rangle}, \beta_1, \beta_2, f_1, f_2, p_1, p_2) \in \mathcal{E}_p$
- Update blockchain as $\mathbb{B}[v_1] = \mathbb{B}[v_1] - \beta_1$ and $\mathbb{B}[v_2] = \mathbb{B}[v_2] - \beta_2$

closePC $(pc_{\langle v_1,v_2\rangle})$

$pc_{\langle v_1,v_2\rangle}$:	PC identifier

- Let $(pc_{\langle v_1,v_2\rangle}, \beta_1, \beta_2, f_1, f_2, p_1, p_2)$ be the corresponding entry in $\mathcal{E}_p$. If such entry does not exist, abort.
- Set $\mathbb{B}[v_1] = \mathbb{B}[v_1] + \beta_1$, $\mathbb{B}[v_2] = \mathbb{B}[v_2] + \beta_2$ and remove from $\mathcal{E}_p$: $(pc_{\langle v_1,v_2\rangle}, \beta_1, \beta_2, f_1, f_2, p_1, p_2)$

openVC $(c_{\langle v_1,v_I\rangle}, c_{\langle v_I,v_2\rangle}, \beta_1, \beta_2, f_1, f_2, p_1, p_2, f_e)$

$c_{\langle v_1,v_I\rangle}$, $c_{\langle v_I,v_2\rangle}$:	PC or VC identifiers
$\beta_{i\in\{1,2\}}$:	Initial VC balance
$f_{i\in\{1,2\}}$:	Base routing fee
$p_{i\in\{1,2\}}$:	Proportional r. fee
f_e:	Establishing fee

- Let $(c_{\langle v_1,v_I\rangle}, \beta'_1, \beta'_I, f'_1, f'_I, p'_1, p'_I)$ and $(c_{\langle v_I,v_2\rangle}, \beta''_I, \beta''_2, f''_I, f''_2, p''_I, p''_2)$ be the entries in $\mathcal{E}_p$ or $\mathcal{E}_v$ corresponding to $c_{\langle v_1,v_I\rangle}$ and $c_{\langle v_I,v_2\rangle}$. If $(\beta_1 + f_e) > \beta'_1$ or $\beta_2 > \beta'_I$ or $\beta_2 > \beta''_2$ or $\beta_1 > \beta''_I$, abort.
- Update the entries in $\mathcal{E}_p$ or $\mathcal{E}_v$ as $(c_{\langle v_1,v_I\rangle}, \beta'_1 - (\beta_1 + f_e), \beta'_I - \beta_2 + f_e, f'_1, f'_I, p'_1, p'_I)$ and $(c_{\langle v_I,v_2\rangle}, \beta''_I - \beta_1, \beta''_2 - \beta_2, f''_I, f''_2, p''_I, p''_2)$
- Add $(vc_{\langle v_1,v_2\rangle}, \beta_1, \beta_2, f_1, f_2, p_1, p_2)$ in $\mathcal{E}_v$

closeVC $(vc_{\langle v_1,v_2\rangle})$

$vc_{\langle v_1,v_2\rangle}$:	VC identifier

- In $\mathcal{E}_v$, remove the corresponding entry $(vc_{\langle v_1,v_2\rangle}, \beta_1, \beta_2, f_1, f_2, p_1, p_2)$ if it exists. If such entry does not exist, abort.
- Update entries in $\mathcal{E}_p$ as $(pc_{\langle v_1,v_I\rangle}, \beta'_1 + \beta_1, \beta'_I + \beta_2, f'_1, f'_I, p'_1, p'_I)$ and $(pc_{\langle v_I,v_2\rangle}, \beta''_I + \beta_1, \beta''_2 + \beta_2, f''_I, f''_2, p''_I, p''_2)$

update{P,V}C $(c_{\langle v_1,v_2\rangle}, \beta)$

$c_{\langle v_1,v_2\rangle}$:	Channel identifier

- Let $(c_{\langle v_1,v_2\rangle}, \beta_1, \beta_2, f_1, f_2, p_1, p_2)$ be the corresponding entry in $\mathcal{E}_p \cup \mathcal{E}_v$. If such entry does not exist or it exists but $\beta_1 < \beta$, abort.
- Update the channel as follows $(c_{\langle v_1,v_2\rangle}, \beta_1 - \beta, \beta_2 + \beta, f_1, f_2, p_1, p_2)$

pay $((c_{\langle s,v_1\rangle}, \dots, c_{\langle v_n,r\rangle}), \beta)$

$(c_{\langle s,v_1\rangle}, \dots, c_{\langle v_n,r\rangle})$:	List of channels
β:	Payment amount

- If the channels do not form a path from sender s to receiver r, abort.
- If there is a channel $(c_{\langle v_i,v_{i+1}\rangle}, \beta_i, \beta_{i+1}, f_i, f_{i+1}, p_i, p_{i+1})$, $i = 0, 1, \dots, n$ ($s = v_0$ and $r = v_{n+1}$), for which $\beta_i < send_i$, abort.
- Update each channel in the path: $(c_{\langle v_i,v_{i+1}\rangle}, \beta_i - send_i, \beta_{i+1} + send_i, f_i, f_{i+1}, p_i, p_{i+1})$, $i = 0, 1, \dots, n$.

Recursive definitions for a payment path $(s = v_0, v_1, \dots, v_n, r = v_{n+1})$.
$send_\ell$, $\ell = 0, \dots, n$, is the amount that node v_ℓ sends to $v_{\ell+1}$: $send_\ell = \beta + \sum_{i=\ell+1}^{n} fee_i$.
fee_ℓ, $\ell = 1, \dots, n$, is the amount that node v_ℓ charges (keeps) for forwarding the payment $fee_\ell = f_\ell + p_\ell \cdot send_\ell = f_\ell + p_\ell(\beta + \sum_{i=\ell+1}^{n} fee_i)$.
We say a sum starting from a higher index than its ending index equals zero.

Fig. 2. Operations in a VPCN. v_1 and v_2 share the VC establishing fee f_e.

Security and privacy for on-path adversaries. On-path adversaries may cause a diverse set of attacks in PCNs [2], [4], [9], [17]–[20] and with significant impact [21]. We employ VCs to defend against on-path adversaries, by bypassing corrupted nodes (cf. Figure 1). We chose to investigate three representative attacks: value privacy, relationship anonymity, and wormhole attacks [2], [4]. We chose these attacks because they are well studied in the literature and note that our approach directly generalizes to other on-path adversarial attacks, such as denial-of-service attacks [9], which we show in [28, Section V-D]

Optimal adversarial strategy. We assume that the adversary has a budget $\mathcal{B}$ for corrupting nodes and uses a deterministic function for selecting which nodes to attack, based on the budget and public information about the PCN. Honest nodes are not aware of $\mathcal{B}$ or the set of nodes $\mathcal{X}$ attacked by the adversary, but use public information about the PCN to estimate the adversarial strategy and $\mathcal{B}$ (e.g., as a fraction of the total PCN capacity). This in turn outputs a set of *potentially* corrupted nodes, say $\tilde{\mathcal{X}}$, where likely $\mathcal{X} \neq \tilde{\mathcal{X}}$.

We choose a strategy to estimate $\tilde{\mathcal{X}}$, such that they are the optimal set of nodes for an adversary to corrupt, i.e., the largest fraction of payments is affected using a fixed budget, based on previous works [21]–[24]. This approach allows the honest nodes to deny an adversary exactly this optimal placement within the graph by bypassing the nodes in $\tilde{\mathcal{X}}$. By denying adversaries the nodes where they can do the most damage, the overall security and privacy against on-path attacks is improved, while attacks become less profitable for the adversary. Moreover, establishing a direct VC between two end-users effectively prevents on-path attacks *regardless of the adversarial strategy*. There are many other strategies, our approach is modular and can be used to study any other adversarial strategy to compute $\tilde{\mathcal{X}}$. Our algorithms use $\tilde{\mathcal{X}}$ as part of their input, thus $\tilde{\mathcal{X}}$'s computation is modular in our design.

On-path attacks. For a path $s - u_1 - \dots - u_n - r$ the attacks on value privacy, on relationship anonymity, and the wormhole attack are defined as follows.

- *Value privacy* [4]: PCN payments ensure that the transaction amount remains private to off-path corrupted users if there are only honest users along the path. This means, that if there are on-path corrupted users, value privacy does not hold anymore, as they can simply see the value and leak it to users not on the path. *Preventing this attack:* For all segments $u_i - x_1 - x_2 - \ldots - x_\ell - u_j$ of the path from s to r, where u_i, u_j are not corrupted and x_p, $p = 1, \ldots, \ell$ are corrupted, build a virtual channel from u_i to u_j.
- *Relationship anonymity* [4]: If an adversary controls two corrupted users u_1 and u_n, they can distinguish who is paying to whom. *Preventing this attack:* If $u_i - x_1 - x_2 - \ldots - x_\ell - u_j$ is a segment of the path from s to r, where u_i, u_j are not corrupted, x_p, $p = 1, \ldots, \ell$ are corrupted and $u_i \in \{s, r\} \vee u_j \in \{s, r\}$. If both s and r are part of such a segment, take one segment (there can be at most two) and build a virtual channel from u_i to u_j.
- *Wormhole attack* [2]: In PCN payments, an adversary can prevent honest users from finalizing payments and effectively steal their fees. For this, the adversary needs to control corrupted nodes on both sides of one or more honest nodes along the path. *Preventing this attack:* Identify all segments $u_i - x_1 - \ldots - x_\ell - y_1 - \ldots - y_m - z_1 - \ldots - z_n - u_j$ of the path from s to r where u_i, u_j and y_p, $p = 1, \ldots, m$ are not corrupted and where x_q, $q = 1, \ldots, \ell$ and z_r, $r = 1, \ldots, n$ are corrupted. For each segment, build one of the following virtual channels: (i) between $u_i - y_1$ (ii) between $y_m - u_j$ (iii) between $x_\ell - z_1$.

Costs of VCs. Once opened, VCs can effectively reduce the fees of payments within a VPCN, as we explained in Section II. However, to create a VC, the endpoints need to pay an establishment fee f_e. Since VCs are currently not used, there is no fee model in practice which we can use. We therefore assume that f_e of a VC over some path with capacity α to be the same as users would charge for forwarding a payment of amount α over that path. I.e., node v_j charges $fee_j = f_{j,j+1} + p_{j,j+1} \cdot (\alpha + \sum_{k=j+1}^{n} fee_k)$, for $j = 1, \ldots, n$. We discuss other potential fee models in [28, Section VII] and note that f_e is modular in our model.

Optimization goal. Our objective is to set up VCs such that the cost for routing a set of transactions is minimized, and no transaction traverses a path prone to an attack. Definition 2 states our optimization goal and Theorem 1 states its hardness. We prove Theorem 1 in [28, Section III] by reduction from the minimum-length disjoint paths problem [30].

Definition 2 (VPCN cost optimization)**.** Given a VPCN $\mathcal{G}$, a set of transactions $\mathcal{T}$, an estimated strategy of an on-path adversary to corrupt nodes, and the estimated budget of the adversary $\mathcal{B}$ for doing so, minimize the cost for routing the transactions in $\mathcal{T}$, such that no transaction is traversing a path that is prone to a given attack. If $\mathcal{B} = 0$, our goal is to minimize the routing fees.

Theorem 1. The VPCN cost optimization problem is NP-hard.

Proof. We reduce an instance of the (NP-complete) minimum-length disjoint paths (MLDP) problem [30] to an instance of our VPCN problem. Consider an instance of the MLDP problem, i.e., an arbitrary directed graph $G = (V, E)$ such that $weight_G(u, v) = 1$, for all $(u, v) \in E$, and two source destination pairs (s_1, d_1) and (s_2, d_2) (two pairs are enough to render the problem NP-complete [30]).

We now build an instance of the VPCN problem. We define $G' = (V, E \cup E')$, where $E' = \{(x, y) \mid (y, x) \in E \wedge (x, y) \notin E\}$, such that $weight_{G'}(x, y) = 1 + \varepsilon$, $\varepsilon = 1/|E|$, if $(x, y) \in E$ and $weight_{G'}(x, y) = 0$ if $(x, y) \in E'$. Thus, for every pair of nodes x, y in G' either $\{(x, y), (y, x)\} \in G'$ or x and y are not adjacent. The payment channels are hence defined as all the pairs of nodes x, y in G' such that $\{(x, y), (y, x)\} \in G'$, with capacity $weight_{G'}(x, y) + weight_{G'}(y, x)$, base fee equal to ε and proportional fee equal to 0. We consider the set of transactions, in the form of (source, destination, amount), to be $\{(s_1, d_1, 1), (s_2, d_2, 1)\}$. We also, assume that creating virtual channels is not possible, as the problem only becomes harder by including them. We set c_{tr}, the target percentage of successful transactions, to 1 (all should be executed).

A solution to the VPCN problem gives the minimum cost payment path for the two transactions. This set of payment paths in G' is using only edges that appear in G, as we set the capacity of the extra edges to zero and since the edge weights can accommodate for only one payment path, it does so with minimum length in G and also the paths are edge disjoint (the capacity suffices for only one transaction). Therefore a solution of the VPCN problem (payment paths) is exactly a solution of the minimum-length disjoint paths problem. □

IV. An efficient greedy algorithm

In the following, we present an efficient greedy algorithm with low running time while ensuring high-quality channel allocations (see also the upcoming evaluation in Section V). Before investigating our overarching optimization goal of Section III, where we aim to prevent any attacks by on-path adversaries and minimize routing fees, our algorithm will optimize for the following goals individually: preventing (i) relationship anonymity attacks, (ii) wormhole attacks, (iii) value privacy attacks, and (iv) minimizing routing fees.

Our greedy algorithm is given as input the PCN, the payments that are to be carried out, and the optimization goal, which can be one of the optimization goals (i) to (iv). Each payment consists of a *sender*, a *receiver*, some *value* and the number of times this payment is carried out (*repetition*). The algorithm will iterate over the list of payments and for each payment, try to find the cheapest path(s) in terms of fees using Dijkstra's algorithm with enough capacity to route it (abstracted as *generate_paths*). Then, the algorithm will compute which nodes to bypass (abstracted as *compute_nodes_to_bypass*) in order to prevent on-path adversary attacks (optimization goals (i) to (iii)) or to minimize routing fees (optimization goal (iv)). Finally, the algorithm constructs virtual channels to bypass these nodes and conducts the payments (abstracted as *construct_vcs* and *conduct_payments*).

This algorithm can be applied locally, by individual nodes who do not know about any payments other than their own. We give a high-level pseudocode of this approach in Algorithm 1.

Algorithm 1 High level greedy algorithm

```
function GREEDY_VC_ALGORITHM(pcn, payments, optimization_goal)
  for (sender, receiver, value, repetition) in payments do
    //find cheapest path(s) with capacity to route each repeat payment
    paths ← generate_paths(pcn, sender, receiver, value, repetition)
    for (path, amount) in paths do
      //Compute the nodes which to bypass, based on optimization_goal
      nodes_byp. ← compute_nodes_to_bypass(path, optimization_goal)
      //Build virtual channels over these nodes
      (pcn, new_path) ← construct_vcs(pcn, path, nodes_byp., amount)
      //Conduct payments and update channel balances
      pcn ← conduct_payments(pcn, new_path, amount)
```

While we will present concrete pseudocode for implementing *compute_nodes_to_bypass* to achieve each optimization goal (i) to (iv) in Algorithms 5 to 8 in [28, Section V-C]. we give a short outline here. For relationship anonymity, it is sufficient to greedily bypass corrupted nodes adjacent to either the sender or the receiver, along short paths. To prevent the wormhole attack, corrupted nodes need to be bypassed such that there are no honest nodes encased by corrupted nodes. For value privacy, all corrupted nodes need to be bypassed.

Regarding fee optimization, we recall that the fee for opening a VC of capacity α is the same as routing a payment of amount α via that path. In our model, this means that it is cheaper to create a VC between sender and receiver, as soon as we carry out a payment on a path more than once. In Section IV-B, we show that there is a synergy between these goals and that opening VCs is beneficial to all goals. Our greedy algorithm runs efficiently on commodity hardware. The time complexity of Algorithm 1 is $\Theta(T \cdot (|E| + |V| \log |V| + D))$, where D is the diameter (cf. [28, Section V-A]).

A. *Example*

To demonstrate our approach, we find the solutions of both the ILP, i.e., the optimal solution computed by an integer linear program (cf. Section VI), and the greedy algorithm on a small example graph. The graph is shown in Figure 3 and consists of two hub-like nodes H_1 and H_2, and four client nodes A, B, C and D. We say that each channel charges a base fee of 1, a proportional fee of 0.001, and has a capacity of 10k, distributed evenly among both users. The transactions that are executed have all values of 10; there are three transactions from A to C, one transaction from A to B, and one transaction from B to C. Further, we assume that H_1 is a malicious node. We chose this graph because it resembles the hub-and-spoke topology of the Lightning Network [31]. The transaction values and fees are chosen for readability: In the fees below, the integer part of the number shows the sum of the base fees, and the fractional part is the sum of the proportional fees.

In Figure 3, we show the VCs constructed by the greedy approach in color and note that they are the same as in the ILP (optimal solution). The optimal solution is to build a virtual channel VC$_1$ of capacity 40 between A and B, another virtual channel VC$_2$ of capacity 40 between B and C, and finally a third virtual channel VC$_3$ of capacity 30 between A and C. Finally, these VCs are used for routing the transactions. The total cost on fees is 3.11, which comes from three times the base fee of 1 (which is 3) and three times the proportional fee (twice for capacity 40, once for capacity 30), which is around .11 for creating the VCs. Then, all sender-receiver pairs are directly connected, so there are no additional routing fees.

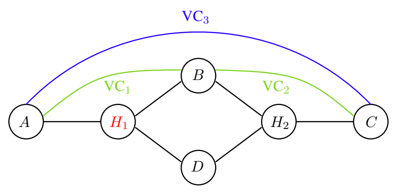

Fig. 3. Results of the ILP and greedy algorithms run on a small sample graph.

In the greedy approach, the same VCs are constructed. Since this algorithm greedily creates the best VCs for each sender-receiver pair individually, VC$_1$ does not have enough capacity and has to be constructed twice. This incurs an extra base fee of value 1. Since the overall VC capacity does not change, the proportional fee of .11 remains, totaling 4.11 in fees.

Finally, if we look at the fees of routing these payments without any s, the total amount spent on fees is 11.11. Since we now need to route the payment from A to C three times over the path, the intermediaries charge a base fee each time, resulting in 9 coins alone. Furthermore, the payments without VCs are prone to value privacy attacks by H_1.

B. *Synergy among objectives*

From the definitions of the three different attacks and the strategies of how to prevent them (cf. Section III), it becomes apparent that there are synergies among the objectives, and some strategies entail others. More concretely, preventing value privacy attacks also prevents attacks on relationship anonymity and wormhole attacks. Optimizing for fees prevents all three security and privacy attacks. We prove this synergy here and confirm these results in our evaluation in Section V.

Let $V(p), R(p), W(p), F(p)$ each be a minimal set (out of potentially many sets) of nodes that are required to be bypassed for preventing value privacy attacks, relationship anonymity attacks, and wormhole attacks, as well as for optimizing the fees, respectively. Section III shows how these sets are derived and a programmatic definition can be found in [28, Algorithms 5 to 8 in Section V-C]. For a given path p, if the nodes of these sets (or a superset) are bypassed, the corresponding attack is prevented. Since $V(p)$ and $F(p)$ both contain all adversarial nodes, optimizing for these objectives also prevents any other attack relying on on-path adversaries, such as denial-of-service attacks [9].

Theorem 2. For any path p, $F(p) \supseteq V(p) \supseteq R(p)$ and $F(p) \supseteq V(p) \supseteq W(p)$.

Proof. $V(p)$ is the set containing all corrupted nodes on the path p. Otherwise, there would be a corrupted node on the path that is not bypassed and value privacy attacks would not be prevented. The two sets $R(p)$ and $W(p)$ do not have any honest nodes by definition (honest nodes do not need to be bypassed). $R(p)$ and $W(p)$ contain thus only malicious nodes, but they do not contain all the malicious nodes of path p. Consider for example path $s - c_1 - h_1 - c_2 - h_2 - c_3 - r$ (c_i representing corrupted and h_i honest nodes), where c_2 is in $V(p)$, but not in $R(p)$. It follows that $V(p) \supseteq R(p)$ and, similarly, $V(p) \supseteq W(p)$.

It remains to show that $F(p) \supseteq V(p)$. For this, we merely observe that the set $F(p)$ contains every node on the path p, which includes every corrupted node, which is $V(p)$. $\square$

V. Empirical evaluation

We conducted extensive simulations to shed light on the optimized deployment of virtual channels, as well as to study the performance of our greedy algorithm (Section IV).

A. Input data preparation and methodology

Graph model and data. Recalling our model in Section III, let $\mathcal{G} := (\mathcal{V}, \mathcal{E} := \mathcal{E}_p \cup \mathcal{E}_v)$ be our VPCN graph with $\mathcal{V}$ the set of nodes, $\mathcal{E}_p$ the set of PCs and $\mathcal{E}_v$ (initially $\emptyset$) the set of PCs. We conduct our experiments on a snapshot of the Lightning Network (LN) from March 4, 2021 [32]. The (largest connected component of the) graph contains 33k channels and 8k nodes that are part of at least one channel. For each channel, we read the capacity, the base, and the relative fee. The total network capacity is 1,167.4 BTC, the average base fee is 3,165 msat (millisatoshi), the average relative fee rate is 32,417 millionth of the satoshis transferred (one BTC is 100M satoshis). Due to the nature of PCNs the individual balance of each user remains private, a common limitation for works investigating PCNs. We assume that each channel capacity is initially evenly distributed between both nodes.

Payments. For payments we sample $r\_pay = 100$ random sender-receiver pairs in the graph and uniform payment amounts $val \in [1, 10]$ satoshis, modeling a micro-payment setting as the average channel capacity is 2.6M satoshis.

Constructing VCs. We create VCs on top of the PCN, both for direct payments between endpoints and for routing other payments through the VCs. Users can charge fees (i) for establishing the VC if they are intermediaries or (ii) for routing payments through the VC if they are endpoints. There exists no fee model for VCs in practice. Therefore, we interpret the base fee as what a hop charges for actively participating in the protocol, and the fee rate as what a hop charges for locking up α coins, i.e., the opportunity cost of that node. We model the establishment fee of a VC with capacity α to be the same as routing a payment of α coins via that path (see Section III). Our solution is modular, other fee models can be used, and we discuss other possible fee models in [28, Section VII]. If a VC is established, its routing fee is set to the fee of the initiating endpoint's underlying channel.

Corruption model. Honest nodes assume there is an attacker who has an estimated budget and who corrupts nodes according to an estimated strategy. In order to corrupt a node $v \in \mathcal{V}$, we say an attacker needs to spend the money that this node $v \in \mathcal{V}$ has locked up in its neighboring channels, i.e., $capacity\_locked(v) := \sum_{w \in \mathcal{V} \setminus \{v\}} (pc_{\langle v,w \rangle}.\beta_1)$.

The assumption that an attacker corrupts nodes that are most beneficial to it, while being cheap to place, is based on previous works [21]–[24]. We parameterize the estimated adversary budget adv_budget as a percentage of the total capacity in all edges of $\mathcal{G}$. The absolute adversary capacity is $adv\_capacity := adv\_budget \cdot total\_network\_capacity$. To choose the best-placed nodes, the adversary computes random payment paths and selects those nodes that appear most often on these paths. Note that no payment is actually carried out, only the paths are computed to find the most used nodes. Let P be a list of num_pay (e.g., 500) randomly chosen payment paths (i.e., paths of connected edges) in $\mathcal{G}$. For every node $v \in \mathcal{V}$, we let $occ(v)$ be the number of their occurrence in P. We define the following cost-benefit ratio for every node v as follows: $cost\_benefit(v) := \frac{occ(v)/num\_pay}{capacity\_locked(v)/adv\_capacity}$.

Let l be a list of every node $v \in \mathcal{V}$ sorted by their cost-benefit ratio in descending order. We determine the list of all corrupted nodes C iterating over l and adding those for which the following condition holds after adding them: $\sum_{n \in C}(capacity\_locked(n)) \leq adv\_budget$

Repeating payments needed for VCs to be cost efficient. If a VC is used only once, it will never cost fewer fees than routing a payment directly through the underlying PCs. Therefore, we investigate the effect of conducting our r_pay payments multiple times.

Measuring fees, security and privacy. The cost of routing the r_pay payments through the PCN without PCs is denoted as $route\_pcn$. The cost of establishing the PCs to prevent a certain type of attack is denoted as $establish\_vc$. The routing cost when using the PCs is $route\_vc$. We are interested in how the following ratio progresses as we increase the number of times that payments are repeated: $fee\_ratio := \frac{establish\_vc + route\_vc}{route\_pcn}$. We further measure how many payment paths are prone to a certain attack, with and without the VCs.

B. Results

We first study the effect that opening VCs while optimizing for each individual goal has on the other goals and on the fees. We fix an adversary budget and corrupt the nodes according to our corruption model. For each payment, we then use VCs (i) to optimize for security or privacy by preventing one of these attacks completely if that payment path is prone to that attack or (ii) to optimize for fees, both according to the algorithms outlined in Section IV. Finally, we measure the impact this has on the two other attacks as well as on the fees. The full algorithm pseudocode can be found in [28, Section V-C].

In our experiment, we investigate value privacy, relationship anonymity, and wormhole attacks. For these experiments we need to choose an adversary budget that results in meaningful security threats from all these attacks. By meaningful we mean that some of our paths (not 0 and not all of them) are susceptible to each of the three different attacks. For this, we

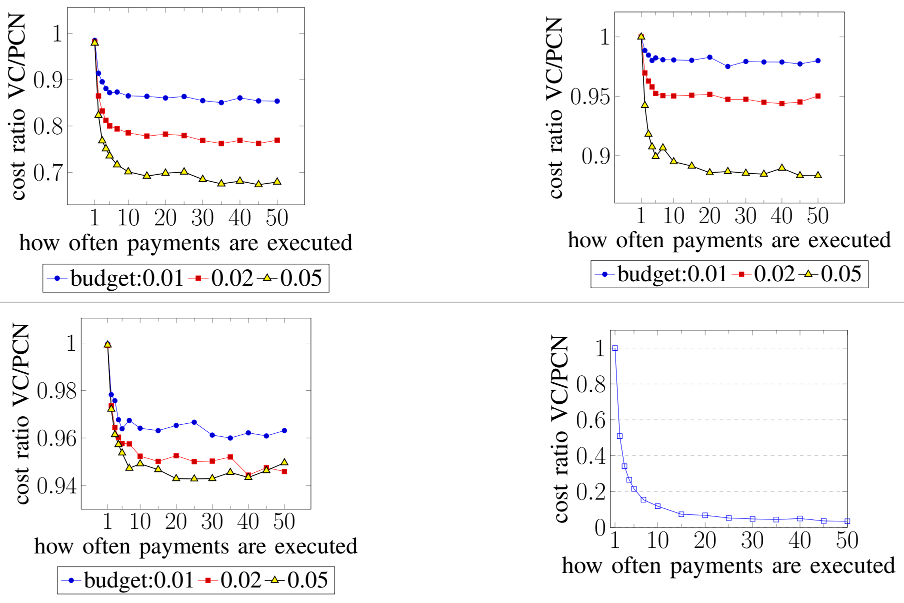

Fig. 4. Optimizing for value privacy (top left), relationship anonymity (top right), wormhole attack (bottom left), and fees (bottom right)

need to compute the percentage of paths that are prone to which attack for different adversary budgets. We expand on this in [28, Section VI-C] and end up choosing 1, 2 and 5%.

Q1: How does preventing one attack affect the money spent on fees? We first measure the cost of routing payments through the PCN without VCs as a baseline. Then, we construct VCs, optimizing for value privacy, relationship anonymity, and wormhole attacks. After constructing the VCs, we measure the cost again. We measure the ratio according to our definition in Section V-A. The VCs are constructed according to the corrupted nodes on the payment paths. Since these paths are randomly chosen and thus different for every run, we conduct each experiment 100 times and compute the average, with the results shown in Figure 4. We observe that for all three budgets, the cost ratio starts out around 1 for one payment. As the number of repeating payments goes up, the cost ratio decreases because the VCs are more effective. Additionally, the more nodes are corrupted and need to be bypassed, the more VCs are constructed and the better this ratio becomes. For relationship anonymity, this ratio goes down to 0.88, for wormhole attack to 0.95, for value privacy to 0.68.

We observe that bypassing nodes to prevent each attack has a positive effect on the fee ratio if the payments are repeated more than once. The best effect can be seen in preventing value privacy attacks. Also, the ratio goes down more steeply for the first 10 payments, afterwards the effect is more flat.

Q2: How does optimizing for fees affect the money spent on fees? Similar to when optimizing for security and privacy goals, we observe a steep decline in the ratio of money spent for fees when constructing VCs to the money spent for fees if we do not construct VCs. The decline slows down later. This ratio halves if there are 2 sequential payments and continues to drop to 0.04 for 50, after which it is almost flat.

Q3: How do the different optimization strategies affect security and privacy? We already compared the effect of

Fig. 5. How many paths are prone to different attacks when optimizing for different goals for an adversary budget of 0.05.

			Paths prone to (PCN; VC)		
Optimizing	# VC	Avg VC length	VP attacks	RA atk.	WH atk.
VP	126	3.1	97; 0	21; 0	35; 0
RA	21	3.6	97; 84	21; 0	35; 28
WH	33	2.2	97; 97	21; 13	35; 0
Fees	100	5.6	97; 0	21; 0	35; 0

the strategies optimizing the different goals on the fees. Now we want to evaluate the effect that they have on the security and privacy goals. For this, we measure how many of our payment paths are prone to each of the different attacks. Then we construct the VCs optimizing each goal and measure how many paths are prone then. In Figure 5 we show for each optimization strategy, (i) how many VCs are constructed, (ii) the average length of each VC, and (iii) for each attack type two values $x; y$, where x is the percentage of paths prone to the attack before building VCs and y is the percentage of paths prone to the attack after building VCs. We notice that optimizing for value privacy also prevents the attacks on relationship anonymity and the wormhole attack. Furthermore, optimizing for fees prevents all three attacks we investigate. These results are in line with Section IV-B.

VI. Exact Algorithm

We present an exact solution to the VPCN cost optimization problem (Definition 2) by modelling it as an Integer Linear Program (ILP). This method computes a globally optimal solution and for that requires information from the whole network. Our goal is to provide a theoretical tool that will help in PCN design or for measuring the quality of practical and local solutions, like the greedy algorithm of Section IV.

Our first challenge is to define the objective function to be optimized. We have three objectives: (a) minimize routing fees, (b) minimize VC creation costs, and (c) maximize the volume of successful transactions. We define the objective function combining items (a) and (b), and form the ILP as a minimization problem. Item (c) will be converted to a constraint (this is common in multi-objective optimization), requiring that the success volume ratio is above a threshold given in the input. The second challenge is to define the invariants that a solution should respect, and based on them specify the ILP's variables and constraints. We identify the following invariants: (i) at most one path is used for routing a transaction, (ii) the transaction success volume ratio should be above the given percentage, (iii) capacities of PCs and VCs are respected, (iv) a VC between i, j over k should be bidirectional, (v) a VC is constructed if and only if it is used for routing a transaction or for constructing a higher-order VC, (vi) payment paths prone to attacks of on-path adversaries are not selected in the ILP solution. We will now define the ILP formally. The full version of this section is available in [28].

Input. The input needed to define the ILP is a PCN (as defined in Section II) including all PC attributes, a set of T transactions $\mathcal{T} = \{transaction_t = (s_t, d_t, trans_t)\}_{t\in[1,T]}$, i.e., (source, destination, amount), a constant $c_{tr} \in [0, 1]$ indicating the required minimum success volume ratio, i.e., if $c_{tr} = 1$ all transactions must be executed, and the the set $\tilde{\mathcal{X}}$ of estimated (by the honest nodes) set of corrupted nodes.

Let $ch_{ij}.bf$ (f_i in Definition 1) and $ch_{ij}.pf$ (p_i in Definition 1) denote the base and proportional forwarding fee of a PC ($ch_{ij} = pc_{ij}$) or a VC ($ch_{ij} = vc^k_{ij}$, where k is the intermediary node), and $pc_{ij}.capacity$ denotes the PC capacity (β_i in Definition 1). We set the VC fees to be equal to those of the underlying initial PC, i.e., the fees of vc^k_{ij} match those of pc_{ik} if vc^k_{ij} is built over pc_{ik}.

Since a VC vc^k_{ij} can be constructed over any combination of two adjacent channels ch_{ik} and ch_{kj}, we assume a recursive structure of VCs and bound the levels of recursion by the input parameter w. Level-0 VCs are constructed over two adjacent PCs. Level-m VCs, $0 < m \leq w$, are constructed over a level-$(m-1)$ VC and an adjacent payment or virtual channel of level at most $m-1$. We define the input VPCN as a directed graph over the set of all nodes V and two sets of edges (channels): E_{PC} (PCs) and E_{VC} (all possible VCs). We define each edge (channel) in E_{VC} by the triple (i, j, id), where i, j are the endpoints and id is a unique edge identifier.

Constants, variables, and macros. We will use three sets of integer variables. The first set includes binary variables that indicate that $transaction_t$ is routed via path P ($path_P(trans_t)$), the second set indicates the capacity of a VC ($vc^k_{ij}.capacity$), and the third set indicates whether a VC exists ($exists\_vc^k_{ij}$).

Let $\mathcal{P}(s_t, d_t)$ be a list of all the paths from a sender s_t to a receiver d_t for $transaction_t$ in the graph $(V, E_{PC} \cup E_{VC})$. The variable $path_P(trans_t) \in \{0, 1\}$ indicates whether $transaction_t$ is routed through path $P \in \mathcal{P}(s_t, d_t)$. For a channel ch_{ij} and $transaction_t$, we define the macro $used(ch_{ij}, t) = \sum_{P\in\mathcal{P}(s_t,d_t)} path_P(trans_t) \cdot In(ch_{ij}, P)$, where $In(ch_{ij}, P)$ is a constant indicating whether $ch_{ij} \in P$. When ch_{ij} is used by a payment route for $transaction_t$ then $used(ch_{ij}, t)$ is 1 (only one path is used for $transaction_t$ due to constraint C1) and otherwise it is 0. Let $routing\_fee(t, P, ch_{ij})$, $ch_{ij} \in \{pc_{ij}, vc^k_{ij}\}$ be the routing fees charged to channel $ch_{ij} \in P$ for $transaction_t$. We note that $routing\_fee(t,\ P, ch_{ij})$ is computed as in Section II if $ch_{ij} \in P$ and is zero otherwise. Let $routing\_cost_{ch_{ij}} = \sum_{t=1}^{T} \sum_{P\in\mathcal{P}(s_t,d_t)} routing\_fee(t, P, ch_{ij}) \cdot path_P(trans_t)$ be the routing fees that are charged for the transactions that traverse channel $ch_{ij} \in \{pc_{ij}, vc^k_{ij}\}$.

Paths including nodes in $\tilde{\mathcal{X}}$ (estimated to be corrupted) should not be used for routing payments, thus we exclude those paths from $\cup_{t=1}^{T} \mathcal{P}(s_t, d_t)$. For instance, for value privacy, we exclude every path P such that $x \in P$, for all $x \in \tilde{\mathcal{X}}$.

We denote the capacity of $vc^k_{ij} \in E_{VC}$ with $vc^k_{ij}.capacity$. We define the VC creation cost as $vc^k_{ij}\_creation\_cost = exists\_vc^k_{ij} \cdot vc^k_{ij}.bf + vc^k_{ij}.pf \cdot vc^k_{ij}.capacity$, where $exists\_vc^k_{ij}$ is a binary variable indicating if vc^k_{ij} exists. If vc^k_{ij} is used for routing transactions ($exists\_vc^k_{ij} = 1 \land vc^k_{ij}.capacity > 0$), then the creation cost is $vc^k_{ij}.bf + vc^k_{ij}.pf \cdot vc^k_{ij}.capacity$. Due to constraint C5, if vc^k_{ij} is not used in a payment path, $exists\_vc^k_{ij} = 0$ and because $vc^k_{ij}.creation\_cost$ appears in the objective function, which we want to minimize, $vc^k_{ij}.capacity$ will be reduced to 0 in any minimal solution. Thus $vc^k_{ij}.creation\_cost = 0$ when vc^k_{ij} is not used in a payment path.

Objective. The objective is to minimize routing and VC creation costs: $\min \sum_{pc_{ij}\in E_{PC}} routing\_cost_{pc_{ij}} + \sum_{vc^k_{ij}\in E_{VC}} (routing\_cost_{vc^k_{ij}} + vc^k_{ij}\_creation\_cost)$.

Constraints. We define five constraints that collectively express the invariants:

(C1) At most one path can be used for routing a transaction: $\sum_{P\in\mathcal{P}(s_t,d_t)} path_P(trans_t) \leq 1$, $\forall t \in [1, T]$.

(C2) The percentage of successful transaction volume should be at least the success volume ratio $c_{tr} \in [0, 1]$: $\sum_{t=1}^{T} trans_t \cdot \sum_{P\in\mathcal{P}(s_t,d_t)} path_P(trans_t) \geq c_{tr} \sum_{t=1}^{T} trans_t$.

(C3) VC and PC capacities should be respected. For every $ch_{ij} \in E = E_{PC} \cup E_{VC}$ we require that

$$\Big[\sum_{vc^j_{ix}\in E_{VC}:vc^j_{ix}=(ch_{ij},\bullet)} (vc^j_{ix}.creation\_cost + vc^j_{ix}.capacity)\Big] + routing\_cost_{ch_{ij}} + trans\_amount(ch_{ij}) \leq ch_{ij}.capacity$$

where $trans\_amount(ch_{ij}) = \sum_{t=1}^{T} trans_t \cdot used(ch_{ij}, t)$ and $\bullet$ is any channel between j and x that can form vc^j_{ix}.

(C4) A VC between nodes i, j over k must exist in both directions (i, k, j) and (j, k, i): $exists\_vc^k_{ij} = exists\_vc^k_{ji}$

(C5) A VC exists, only if it is used for routing a transaction or to construct a VC of higher recursive order:
$exists\_vc^k_{ij} \leq \sum_{t=1}^{T} used(vc^k_{ij}, t) + \sum\limits_{vc^\ell_{sr}\in rec(vc^k_{ij})} exists\_vc^\ell_{sr}$
and $exists\_vc^k_{ij} \geq$
$\frac{1}{T+|E_{VC}|}\big(\sum_{t=1}^{T} used(vc^k_{ij}, t) + \sum\limits_{vc^\ell_{sr}\in rec(vc^k_{ij})} exists\_vc^\ell_{sr}\big)$,
where $rec(vc^k_{ij})$ includes all $vc^\ell_{sr} \in E_{VC}$ that are built over vc^k_{ij}. The first inequality enforces $exists\_vc^k_{ij} = 0$ if the VC is not used and the second one $exists\_vc^k_{ij} = 1$ otherwise.

Output. We can determine all created VCs by checking $exists\_vc_{ij}^{k}$ and if $transaction_t$ is successful by the value of $\sum_{P\in\mathcal{P}(s_t,d_t)} path_P(trans_t) \in \{0,1\}$.

ILP experiments. The ILP has exponentially many variables and constraints (asymptotically), since it needs to decide which subset of all paths minimizes the objective. We implemented the ILP [33] using Python and Gurobi [34]. We were able to run it with at most 15 nodes, 30 channels, and 5 transactions (cf. [28, Section IV] for results and discussion on how to make the ILP tractable by sacrificing accuracy).

VII. Conclusion

Motivated by the potential benefits of virtual channels to reduce transaction fee costs as well as to improve security and privacy guarantees in PCNs, we presented a first systematic study of the virtual channel setup problem. We have shown that the problem can be formulated as an optimization problem and proved that the problem is NP-hard. We presented a fast greedy algorithm and using simulations on the Lightning Network, we confirmed the benefits of our optimization approach. We also modeled the VPCN cost optimization problem as an integer linear program (ILP) for obtaining exact solutions.

We believe that our work opens several interesting avenues for future research, such as studying different fee models, the effect of timing, i.e., adding time frames in which transactions are to be executed, VC lifetimes, or more dynamic strategies for the attacker or how the attacker can best react to the countermeasures proposed in this work.

Acknowledgements. This work has been supported by the European Research Council (ERC) under the Horizon 2020 research (grant 771527-BROWSEC); by the Austrian Science Fund (FWF) through the SpyCode SFB project F8510-N and the project CoRaF (agreement 2020388); by CoBloX; by the Austrian Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development and the Christian Doppler Research Association through the Christian Doppler Laboratory Blockchain Technologies for the Internet of Things (CDL-BOT); by the German Federal Ministry of Education and Research (BMBF) grant 16KISK020K (6G-RIC); by MCIN/AEI/10.13039/501100011033 and European Union NextGenerationEU/PRTR through PRODIGY Project (TED2021-132464B-I00), grant PID2022-142290OB-I00 and grant IJC2020-043391-I.

References

[1] J. Poon and T. Dryja, “The Bitcoin Lightning Network:,” 2016.
[2] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei, “Anonymous Multi-Hop Locks for Blockchain Scalability and Interoperability,” in *NDSS*, 2019.
[3] L. Aumayr, P. Moreno-Sanchez, A. Kate, and M. Maffei, “Blitz: Secure multi-hop payments without two-phase commits,” in *30th USENIX Security Symposium*, 2021.
[4] G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, and S. Ravi, “Concurrency and privacy with payment-channel networks,” in *ACM SIGSAC Conference on Computer and Communications Security*, 2017.
[5] E. Heilman, L. Alshenibr, F. Baldimtsi, A. Scafuro, and S. Goldberg, “Tumblebit: An untrusted bitcoin-compatible anonymous payment hub,” in *Network and Distributed System Security Symposium*, 2017.
[6] E. Tairi, P. Moreno-Sanchez, and M. Maffei, “A^{2}L: Anonymous Atomic Locks for Scalability in Payment Channel Hubs,” in *IEEE SP*, 2021.
[7] C. Egger, P. Moreno-Sanchez, and M. Maffei, “Atomic Multi-Channel Updates with Constant Collateral in Bitcoin-Compatible Payment-Channel Networks,” in *ACM CCS*, 2019.
[8] M. Jourenko, M. Larangeira, and K. Tanaka, “Payment Trees: Low Collateral Payments for Payment Channel Networks,” in *Financial Cryptography and Data Security*, 2021.
[9] S. Tochner, A. Zohar, and S. Schmid, “Route hijacking and dos in off-chain networks,” in *Advances in Financial Technologies (AFT)*, 2020.
[10] S. Dziembowski, L. Eckey, S. Faust, and D. Malinowski, “Perun: Virtual Payment Hubs over Cryptocurrencies,” in *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 106–123, May 2019.
[11] L. Aumayr, M. Maffei, O. Ersoy, A. Erwig, S. Faust, S. Riahi, K. Hostáková, and P. Moreno-Sanchez, “Bitcoin-compatible virtual channels,” in *2021 IEEE Symposium on Security and Privacy*, 2021.
[12] M. Jourenko, M. Larangeira, and K. Tanaka, “Lightweight virtual payment channels,” in *Cryptology and Network Security, CANS*, 2020.
[13] L. Aumayr, P. Moreno-Sanchez, A. Kate, and M. Maffei, “Breaking and fixing virtual channels: Domino attack and donner,” in *Network and Distributed System Security (NDSS) Symposium*, 2023.
[14] G. Avarikioti, R. Scheuner, and R. Wattenhofer, “Payment networks as creation games,” in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, 2019.
[15] Z. Avarikioti, L. Heimbach, Y. Wang, and R. Wattenhofer, “Ride the lightning: The game theory of payment channels,” in *Financial Cryptography and Data Security*, 2020.
[16] O. Ersoy, S. Roos, and Z. Erkin, “How to profit from payments channels,” in *Financial Cryptography and Data Security*, 2020.
[17] W. Tang, W. Wang, G. Fanti, and S. Oh, “Privacy-utility tradeoffs in routing cryptocurrency over payment channel networks,” *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2020.
[18] U. Nisslmueller, K.-T. Foerster, S. Schmid, and C. Decker, “Toward active and passive confidentiality attacks on cryptocurrency off-chain networks,” in *ICISSP*, 2020.
[19] S. Tripathy and S. K. Mohanty, “Mappcn: Multi-hop anonymous and privacy-preserving payment channel network,” in *International Conference on Financial Cryptography and Data Security*, 2020.
[20] EmelyanenkoK, “Payment channel congestion via spam-attack,” in *https://github.com/lightningnetwork/lightning-rfc/issues/182*, 2020.
[21] S. Tikhomirov, P. Moreno-Sanchez, and M. Maffei, “A quantitative analysis of security, anonymity and scalability for the lightning network,” in *IEEE European Symposium on Security and Privacy Workshops*, 2020.
[22] G. Kappos, H. Yousaf, A. Piotrowska, S. Kanjalkar, S. Delgado-Segura, A. Miller, and S. Meiklejohn, “An Empirical Analysis of Privacy in the Lightning Network,” in *Financial Cryptography and Data Security ’21*.
[23] R. Pickhardt, S. Tikhomirov, A. Biryukov, and M. Nowostawski, “Security and privacy of lightning network payments with uncertain channel balances,” *CoRR*, vol. abs/2103.08576, 2021.
[24] S. P. Kumble, D. Epema, and S. Roos, “How lightning’s routing diminishes its anonymity,” in *The 16th International Conference on Availability, Reliability and Security*, ARES 2021, 2021.
[25] P. McCorry, S. Bakshi, I. Bentov, S. Meiklejohn, and A. Miller, “Pisa: Arbitration outsourcing for state channels,” in *ACM Conference on Advances in Financial Technologies (AFT)*, pp. 16–30, 2019.
[26] G. Avarikioti, E. K. Kogias, and R. Wattenhofer, “Brick: Asynchronous state channels,” in *Financial Cryptography and Data Security*, 2021.
[27] J. Khamis and O. Rottenstreich, “Demand-aware channel topologies for off-chain payments,” in *COMSNETS*, 2021.
[28] L. Aumayr, E. Ceylan, Y. Kopyciok, M. Maffei, P. Moreno-Sanchez, I. Salem, and S. Schmid, “Optimizing virtual channel establishment in the face of on-path adversaries,” *CoRR*, vol. abs/2011.14341, 2024. Extended technical report, https://arxiv.org/abs/2011.14341.
[29] M. Green and I. Miers, “Bolt: Anonymous Payment Channels for Decentralized Currencies,” in *ACM CCS*, 2017.
[30] T. Eilam-Tzoreff, “The disjoint shortest paths problem,” *Discrete applied mathematics*, vol. 85, no. 2, pp. 113–138, 1998.
[31] P. Zabka, K.-T. Foerster, S. Schmid, and C. Decker, “A centrality analysis of the lightning network,” 2022.
[32] Fiatjaf, “lnchannels.” https://web.archive.org/web/20210612140032/https://ln.fiatjaf.com/, 2021.
[33] Y. Kopyciok, “vc-optimizer.” https://github.com/ykpyck/vc-optimizer, 2023.
[34] Gurobi Optimization, LLC, “Gurobi Optim. Reference Manual,” 2023.