Computer Communications 238 (2025) 108155

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Optimizing virtual payment channel establishment in the face of on-path adversaries[☆]

Lukas Aumayr [a,b,c,*], Esra Ceylan [d], Yannik Kopyciok [e], Matteo Maffei [c,h], Pedro Moreno-Sanchez [f], Iosif Salem [e,g], Stefan Schmid [e]

[a] *University of Edinburgh, 10 Crichton St, Edinburgh, EH8 9AB, United Kingdom*
[b] *Common Prefix, Louizis Riankour 64, Athens, 115 23, Greece*
[c] *TU Wien, Karlsplatz 13, Vienna, 1040, Austria*
[d] *University of Vienna, Universitätsring 1, Vienna, 1010, Austria*
[e] *TU Berlin, Straße des 17. Juni 135, Berlin, 10623, Germany*
[f] *IMDEA Software Institute, Campus de Montegancedo s/n, Madrid, 28223, Spain*
[g] *ZeroPoint Technologies AB, Falkenbergsgatan 3, Gothenburg, 41285, Sweden*
[h] *Christian Doppler Laboratory Blockchain Technologies for the Internet of Things, Blohmstraße 15, Hamburg, 21079, Germany*

ARTICLE INFO

Keywords:
Payment channel networks
Virtual channels
Optimization
Security
Privacy

ABSTRACT

Payment channel networks (PCNs) are among the most promising solutions to the scalability issues in permissionless blockchains, allowing parties to pay each other off-chain through a path of payment channels (PCs). However, the cost of routing transactions is proportional to the number of intermediaries since each charges a fee. Analogous to other networks, malicious intermediaries on the path can lead to security/privacy threats. Virtual channels (VCs), i.e., bridges over PC paths, mitigate the above PCN issues: Intermediaries participate only in the VC setup but in no future VC payments. However, creating a VC has a cost that must be paid out of the bridged PCs' balance. Currently, we are missing guidelines on how/where to set up VCs. Ideally, VCs should minimize transaction costs while mitigating security and privacy threats from on-path adversaries.

In this work, we address for the first time the VC setup problem, formalizing it as an optimization problem. We present an integer linear program (ILP) computing the globally optimal VC setup strategy in terms of cost, security, and privacy. We accompany this expensive ILP with a fast, greedy algorithm. Our model and algorithms can be used with any on-path adversary whose strategy can be expressed as a set of corrupted nodes. We evaluate the greedy algorithm over a snapshot of the Lightning Network (LN), the largest Bitcoin-based PCN. Our results confirm that the greedy strategy minimizes costs while protecting against security and privacy threats and may serve the LN community as guidelines for VC deployment.

1. 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 min to be confirmed.

Payment channels (PC) have emerged as one of the most promising scalability solutions, and instances such as the Lightning Network [2] 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.

☆ This is the extended version of the IFIP NETWORKING 2024 conference paper Aumayr et al. [1].
* Corresponding author at: University of Edinburgh, 10 Crichton St, Edinburgh, EH8 9AB, United Kingdom.
E-mail addresses: laumayr@ed.ac.uk (L. Aumayr), esra.ceylan@univie.ac.at (E. Ceylan), kopyciok@campus.tu-berlin.de (Y. Kopyciok), matteo.maffei@tuwien.ac.at (M. Maffei), pedro.moreno@imdea.org (P. Moreno-Sanchez), iosif.salem@zptcorp.com (I. Salem), stefan.schmid@tu-berlin.de (S. Schmid).

https://doi.org/10.1016/j.comcom.2025.108155
Received 22 December 2024; Received in revised form 12 March 2025; Accepted 27 March 2025
Available online 22 April 2025

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 [3–9] 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 opens the door to a number of security and privacy issues, such as route hijacking [10], wormhole attacks [3] or user anonymity [5], just to name a few.

Recently, the concept of virtual channels (VCs) [11–14] 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 virtual channel.

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 aim to optimally bypass adversarial nodes by constructing virtual channels over them. 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 [15–17], none of them considered virtual channels, 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 [3,5,10,18–21]. Such attacks have been shown to potentially have a severe impact in practice [22]. 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 and study three exemplary attacks. Specifically, our algorithms take as input the set of nodes that honest nodes aim to neutralize as high-value targets for an adaptive adversary. The adversary selects nodes based on a cost–benefit trade-off, maximizing their impact within a given budget. The honest nodes counter this by preemptively bypassing the most profitable nodes for the adversary. The derivation of this set by the honest nodes is orthogonal to our solutions. To demonstrate our approach, we focus on adversarial strategies that affect the largest fraction of payments [22–25] and analyze their impact on value privacy, relationship anonymity, and the wormhole attack. Note that our approach is general and allows for any adversarial strategy. While we assume an adaptive adversary selecting nodes based on a cost–benefit trade-off, our framework can also accommodate static corruption strategies, where the adversary pre-selects nodes according to any other metric.

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 PCNs software and make them accessible to the PCN users.

Paper organization. This is the extended version of the IFIP NETWORKING 2024 conference paper Aumayr et al. [1]. We introduce background knowledge on PCNs in Section 2 and present our model and problem formulation in Section 3. We present our algorithms in Section 4 (exact) and 5 (greedy), and evaluate the greedy algorithm in Section 6.

2. Background and problem overview

Payment channel networks (PCNs). A PCN [5] 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

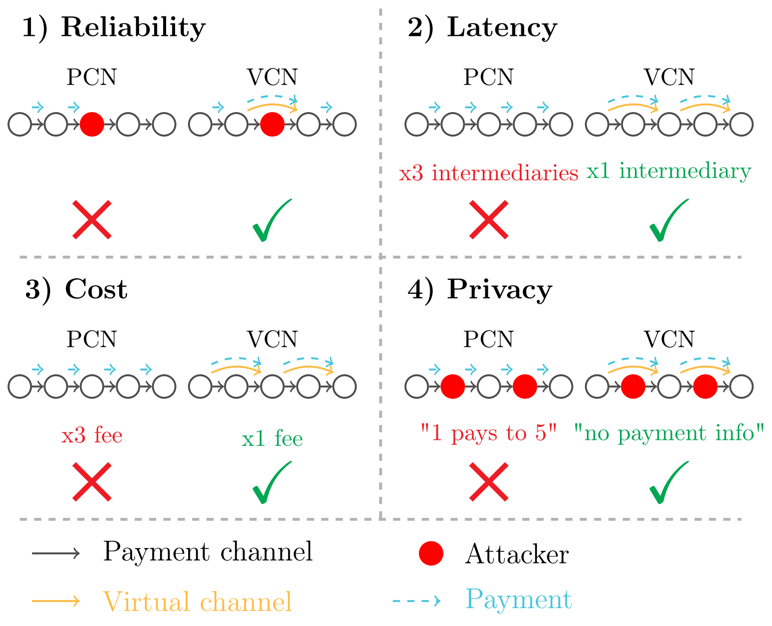


Fig. 1. Comparison between PCN and VCN.

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 $s \to u_1 \to \dots \to u_n \to r$. 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 $j = 1, \dots, n$ (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 Fig. 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 Fig. 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 [5,26] 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., [3,4], 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 Fig. 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 Fig. 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 Fig. 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 *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.

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?

3. Modeling 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 $(v, \beta^{on-chain})$ where v denotes an address of the underlying blockchain and $\beta^{on-chain}$ 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 Fig. 2.

Security and privacy for on-path adversaries. On-path adversaries may cause a diverse set of attacks in PCNs [3,5,10,18–21] and with significant impact [22]. We employ VCs to defend against on-path adversaries, by bypassing corrupted nodes (cf. Fig. 1). We chose to investigate three representative attacks: value privacy, relationship anonymity, and wormhole attacks [3,5]. 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 [10], which we later show in Section 5.4.

Optimal adversarial strategy. We assume that the adversary has a budget $\mathcal{B}$ for corrupting nodes and selects which nodes to attack using a deterministic function based on $\mathcal{B}$, public information about the PCN, and the countermeasures deployed by honest nodes. Honest nodes do not know $\mathcal{B}$ or the exact set of corrupted nodes $\mathcal{X}$ but assume a worst-case scenario, where an adaptive adversary corrupts the most profitable nodes *after* countermeasures, i.e., opening VCs, have been deployed. To minimize the adversary's impact, the best strategy for honest nodes is to preemptively bypass the high-value targets $\tilde{\mathcal{X}}$, which is the set of nodes that would be most profitable for the adversary to corrupt *in the absence of countermeasures*. The honest nodes estimate $\mathcal{B}$ (e.g., as a fraction of the total PCN capacity) to inform their selection of $\tilde{\mathcal{X}}$.

We compute $\tilde{\mathcal{X}}$ by identifying the nodes that an adversary would optimally corrupt, choosing those that maximize the fraction of affected payments within a fixed budget, based on prior work [22–25]. For more details, see *Countermeasures* in Section 6. Honest nodes apply countermeasures by opening VCs to bypass the nodes in $\tilde{\mathcal{X}}$, denying the adversary access to their most profitable targets. This improves security and privacy against on-path attacks while reducing the adversary's overall effectiveness. Notably, establishing a direct VC between two end-users eliminates on-path attacks *regardless of the adversarial strategy*.

Our approach is modular: different adversarial strategies can be incorporated into the computation of $\tilde{\mathcal{X}}$ without modifying the framework. Our algorithms take $\tilde{\mathcal{X}}$ as input, ensuring flexibility in adapting to various adversarial models.

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

- *Value privacy* [5]: 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 $u_i - x_1 - x_2 - \cdots - x_\ell - u_j$ 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* [5]: If an adversary controls two corrupted users u_1 and u_n, they can distinguish who is paying to whom. *Preventing this attack:* If $u_i - x_1 - x_2 - \cdots - x_\ell - u_j$ 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* [3]: 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–⋯–x_ℓ–y_1–⋯–y_m–z_1–⋯–$z_n - u_j$ 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 . 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 Section 7 and note that f_e is modular in our model.

Optimization goal. Our objective is to set up virtual channels such that the cost for routing a set of transactions is minimized, and no transaction is traversing a path prone to an attack. Definition 2 consolidates our optimization goal and its hardness is proven in Theorem 1.

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 the estimation of the adversary's budget is $\mathcal{B} = 0$ 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 [27] 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 [27]).

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. □

4. Exact algorithm

We present an exact solution to the VPCN cost optimization problem (Definition 2) by modeling it as an Integer Linear Program (ILP). Our first challenge is to define the objective function to be optimized. We have three objectives: (a) minimize routing fees, (b) minimize virtual channel creation costs, and (c) maximize successful transactions (either in number or in volume). We will define the objective function using 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 ratio is above a threshold given in the input. Thus, different threshold values might yield different solutions.

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 ratio should be above the given percentage, (iii) capacities of payment and virtual channels 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. From a geometric point of view, the constraints define a set (polytope) of feasible solutions and an ILP solver outputs a feasible solution (if any) within this set that produces the minimum value for the objective function, i.e. the function that expresses objectives (a) and (b) as a linear combination of the variables. We will now define the ILP formally.

Input. The input needed to define the ILP is a PCN (as defined in Section 2) including all the payment channels and their 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 or volume ratio, i.e., if $c_{tr} = 1$ all transactions must be executed, and the set $\tilde{\mathcal{X}}$ of estimated (by the honest nodes) set of corrupted nodes.

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

Since a virtual channel vc^k_{ij} can be constructed over any combination of two adjacent payment or virtual 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 virtual channels are constructed over two adjacent payment channels. Level-m virtual channels, $0 < m \leq w$, are constructed over a level-$(m-1)$ virtual channel and an adjacent payment or virtual channel of level at most $m-1$. The VPCN that we provide as part of the ILP input will be a fusion of all PCs and all possible VCs, such that the ILP solver can decide which VCs to utilize. To distinguish which VCs were used for payment paths or enforced for bypassing corrupted nodes, we formally 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. The edge id will allow us to distinguish two level-m ($m>0$) VCs vc^k_{ij} between i and j over k built over different VCs, e.g. vc^u_{kj} and vc^v_{kj}. We define $E_{PC} = \{(i, j, \langle i, j\rangle) \mid pc_{ij} \text{ exists}\}$. We then define E_{VC} as the union of all possible VCs for each level 0 to w. We define $E^0_{VC} = \{(i, j, ch_{ik}.id \circ ch_{kj}.id) \mid ch_{ik} = (i, k, id) \wedge ch_{kj} = (k, j, id') \wedge ch_{ik}, ch_{kj} \in E_{PC}\}$, where $\circ$ is a function that joins two ids to a unique new id, e.g. $id \circ id' = \langle id, id'\rangle$. For $1 \leq m \leq w$, $E^m_{VC} = \{(i, j, ch_{ik}.id \circ ch_{kj}.id) \mid ch_{ik} = (i, k, id) \wedge ch_{kj} = (k, j, id') \wedge ch_{ik}, ch_{kj} \in E_{PC} \cup (\cup^{m-1}_{\ell=0} E^\ell_{VC}) \wedge \{ch_{ik}, ch_{kj}\} \cap E^{m-1}_{VC} \neq \emptyset\}$. For example, the edge id can be a breakdown of all edges (channels) building it: the id of a PC between i, j is $\langle i, j\rangle$, the id of a level-0 VC between i, j over k is $\langle\langle i, k\rangle, \langle k, j\rangle\rangle$, and the id of a level-$m$ VC consisting of two channels ch_x and ch_y is $\langle ch_x.id, ch_y.id\rangle$, where $m \leq w$.

The recursion depth w can make E_{VC} and hence the ILP size exponential. For example, for $w = 0$, we need to consider $\mathcal{O}(\binom{n}{2}) = \mathcal{O}(n^2)$ paths of size 2 for constructing all possible level-0 VCs. However, when considering $w = \Theta(n)$, the size of E_{VC} becomes exponential, as it is proportional to $\mathcal{O}(\sum^n_{k=2}\binom{n}{k}) = \mathcal{O}(2^n)$.

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 virtual channel ($vc^k_{ij}.capacity$), and the third set indicates whether a virtual channel 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 $P \in \mathcal{P}(s_t, d_t)$. This set of variables is exponential on the number of nodes, but our goal here is to design an exact solution to an NP-complete problem, thus this is expected. The exact solution is necessary step before designing fast exact solution implementations or approximations (e.g. ILP relaxations and rounding rules). 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 2 if $ch_{ij} \in P$ and is zero otherwise. Let $routing\_cost_{ch_{ij}} = \sum^T_{t=1}\sum_{P\in\mathcal{P}(s_t,d_t)} routing\_fee(t, P, ch_{ij}) \cdot path_P(trans_t)$ 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 $\tilde{\mathcal{X}}$, i.e., estimated to be corrupted, should not be used for routing payments, thus we exclude those paths from $\cup^T_{t=1}\mathcal{P}(s_t, d_t)$. For instance, for value privacy, we exclude every path P such that $x \in P$, for all $x \in \tilde{\mathcal{X}}$. Thus, all remaining input paths can be safely selected by the ILP solver.

We denote the capacity of a virtual channel $vc^k_{ij} \in E_{VC}$ with $vc^k_{ij}.capacity$. We define the virtual channel creation cost as $vc^k_{ij}\_creation\_cost = exists\_vc^k_{ij} \cdot vc^k_{ij}.base\_fee + vc^k_{ij}.prop\_fee \cdot vc^k_{ij}.capacity$, where $exists\_vc^k_{ij}$ is a binary variable indicating if vc^k_{ij} exists. If vc^k_{ij} is used for routing transactions ($exists\_vc^k_{ij} = 1 \wedge vc^k_{ij}.capacity > 0$), then the creation cost is $vc^k_{ij}.base\_fee + vc^k_{ij}.prop\_fee \cdot vc^k_{ij}.capacity$. 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 virtual channel 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:

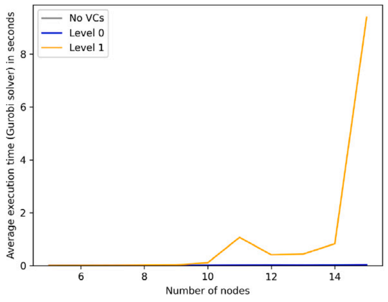
(C1) At most one path can be used for routing a transaction: $\sum_{P\in\mathcal{P}(s_t,d_t)} path_P(trans_t) \leq 1$,
$\forall t \in [1, T]$.

(C2) The percentage of successful transactions or transacted volume should be at least $c_{tr} \in [0, 1]$. To define this constraint we first define the following sum $\sum_{P\in\mathcal{P}(s_t,d_t)} path_P(trans_t)$ that is 1 when a transaction is successful (only one path routes the transaction) and 0 otherwise (no path is selected). Note that this sum is binary due to C1. We then express the constraint as follows: $\sum^T_{t=1} trans_t \cdot \sum_{P\in\mathcal{P}(s_t,d_t)} path_P(trans_t) \geq c_{tr}\sum^T_{t=1} trans_t$, where c_{tr} is the success volume ratio. In case c_{tr} is the minimum percentage of successful transactions, this constraint becomes the following: $\sum^T_{t=1}\sum_{P\in\mathcal{P}(s_t,d_t)} path_P(trans_t) \geq c_{tr}T$.

(C3) Virtual and payment channel capacities should be respected. The load on each channel is the sum of routing costs, transaction amounts, as well as the VC creation costs and capacities for higher order VCs in E_{VC} that use the channel (we denote those with $vc^j_{ix} = (ch_{ij}, \bullet)$, where $\bullet$ is any channel between j and x in E that can form vc^j_{ix}). Thus for every $ch_{ij} \in E = E_{PC} \cup E_{VC}$ we require that $routing\_cost_{ch_{ij}} + trans\_amount(ch_{ij}) + \sum_{vc^j_{ix}\in E_{VC}: vc^j_{ix}=(ch_{ij},\bullet)}(vc^j_{ix}.creation\_cost + vc^j_{ix}.capacity) \leq ch_{ij}.capacity$ where $trans\_amount(ch_{ij}) = \sum^T_{t=1} trans_t \cdot used(ch_{ij}, t)$. If possible by the capacities, C2 forces the ILP solver to set some $path_P(trans_t)$ variables to 1, i.e., some paths are selected for routing transactions and thus the 0 solution (no successful transaction) is prohibited for the inputs where a feasible solution exists. Moreover, if a VC is in a path selected by the ILP solution for routing a payment, then the routing cost and the transaction amount that are charged to this VC's capacity are positive, and thus the VC capacity is positive and lower bounded by this amount. Since VC capacities are part of the objective function, any minimal solution will assign the minimal VC capacity for routing transactions or creating higher order VCs. Similarly, VC capacity will be 0 for VCs that are not used.

(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: $exists\_vc^k_{ij} \leq \sum^T_{t=1} used(vc^k_{ij}, t) + \sum_{vc^\ell_{sr}\in rec(vc^k_{ij})} exists\_vc^\ell_{sr}$ and $exists\_vc^k_{ij} \geq \frac{1}{T+|E_{VC}|}\left(\sum^T_{t=1} used(vc^k_{ij}, t) + \right.$



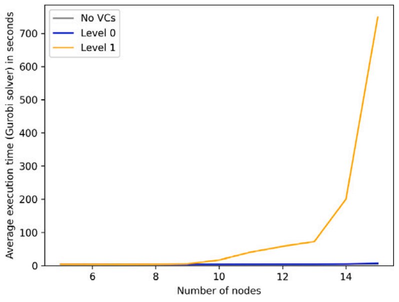


Fig. 3. Average execution time to number of nodes, with 5 transactions. On the left we show the execution time for the Gurobi Solver and on the right the execution time for building the ILP model. Both show exponential growth and the solver started to crash for some graphs with 15 nodes.

$\sum_{vc^{\ell}_{sr}\in rec(vc^k_{ij})} exists\_vc^{\ell}_{sr}\Big)$, 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 enforces $exists\_vc^k_{ij} = 1$ if the VC is used. Note that, $exists\_vc^k_{ij}$ appears in $vc^k_{ij}.creation\_cost$ as a factor of $vc^k_{ij}.base\_fee$, making the VC creation cost calculation accurate. We give an example run of the ILP in Section 4.1.

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

Computational complexity. 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 using Python and Gurobi [28]. We were able to run it with at most 15 nodes, 30 channels, and 5 transactions (cf. Section 4.2 and Fig. 3). One way to make the ILP solution computation more tractable is to restrict $w = \mathcal{O}(1)$ (VC recursion bound) and to limit the number of possible payment paths per transaction to be polynomially many. While this restriction can allow us to run the exact algorithm for larger networks, it limits the creation of VCs for distant nodes.

4.1. ILP example

We apply the ILP to the example graph of Fig. 4 with the same input. In this example we also assume that the capacities are enough for routing all transactions and opening the VCs shown in the figure (VC_1, VC_2, VC_3), it is possible to build level-0 and level-1 VCs, the minimum success volume ratio is 1, and that $\tilde{\mathcal{X}} = \{H_1\}$. We first remove from the ILP input all paths containing H_1. The ILP will output VC_1, VC_2, VC_3 as the paths used for A-to-B, B-to-C, and A-to-C transactions, respectively. These payment paths are the cheapest ones for the corresponding transactions and it is possible to build them given the underlying PC capacities. Note that any other path would be longer and more costly due to linear routing fees.

We now check how all constraints are respected. According to C1, only the payment paths VC_1, VC_2, VC_3 will be used ($path_P(trans_t) = 1$ only for those paths and 0 for all others). C2 will be satisfied since all transactions are successful. C3 is satisfied because we assumed there are enough capacities to build all three VCs and route all transactions. In fact, since the VC capacities are minimized in the objective function, they will be just enough to route the input transactions. C4 will force VC_1, VC_2, VC_3 exist in both directions. The first inequality of C5 will force all VCs in $E_{VC} \setminus \{VC_1, VC_2, VC_3\}$ to not exist since the right side of the inequality will be 0, while the second inequality will force VC_1, VC_2, VC_3 to exist since the right side of the inequality will be a positive value in $(0,1]$.

4.2. ILP experiments

We implemented the ILP with Gurobi [28] and experimented with the size of the graphs it can solve. The github repository of our implementation and experimental evaluation can be found in [29]. We experimented both with random graphs and with a custom heuristic that produces random graphs that resemble the Lightning Network, i.e. graphs with dense core and sparse boundary. To compute the latter, we extracted the probability distribution of the percentage of nodes to which a node is connected to, i.e. d_i/N, where d_i is the degree of node i and N is the number of nodes in the Lightning Network snapshot we used. Then, given the size of the new (smaller) graph, say k, we sampled k values uniformly at random from that distribution. We fixed the sample graph edges to nodes ratio to 2:1, and when needed we added some more edges randomly from the leaf nodes to maintain the desired characteristics of the LN. Both for random graphs and for the graphs computed with our heuristic, we computed each case by running each experiments 25 times and taking the average value. We show our findings in Fig. 3.

As expected due to the NP-hardness of the studied problem, the execution time grows exponentially with the number of nodes. The maximum number of nodes for which the execution was terminating was 15. While this small number of nodes is expected and far smaller than the current size of the Lightning Network, the exact algorithm can still be useful for small sub-networks. One such example can be a sub-network of Lightning Network's highly connected (hub) nodes. Those nodes are vital in transaction routing within a PCN [30] and across PCNs [31].

5. 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 6). Before investigating our overarching optimization goal of Section 3, 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)–(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)–(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

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

While we will present concrete pseudocode for implementing the function *compute_nodes_to_bypass* to achieve each optimization goal (i)–(iv) in Algorithms 5–8 in Section 5.3, 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 5.4, 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, as we show now.

5.1. Runtime analysis of greedy algorithm

Our greedy algorithm is efficient enough, such that we can run the experiments we conduct in Section 6 on a Lightning Network snapshot on commodity hardware. Dijkstra's algorithm has a runtime of $\Theta(|E| + |V|\log|V|)$. We need to run Dijkstra's algorithm to find the shortest path for each payment because the PCN topology and the channel capacity change after each payment and VC construction, and the payment amount is different. Additionally, for each payment, we need to traverse each edge to (temporarily) remove edges that do not have enough capacity to route each payment. Finally, identifying which nodes to bypass, creating the VC, and routing the payment are all linear in the number of nodes on the payment path. Thus, if T is the number of all payments and D the network diameter when considering only PCs (and a bound to the maximum length of a payment path), the total complexity of Algorithm 1 is $\Theta(T \cdot (|E| + |V|\log|V| + D))$.

5.2. 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 4), and the greedy algorithm on a small example graph. The graph is shown in Fig. 4 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 [32].

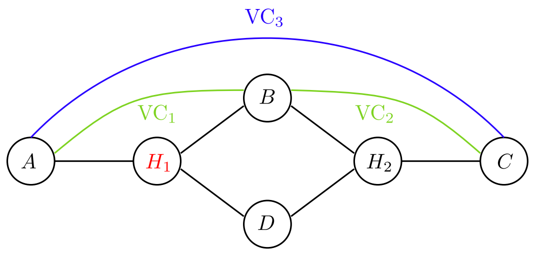


Fig. 4. Results of the ILP and greedy algorithms run on a small sample graph.

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 Fig. 4, 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.

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.

5.3. Algorithms

In this section, we give pseudocode definitions for the algorithms used to conduct our empirical evaluation. Algorithm 2 describes the algorithm for computing the cost ratio for preventing attacks. Algorithm 4 describes the algorithm for computing the cost ratio for the goal of optimizing fees. Algorithm 3 is the subprocedure to compute the corrupted nodes.

Further, we present the algorithms that return the sets of nodes that need to be bypassed to prevent value privacy attacks (Algorithm 5), relationship anonymity attacks (Algorithm 6), wormhole attacks (Algorithm 7) and to optimize fees (Algorithm 8).

5.4. Synergy among objectives

From the definitions of the three different attacks and the strategies of how to prevent them, 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. Furthermore, following our fee optimization algorithm also prevents all three security and privacy attacks (see Section 6).

On a high level, for a given path p, let $V(p), R(p), W(p), F(p)$ be one out of potentially multiple sets of nodes that are at least required to be bypassed for preventing value privacy attacks, relationship anonymity attacks, and wormhole attacks, as well as for optimizing the fees, respectively. That is, if these nodes or a superset of them are bypassed,

Algorithm 2 Compute cost ratio of bypassing corrupted nodes

```
1: function COMPUTE_COST_RATIO(pcn, adv_budget, val_range, n, repetition)
2:     pcn_vc ← copy(pcn) //Copy the PCN graph for the VC simulation
3:     c ← compute_corrupted_nodes(pcn, adv_budget) //compute the most profitable corrupted nodes to a given adversary budget
4:     payments ← generate_payments(pcn, val_range, n, repetition) //generates n random payments with a random value in val_range in a PCN graph
5:     establish_cost_vc ← 0
6:     route_cost_vc ← 0
7:     unsuccessful_vc ← 0
8:     route_cost_pcn ← 0
9:     unsuccessful_pcn ← 0
10:    for (path, value, repetition) in payments do
11:        cost, new_path ← bypass_corrupted_nodes(c, path, value, repetition, pcn_vc)
12:        establish_cost_vc ← establish_cost_vc + cost
13:        for $i$ ← 0; $i$ < repetition ; $i++$ do
14:            success, fees_vc ← conduct_payment(pcn_vc, new_path, value) //executes the payment over the given path
15:            if success then
16:                route_cost_vc ← route_cost_vc + fees_vc
17:            else
18:                unsuccessful_pcn ← unsuccessful_pcn + 1
19:            success, fees_pcn ← conduct_payment(pcn, path, value) //executes the payment over the given path
20:            if success then
21:                route_cost_pcn ← route_cost_pcn + fees_pcn
22:            else
23:                unsuccessful_pcn ← unsuccessful_pcn + 1
24:    return $\frac{\text{establish\_cost\_vc + route\_cost\_vc}}{\text{route\_cost\_pcn}}$
```

Algorithm 3 Compute corrupted nodes

```
1: function COMPUTE_CORRUPTED_NODES(pcn, adv_budget)
2:     payments ← generate_payments(pcn, val_range, n, repetition) //generates n random payments
3:     node_map<node,int> ← {}
4:     for (path, value, repetition) in payments do
5:         for node in path do
6:             node_map[node] ← node_map[node]+1
7:     cost_benefit_map<node,int> ← {}
8:     for node in node_map.keys() do
9:         cost_benefit_map[node] ← $\frac{\text{(node\_map[node]/n)}}{\text{(cost(node)/adv\_budget)}}$
10:    corrupted_nodes ← []
11:    for (node,cost) in cost_benefit_map.sort_by_value(desc) do
12:        if adv_budget ≥ cost(node) then
13:            corrupted_nodes ← corrupted_nodes.append(node)
14:            adv_budget ← adv_budget - cost(node)
15:    return corrupted_nodes
```

the corresponding attack is prevented. We define these sets formally as the return values of Algorithm in Section 5.3. Since $V(p), F(p)$ contain (at least) all adversarial nodes, optimizing for these objectives also prevents any other attack relying on on-path adversaries, such as denial-of-service attacks [10].

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

Algorithm 4 Compute cost ratio for saving fees

```
1: function COMPUTE_COST_RATIO(pcn, val_range, n, repetition)
2:     pcn_vc ← copy(pcn) //Copy the PCN graph for the VC simulation
3:     payments ← generate_payments(pcn, val_range, n, repetition) //generates n random payments with a random value in val_range in a PCN graph
4:     establish_cost_vc ← 0
5:     route_cost_vc ← 0
6:     unsuccessful_vc ← 0
7:     route_cost_pcn ← 0
8:     unsuccessful_pcn ← 0
9:     for (path, value, repetition) in payments do
10:        cost, new_path ← open_profitable_vcs(c, path, value, repetition, pcn_vc)
11:        establish_cost_vc ← establish_cost_vc + cost
12:        for $i$ ← 0; $i$ < repetition ; $i++$ do
13:            success, fees_vc ← conduct_payment(pcn_vc, new_path, value) //executes the payment over the given path
14:            if success then
15:                route_cost_vc ← route_cost_vc + fees_vc
16:            else
17:                unsuccessful_pcn ← unsuccessful_pcn + 1
18:            success, fees_pcn ← conduct_payment(pcn, path, value) //executes the payment over the given path
19:            if success then
20:                route_cost_pcn ← route_cost_pcn + fees_pcn
21:            else
22:                unsuccessful_pcn ← unsuccessful_pcn + 1
23:    return $\frac{\text{establish\_cost\_vc + route\_cost\_vc}}{\text{route\_cost\_pcn}}$
```

Algorithm 5 Generating set of nodes to bypass for preventing VP

```
1: function COMPUTE_BYPASS_NODES_VP(path, corrupted_nodes)
2:     nodes_vp_path ← {}
3:     for node in path do
4:         if node in corrupted_nodes then
5:             nodes_vp_path ← nodes_vp_path ∪ {node}
6:     return nodes_vp_path
```

Algorithm 6 Generating set of nodes to bypass for preventing RA

```
1: function COMPUTE_BYPASS_NODES_RA(path, corrupted_nodes)
2:     Let nodes_ra_path_l be the set of connected nodes in corrupted_nodes adjacent to the sender path[0]
3:     Let nodes_ra_path_r be the set of connected nodes in corrupted_nodes adjacent to the receiver path[length(path)-1]
4:     if length(nodes_ra_path_l) < length(nodes_ra_path_r) then
5:         return nodes_ra_path_l
6:     return nodes_ra_path_r
```

Proof. From the definitions of Algorithms 5–8, we observe the following. $V(p)$ is the set containing all corrupted nodes on the path p. Intuitively, if it is not, then there is a corrupted node left on the path which is not bypassed, thus value privacy attacks are not 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 $V(p) \supseteq W(p)$.

Algorithm 7 Generating set of nodes to bypass for preventing WH

```
1:  function COMPUTE_BYPASS_NODES_WH(path, corrupted_nodes)
2:      nodes_wh_path ← {}
3:      for while there exist honest nodes in path that is surrounded by corrupted nodes do
4:          Let hon_nodes be one of these honest nodes surrounded by corrupted nodes
5:          Let wh_l be the set of connected nodes in corrupted_nodes adjacent to the left of hon_nodes
6:          Let wh_r be the set of connected nodes in corrupted_nodes adjacent to the right of hon_nodes
7:          if length(wh_l) < length(wh_r) then
8:              Remove wh_l from path
9:              Add wh_l to nodes_wh_path
10:         else
11:             Remove wh_r from path
12:             Add wh_r to nodes_wh_path
                ▷ Note that one could also bypass hon_nodes, but we do not for simplicity here
13:     return nodes_wh_path
```

Algorithm 8 Generating set of nodes to bypass for optimizing fees

```
1:  function OPTIMIZE_FEES(path, corrupted_nodes)
2:      nodes_fees_path ← path
3:      Remove first and last element from nodes_fees_path
4:      return nodes_fees_path
```

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 $V(p)$. □

6. 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 5).

6.1. Input data preparation and methodology

Graph model and data. Recalling our model in Section 3, 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 [33]. 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 3). Our solution is modular, other fee models can be used, and we discuss other possible fee models in Section 7. If a VC is established, its routing fee is set to the fee of the initiating endpoint's underlying channel.

Countermeasures. Honest nodes assume there is an attacker with an estimated budget who corrupts nodes according to an estimated strategy. In particular, as discussed in Section 3, we model an adaptive adversary who selects nodes with the best cost–benefit ratio after observing the honest nodes' countermeasures. The honest nodes preemptively identify a set $\tilde{\mathcal{X}}$ of high-value target nodes, representing those that would be most profitable for the adversary to corrupt in the absence of countermeasures. We formally define how $\tilde{\mathcal{X}}$ is computed. 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., $capacity\_locked(v) := \sum_{w \in \mathcal{V} \setminus \{v\}} (pc_{\langle v,w \rangle} \cdot \beta_1)$.

The assumption that an attacker corrupts nodes that are most beneficial to it, while being cheap to place, is based on previous works [22–25]. 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 $\tilde{\mathcal{X}}$ iterating over l and adding those for which the following condition holds after adding them: $\sum_{n \in \tilde{\mathcal{X}}} (capacity\_locked(n)) \leq adv\_budget$

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.

6.2. 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 5. 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 Algorithms 5–8 in Section 5.3.

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

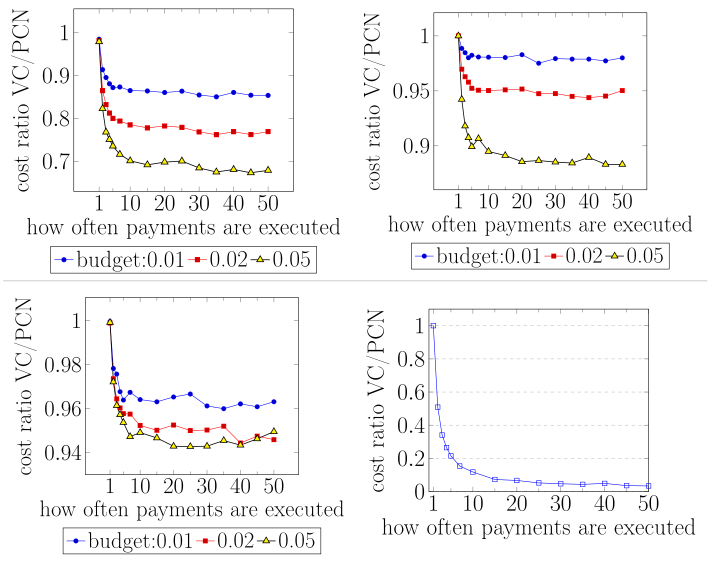


Fig. 5. Optimizing for value privacy (top left), relationship anonymity (top right), wormhole attack (bottom left), and fees (bottom right)

			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

Fig. 6. How many paths are prone to different attacks when optimizing for different goals for an adversary budget of 0.05.

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 need to compute the percentage of paths that are prone to which attack for different adversary budgets. We expand on this in Section 6.3 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 6.1. 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 Fig. 5. 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 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 Fig. 6 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 5.4.

6.3. Computing the percentage of prone paths for different adversary budgets

We choose different adversary budgets ranging from 0.01% to 80% of the PCN capacity and check, how many of our payments are prone to the different attacks. We plot our results in Fig. 7. We observe that value privacy attack is the cheapest, only 0.1% yields more than half of the payments being prone. Note that an adversary budget of 0.1% of the total capacity is still significant and requires in our data the staking of roughly 1.2 BTC. Achieving the wormhole attack is more expensive and the effectiveness peaks at around 5% (58.4 BTC in our data) as adversary budget, before going down again. Recall that in a wormhole attack, the adversary surrounds honest intermediaries on a payment path with two corrupted nodes and then steals their fees. The effectiveness of this attack decreases when the adversary's budget is too

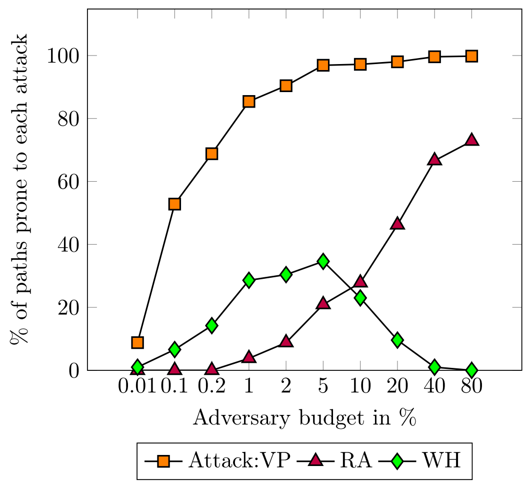


Fig. 7. Number of prone paths.

high because the attack requires honest nodes between corrupted nodes to extract fees. If too many nodes are corrupted, there are fewer honest intermediaries left on payment paths, limiting the adversary's ability to steal fees. The most expensive attack to mount for our corruption strategy is relationship anonymity. Only after 10% adversary budget it affects more paths than wormhole attack.

Note that an adversary can use its corrupted nodes to carry out any or all of these attacks. Note that there might exist other strategies for corrupting nodes that are more effective for relationship anonymity and wormhole attacks (cf. [22]). From these results we choose 1, 2 and 5 percent as the adversary budget we want to investigate further, as with these budgets there are some (and not all) paths that are prone to each of the attacks and to be able to compare them more easily.

7. Discussion

Fair establishment fee model for intermediaries. VCs are not used yet in practice, and thus, we do not know how fees are going to be charged. In Section 6, we assumed that the fee of opening a VC with some capacity α is the same as routing the payment of the same value through this path. Of course, other establishment fee models are possible.

First, in order to be fair to both the endpoints and intermediaries, VCs should have a limited lifespan. Remember that VCs require intermediaries to lock up some funds that they cannot use for other payments for which they would otherwise be able to charge fees. Note that this is different to PCs, where it is not problematic for two users to lock up their funds potentially indefinitely, as they do it only for their own funds. In other words, the fees that the intermediaries receive have to be proportionate to the time for which the money is locked up, i.e., the lifespan of a VC. Since channel capacities are finite, this implies that the lifespan is also finite.

The way we previously modeled fees is that VC creation fees are independent of the lifespan of a channel. In fact, one could argue that the fees for routing payments should then also be dependent on the collateral timeout, which, in practice, it is not. However, it is hard to model this in our experiments, as due to the nature of the PCN, it is unknown how many payments are processed in which time frames. For this reason, we chose our simplified fee model.

In order for VCs to still be profitable in this fairer fee model, one would have to find a number k of payment repetitions (for each of our n payments), a lifetime of the VC t, the base fee f, the fee p proportional to the amount and a fee v that is proportional to the lifetime and the capacity of the VC, such that the inequality $\sum_n(\sum_k(f + \alpha \cdot p)) \geq \sum_n(f + k \cdot \alpha \cdot p + k \cdot \alpha \cdot v \cdot t)$ holds.

This simplifies to the following: $(k-1) \cdot f \geq k \cdot \alpha \cdot v \cdot t$. We note that this inequality is not exact, as the forwarded amount gets smaller when intermediaries already deduct fees. Nonetheless, the investigation of this or other fee models is interesting future work.

Utilizing bidirectionality of VCs. The greedy algorithms create virtual channels with transactions in one direction in mind. This means that on a path where several transactions are executed, the capacity is chosen as the sum of the amount of these transactions. In reality, the capacity can be lower if there are transactions in the other direction in the same time frame. Since we do not model any timing as mentioned above, we also do not capture this.

A practical scenario where this is useful might be where two payment providers route a substantial amount of payments through a hub back and forth. Note that the sum of all their payments can be very large, but since they send it back and forth the capacity of the virtual channel can be smaller.

Adversaries agreeing to open VCs. One could argue that an adversary trying to conduct an on-path attack would not agree to create a VC because the adversary would hinder its own attack capabilities. However, refusal to participate in creating a VC will be noticed by the sender. Thus, a sender concerned with security and privacy can always prevent such an attack by finding an alternative path.

Blocking capacity with VC. If a VC is used only sparsely, then it might block the capacity of the underlying payment channels. Thus, it should either be used for routing many payments or be open only for a short amount of time.

VC routing fee. Similar to the establishment fee, it is unclear what fee the users of a virtual channel will charge in practice for letting other users route their payments through the virtual channel. In this work, we assume this fee to be the same as fees that are charged for one of the underlying channels.

Privacy of VC creation. Following our strategies, we aim to prevent the on-chain privacy attacks on value privacy and relationship anonymity for payments. However, since opening or closing a virtual channel is an operation in which potentially corrupted intermediaries participate, some information might be leaked while opening or closing a virtual channel.

We assume that VCs are used in the same way as payment channels, e.g., for routing payments through them. To make other users aware of accepting payment being routed through channels, channels need to be announced publicly. Therefore, we disregard privacy concerns for VC opening/closing and instead assume that endpoints and capacity of VCs are announced publicly, thereby already leaking the information on endpoints and capacity. However, it is important to note that the leakage from VC setup is significantly less fine-grained than the direct exposure of payment details: while an adversary may learn that a VC has been established, it does not reveal when or how frequently payments occur within it. Future research could explore techniques to further mitigate privacy leakage during VC setup while maintaining routing functionality.

Using VCs for routing. In our evaluation of the greedy algorithm, we consider the payments that are repeated once or multiple times and conclude that if payments are conducted more than once along a path, opening a direct VC is the best strategy. However, for payments that are conducted only once, the VCs that were created in this fashion can be used for having shorter paths, which means having fewer fees and being less at risk for attacks due to having fewer intermediaries.

Adversarial budget and repeated VC-opening. Our model assumes an adversary with a fixed budget that is fully deployed at once. However, a more sophisticated analysis could consider a multi-round setting, where the adversary distributes its budget dynamically as honest nodes open new virtual channels (VCs). A promising future direction is to formalize this as a VC-opening game in a repeated, round-based setting, where both honest nodes and the adversary iteratively adjust their actions. While our work provides a step towards understanding these interactions, developing a model for a multi-round VC-opening setting, with adaptive strategies, and the adversary optimally distributing its budget over time remains an open challenge for future research.

Real-world adversaries and deployment challenges. Real-world deployment introduces challenges such as network congestion, fluctuating liquidity, and limited user adoption. These can be modeled by non-bypassable nodes (low adoption) and edges with payment limits (congestion, fluctuating liquidity). Additionally, real-world adversaries may exhibit long-term strategic behavior, try to manipulate routing, perform probing [34] or other side-channel attacks not captured in our model. Exploring VC-opening strategies under such constraints and adversarial models remains an important direction for future research.

8. Related work

Over the last years, significant research efforts have been devoted to the design and analysis of efficient and secure payment channel networks [35–37]. Motivated by topology-based attacks [38], the possibility of route hijacking [10] as well as vulnerabilities, e.g., related to the privacy [18,19,23] and anonymity of PCN users [39,40], to just name a few examples, much existing literature revolves around network connectivity [38], the payment routing system [5,10], as well as privacy aspects, e.g., of route discovery [18,30].

To ensure anonymity, payment-channel networks usually rely on privacy-enhancing cryptographic schemes (e.g., onion routing) to implement the 2-phase commit payment operation. PrivPay [41], SilentWhispers [42], Fulgor/Rayo [5], AMHL [3] provide privacy-preserving multi-hop payment protocols which come with formal guarantees. SpeedyMurmurs [43] formalizes and addresses concrete notions of privacy in the context of payment routing. SpiderNetwork [44] improves the effectiveness of source routing in a dynamic PCN by favoring routes that minimize the balance difference using on-chain rebalancing. A privacy-preserving approach to discovering low-cost routes was recently presented by Pietrzak et al. [30]. Blitz [4] is a 1-phase payment scheme, which, similar to AMHL, provides security against wormhole attacks [3]. None of the payment-based approaches, however, hide the value of the payment to intermediaries or decrease routing fees.

Therefore, an intriguing approach to improving the security and efficiency of payment channel networks is the use of virtual channels. These have been introduced by Dziembowski et al. [11] to overcome the requirement that intermediaries along a channel route need to be online (a concern also considered in [45,46]) 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 [12–14].

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 [47], Khamis and Rottenstreich studied how to amortize the creation of new channels through reduced routing costs, however, without accounting for security aspects.

9. 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, and more dynamic adversarial strategies. In particular, further exploration is needed on how an adversary can optimally adapt to the countermeasures proposed in this work and how honest nodes can refine their defenses accordingly.

CRediT authorship contribution statement

Lukas Aumayr: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Esra Ceylan:** Investigation. **Yannik Kopyciok:** Software, Investigation. **Matteo Maffei:** Writing – review & editing, Supervision. **Pedro Moreno-Sanchez:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Investigation, Conceptualization. **Iosif Salem:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Stefan Schmid:** Writing – review & editing, Writing – original draft, Visualization, Supervision, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been supported by the European Research Council (ERC) under 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 Research Foundation (DFG), project ReNO (SPP 2378); by MCIN/AEI/ 10.13039/501100011033 and European Union NextGenerationEU/PRTR via the PRODIGY Project (TED2021-132464B-I00), grants PID2022-142290OB-I00 as well as IJC2020-043391-I.

Data availability

Data will be made available on request.

References

[1] L. Aumayr, E. Ceylan, Y. Kopyciok, M. Maffei, P. Moreno-Sanchez, I. Salem, S. Schmid, Optimizing virtual payment channel establishment in the face of on-path adversaries, in: 2024 IFIP Networking Conference, IFIP Networking, 2024, pp. 1–10, http://dx.doi.org/10.23919/IFIPNetworking62109.2024.10619889.
[2] J. Poon, T. Dryja, The bitcoin lightning network:, 2016, p. 59.
[3] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, M. Maffei, 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.
[4] L. Aumayr, P. Moreno-Sanchez, A. Kate, M. Maffei, Blitz: Secure multi-hop payments without two-phase commits, in: 30th USENIX Security Symposium, USENIX Security 21, USENIX Association, 2021, pp. 4043–4060.
[5] G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, S. Ravi, Concurrency and privacy with payment-channel networks, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 455–471.

[6] E. Heilman, L. Alshenibr, F. Baldimtsi, A. Scafuro, S. Goldberg, Tumblebit: An untrusted bitcoin-compatible anonymous payment hub, in: Network and Distributed System Security Symposium, 2017.

[7] E. Tairi, P. Moreno-Sanchez, M. Maffei, A\(\^mbox2\)L: anonymous atomic locks for scalability and interoperability in payment channel hubs, 2019, p. 589, IACR Cryptol. ePrint Arch. 2019.

[8] C. Egger, P. Moreno-Sanchez, M. Maffei, Atomic Multi-Channel Updates with Constant Collateral in Bitcoin-Compatible Payment-Channel Networks, Tech. Rep. 583, 2019.

[9] M. Jourenko, M. Larangeira, K. Tanaka, Payment Trees: Low Collateral Payments for Payment Channel Networks, Tech. Rep., 2021, Financial Cryptography and Data Security.

[10] S. Tochner, A. Zohar, S. Schmid, Route hijacking and DoS in off-chain networks, in: Proc. ACM Conference on Advances in Financial Technologies, AFT, 2020.

[11] S. Dziembowski, L. Eckey, S. Faust, D. Malinowski, Perun: Virtual payment hubs over cryptocurrencies, in: 2019 IEEE Symposium on Security and Privacy, SP, 2019, pp. 106–123, http://dx.doi.org/10.1109/SP.2019.00020.

[12] L. Aumayr, M. Maffei, O. Ersoy, A. Erwig, S. Faust, S. Riahi, K. Hostáková, P. Moreno-Sanchez, Bitcoin-compatible virtual channels, in: 2021 IEEE Symposium on Security and Privacy, SP, IEEE Computer Society, 2021, pp. 901–918, http://dx.doi.org/10.1109/SP40001.2021.00097.

[13] M. Jourenko, M. Larangeira, K. Tanaka, Lightweight virtual payment channels, in: S. Krenn, H. Shulman, S. Vaudenay (Eds.), Cryptology and Network Security - 19th International Conference, CANS 2020, Proceedings, Springer, 2020, pp. 365–384, http://dx.doi.org/10.1007/978-3-030-65411-5_18.

[14] L. Aumayr, P. Moreno-Sanchez, A. Kate, M. Maffei, Breaking and fixing virtual channels: Domino attack and donner, in: Network and Distributed System Security (NDSS) Symposium, 2023.

[15] G. Avarikioti, R. Scheuner, R. Wattenhofer, Payment networks as creation games, in: Data Privacy Management, Cryptocurrencies and Blockchain Technology - ESORICS 2019 International Workshops, DPM 2019 and CBT 2019, Luxembourg, September 26-27, 2019, Proceedings, in: Lecture Notes in Computer Science, vol. 11737, Springer, 2019, pp. 195–210, http://dx.doi.org/10.1007/978-3-030-31500-9_12.

[16] Z. Avarikioti, L. Heimbach, Y. Wang, R. Wattenhofer, Ride the lightning: The game theory of payment channels, in: J. Bonneau, N. Heninger (Eds.), Financial Cryptography and Data Security - 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10-14, 2020 Revised Selected Papers, in: Lecture Notes in Computer Science, vol. 12059, Springer, 2020, pp. 264–283, http://dx.doi.org/10.1007/978-3-030-51280-4_15.

[17] O. Ersoy, S. Roos, Z. Erkin, How to profit from payments channels, in: J. Bonneau, N. Heninger (Eds.), Financial Cryptography and Data Security - 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10-14, 2020 Revised Selected Papers, in: Lecture Notes in Computer Science, vol. 12059, Springer, 2020, pp. 284–303, http://dx.doi.org/10.1007/978-3-030-51280-4_16.

[18] W. Tang, W. Wang, G. Fanti, S. Oh, Privacy-utility tradeoffs in routing cryptocurrency over payment channel networks, Proc. the ACM Meas. Anal. Comput. Syst. 4 (2) (2020) 1–39.

[19] U. Nisslmueller, K.-T. Foerster, S. Schmid, C. Decker, Toward active and passive confidentiality attacks on cryptocurrency off-chain networks, in: Proc. 6th International Conference on Information Systems Security and Privacy, ICISSP, 2020.

[20] S. Tripathy, S.K. Mohanty, Mappcn: Multi-hop anonymous and privacy-preserving payment channel network, in: International Conference on Financial Cryptography and Data Security, Springer, 2020, pp. 481–495.

[21] EmelyanenkoK, Payment channel congestion via spam-attack, 2020, https://github.com/lightningnetwork/lightning-rfc/issues/182.

[22] S. Tikhomirov, P. Moreno-Sanchez, M. Maffei, A quantitative analysis of security, anonymity and scalability for the lightning network, in: IEEE European Symposium on Security and Privacy Workshops, EuroS&P Workshops 2020, Genoa, Italy, September 7-11, 2020, IEEE, 2020, pp. 387–396, http://dx.doi.org/10.1109/EuroSPW51379.2020.00059.

[23] G. Kappos, H. Yousaf, A. Piotrowska, S. Kanjalkar, S. Delgado-Segura, A. Miller, S. Meiklejohn, An empirical analysis of privacy in the lightning network, in: Financial Cryptography and Data Security, 2021.

[24] R. Pickhardt, S. Tikhomirov, A. Biryukov, M. Nowostawski, Security and privacy of lightning network payments with uncertain channel balances, 2021, CoRR. arXiv:2103.08576.

[25] S.P. Kumble, D. Epema, S. Roos, How lightning's routing diminishes its anonymity, in: The 16th International Conference on Availability, Reliability and Security, in: ARES 2021, Association for Computing Machinery, New York, NY, USA, 2021, http://dx.doi.org/10.1145/3465481.3465761.

[26] M. Green, I. Miers, Bolt: anonymous payment channels for decentralized currencies, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, in: CCS '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 473–489, http://dx.doi.org/10.1145/3133956.3134093.

[27] T. Eilam-Tzoreff, The disjoint shortest paths problem, Discrete Appl. Math. 85 (2) (1998) 113–138.

[28] Gurobi Optimization, LLC, Gurobi optimizer reference manual, 2023, URL https://www.gurobi.com.

[29] Y. Kopyciok, vc-optimizer, 2023, https://github.com/ykpyck/vc-optimizer.

[30] K. Pietrzak, I. Salem, S. Schmid, M. Yeo, LightPIR: Privacy-preserving route discovery for payment channel networks, in: Proc. IFIP Networking, 2021.

[31] L. Aumayr, Z. Avarikioti, I. Salem, S. Schmid, M. Yeo, X-Transfer: Enabling and optimizing cross-PCN transactions, 2025, Cryptology ePrint Archive, Paper 2025/272. URL https://eprint.iacr.org/2025/272.

[32] P. Zabka, K.-T. Foerster, S. Schmid, C. Decker, A centrality analysis of the lightning network, 2022, arXiv:2201.07746.

[33] Fiatjaf, lnchannels, 2021, https://web.archive.org/web/20210612140032/https://ln.fiatjaf.com/.

[34] A. Biryukov, G. Naumenko, S. Tikhomirov, 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, Springer-Verlag, Berlin, Heidelberg, 2022, pp. 337–357, http://dx.doi.org/10.1007/978-3-031-18283-9_16.

[35] T. Neudecker, H. Hartenstein, Network layer aspects of permissionless blockchains, IEEE Commun. Surv. Tutor. 21 (1) (2018) 838–857.

[36] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, A. Gervais, Sok: Layer-two blockchain protocols, in: International Conference on Financial Cryptography and Data Security, 2020, pp. 201–226.

[37] M. Dotan, Y.-A. Pignolet, S. Schmid, S. Tochner, A. Zohar., Survey on blockchain networking: Context, state-of-the-art, challenges, in: Proc. ACM Computing Surveys, CSUR, 2021.

[38] E. Rohrer, J. Malliaris, F. Tschorsch, Discharged payment channels: Quantifying the lightning network's resilience to topology-based attacks, in: 2019 IEEE European Symposium on Security and Privacy Workshops, EuroS&PW, IEEE, 2019, pp. 347–356.

[39] E. Rohrer, F. Tschorsch, Counting down thunder: Timing attacks on privacy in payment channel networks, in: AFT '20: 2nd ACM Conference on Advances in Financial Technologies, New York, NY, USA, October 21-23, 2020, ACM, 2020, pp. 214–227, http://dx.doi.org/10.1145/3419614.3423262.

[40] M. Romiti, F. Victor, P. Moreno-Sanchez, P.S. Nordholt, B. Haslhofer, M. Maffei, Cross-layer deanonymization methods in the lightning protocol, 2021, arXiv:2007.00764 [Cs].

[41] P. Moreno-Sanchez, A. Kate, M. Maffei, K. Pecina, Privacy preserving payments in credit networks: enabling trust with privacy in online marketplaces, in: 22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015, The Internet Society, 2015.

[42] G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, SilentWhispers: Enforcing security and privacy in credit networks, in: Network and Distributed System Security Symposium, 2017.

[43] S. Roos, P. Moreno-Sanchez, A. Kate, I. Goldberg, Settling payments fast and private: Efficient decentralized routing for path-based transactions, in: 25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018, The Internet Society, 2018.

[44] V. Sivaraman, S.B. Venkatakrishnan, M. Alizadeh, G.C. Fanti, P. Viswanath, Routing cryptocurrency with the spider network, in: Proceedings of the 17th ACM Workshop on Hot Topics in Networks, HotNets 2018, Redmond, WA, USA, November 15-16, 2018, ACM, 2018, pp. 29–35, http://dx.doi.org/10.1145/3286062.3286067.

[45] P. McCorry, S. Bakshi, I. Bentov, S. Meiklejohn, A. Miller, Pisa: Arbitration outsourcing for state channels, in: ACM Conference on Advances in Financial Technologies, AFT, 2019, pp. 16–30.

[46] G. Avarikioti, E.K. Kogias, R. Wattenhofer, Brick: Asynchronous state channels, 2019, arXiv preprint arXiv:1905.11360.

[47] J. Khamis, O. Rottenstreich, Demand-aware channel topologies for off-chain payments, in: 2021 International Conference on Communication Systems & Networks, COMSNETS, IEEE, 2021, pp. 272–280.