LightPIR: Privacy-Preserving Route Discovery for Payment Channel Networks

Krzysztof Pietrzak* Iosif Salem† Stefan Schmid‡ Michelle Yeo§

*§IST Austria †‡Faculty of Computer Science, University of Vienna, Austria

{pietrzak, michelle.yeo}@ist.ac.at {iosif.salem, stefan_schmid}@univie.ac.at

***Abstract*—Payment channel networks are a promising approach to improve the scalability of cryptocurrencies: they allow to perform transactions in a peer-to-peer fashion, along multi-hop routes in the network, without requiring consensus on the blockchain. However, during the discovery of cost-efficient routes for the transaction, critical information may be revealed about the transacting entities.**

This paper initiates the study of privacy-preserving route discovery mechanisms for payment channel networks. In particular, we present *LightPIR*, an approach which allows a client to learn the shortest (or cheapest in terms of fees) path between two nodes without revealing any information about the endpoints of the transaction to the servers. The two main observations which allow for an efficient solution in *LightPIR* are that: (1) surprisingly, hub labelling algorithms – which were developed to preprocess "street network like" graphs so one can later efficiently compute shortest paths – also perform well for the graphs underlying payment channel networks, and that (2) hub labelling algorithms can be conveniently combined with private information retrieval.

***LightPIR* relies on a simple hub labeling heuristic on top of existing hub labeling algorithms which leverages the specific topological features of cryptocurrency networks to further minimize storage and bandwidth overheads. In a case study considering the Lightning network, we show that our approach is an order of magnitude more efficient compared to a privacy-preserving baseline based on using private information retrieval on a database that stores all pairs shortest paths.**

***Index Terms*—Cryptocurrencies, blockchain, offchain networks, PIR, Lightning**

I. Introduction

As the popularity of cryptocurrencies is growing explosively, the inefficiency of the underlying consensus protocol increasingly becomes a bottleneck, preventing fast payments at scale. A promising solution to this scalability challenge are emerging payment channel networks [1] such as the Lightning network [2], which allow to perform transactions off-chain, without requiring consensus on the blockchain. In a nutshell, a payment channel is a cryptocurrency transaction which escrows or dedicates money on the blockchain for exchange with a given user and duration. Users can also interact if they do not share a direct payment channel: they can route transactions through intermediaries. To incentivize nodes (the intermediaries) to contribute to the transaction routing, payment channel networks typically allow intermediaries to charge nodes which route through them a nominal fee. This is also the approach taken in the Lightning network which serves us as a case study in this paper.

Payment channel networks however also introduce new challenges. In particular, since different routes come at different fees, nodes require scalable mechanisms to discover "short" (i.e., low-cost) routes. To ensure scalability and support lightweight nodes (e.g., wallets) in the network, these route discovery mechanisms must be efficient, both in terms of disk space and control traffic. Especially since payment channel networks can be more dynamic compared to classic communication networks (e.g., channels and liquidities can change over time) [1], it is crucial that resource-constrained nodes do not have to store and maintain the entire network topology to be able to compute a route toward the transaction's target. Thus, over the last years, several innovative approaches for a more scalable routing in payment channel networks have been proposed [3]–[8]. For example, Lightning introduced the notion of trampoline nodes [5] to provide route discovery services to light-weight clients. The trampoline nodes are more powerful nodes which know the entire topology (e.g., using gossiping); a wallet then just needs to know how to reach the route server nodes in its neighborhood and can then request the desired route.

Besides scalability, payment channel networks also need to provide a high degree of privacy: it is critical that other nodes in the payment channel networks cannot learn who transacts with whom. In order to provide anonymity in the transaction routing, payment channel networks typically implement onion routing [4], [6], [9]. However, while onion routing provides security for the routing process, it does not provide security for the route *discovery* process. In particular, by requesting shortest paths to certain destinations from the route discovery servers (e.g., the trampoline nodes in Lightning), a node may reveal critical information about its planned financial transactions [10], [11].

Motivated by this shortcoming of state-of-the-art payment channel networks, this paper initiates the study of the design of scalable privacy-preserving route discovery mechanisms for payment channel networks.

The main contribution of this paper is the first scalable and

*Funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (682815-TOCNeT).

†‡Funded by Austrian Science Fund's (FWF) project ADVISE (I 4800-N).

‡Research funded (in part) by the Vienna Science and Technology Fund (WWTF), project ICT19-045, 2020-2024.

ISBN 978-3-903176-39-3© 2021 IFIP

privacy-preserving route discovery mechanism for payment channel networks, *LightPIR*[1]. *LightPIR* provides information theoretic security guarantees, and is evaluated both analytically as well as empirically, using the Lightning network. We show that our approach is efficient and practical, and may hence be useful also in other networks where the security of route discovery is critical. Concretely, we tested our approach with different snapshots of the Lightning network taken over a period of two years and our heuristic produces a small hub database (about 2MB), which at the moment is an improvement by an order of magnitude compared to the size of the complete network. With the growth of the network, this discrepancy will change further in favor of approaches similar to ours. We note that our techniques are generic and can be applied to any payment channel network.

The remainder of this paper is organized as follows. In Section II we formally define the problem we consider. Section III presents several alternative solutions to our problem and highlights the potential drawbacks of each solution. We then proceed to state our main contributions in Section IV which is an efficient solution that combines PIR and hub-labelling to solve our problem; our solution also relies on a heuristic which optimises the hub set sizes, to further improve efficiency. Section V contains an evaluation of our method on various snapshots of the Lightning network over a two year period. In Section VI we review related work, and finally we discuss future research directions in Section VII.

II. Problem Statement

We model a payment network by a public directed graph $G = (V, E)$, where each edge $e \in E$ has some cost $c(e) \geq 0$. The nodes are the clients and an edge $e = (u, v)$ means there's a channel available from u to v with transaction cost $c(e)$. We reserve the letters $n = |V|$ to denote the number of vertices, $m = |E|$ to denote the number of edges, $\delta = |E|/|V|$ to denote the average degree and d for the edge length of the longest shortest path in G (if all edge costs are identical this is just the diameter). We assume the nodes are labelled by λ-bit strings, while $\lambda = \lceil \log n \rceil$ is sufficient, we leave this as a parameter as one might want to use longer labels in practice.

Below we give our definition of a *private route discovery mechanism*. The definition is very similar to the classic definition of private information retrieval (PIR): on the one hand, is more restricted as we only consider the particular problem of retrieving shortest paths; on the other hand it is more general as the client is not just supposed to recover some database entry, but can perform a more general computation on the retrieved answers to compute the path.

A. Private Route Discovery Mechanism (PRDM) Definition

The involved parties are a content provider $\mathcal{CP}$, a set of servers $\mathcal{S}_1, \ldots, \mathcal{S}_k$ and a client $\mathcal{C}$. The content provider $\mathcal{CP}$ gets as input a graph $G = (V, E)$ and processes it into t databases $\mathsf{DB}_1, \ldots, \mathsf{DB}_k$. DB_i is then sent to $\mathcal{S}_i$. The client $\mathcal{C}$ provides two nodes $s, t \in V$ as input and can now interact with the servers. The **correctness property** we require is that $\mathcal{C}$'s final output is a shortest path from s to t in G (assuming $\mathcal{CP}$ and the $\mathcal{S}_i$'s follow the protocol). The **security guarantee** we require is (ℓ, k)-privacy: the view of any of the $\ell > 0$ out of the k servers should not reveal any information about the query s, t. We are interested in finding **efficient** schemes, that is, the scheme has to scale well with the potential growth of the Lightning network.

Let us stress that once we learn a shortest path in a payment channel network like Lightning (where shortest will typically mean the path with lowest fees), it is not guaranteed that we can actually route a payment through that path: an endpoint on a channel on the path might not hold enough balance to forward the transaction [10]. Usually only the existence of a channel, its fees and the total balance are public, while the exact balances on channels are typically private information (if they weren't, one could learn about transfers by just observing how balances at the endpoints change [11]).

III. Possible Approaches and Drawbacks

We start by giving a description of several schemes that fulfill both the correctness property and the security guarantee as described above but have varying degrees of efficiency. Note that for this problem efficiency can be defined in terms of several costs, namely, the amount of storage incurred by the server or the client, the amount of computation involved on the server or the client sides in order to get the final output, and the amount of data communicated between the client and the server. The amount of storage incurred by the server is a crucial parameters. Other important costs are the communication complexity and the amount of computation the client incurs in order to retrieve the desired shortest path, as client-side machines are typically not as powerful as server-side machines. We start with discussing the most naive solutions, gradually augmenting them with simple strategies to improve their efficiency. To provide a summary of the efficiency of each method, we present all the relevant asymptotic costs described above for each method in Table I.

A. Trivial Solution: Downloading Entire Graph

A trivial $(1, 1)$-private solution is to simply have the client download the entire graph and compute the shortest path locally. Although the server does not need to perform any computation for this solution, the communication from server to client is significant, i.e., the entire graph of size $\approx \lambda \cdot n \cdot \delta$, and the client has to perform a non-trivial computation (e.g., Dijkstra which is $\mathcal{O}(m + n \log n)$) to retrieve the desired shortest path. In addition, the client also has to store the entire graph. Taking into account the fact that in most cases clients only sporadically send small amounts of queries, and also that the underlying network needs to be frequently updated to account for new users and channels, it is useful to push the heavy computation and storage burden to the server; the latter typically runs on machines with superior hardware and storage capacity.

[1]The name stands for private information retrieval for Lightning-like networks.

	Server storage	Communication	Server computation	Client computation	Client storage
Trivial (download entire graph)	$\mathcal{O}(n\lambda\delta)$	$\mathcal{O}(n\lambda\delta)$	$\mathcal{O}(1)$	$\mathcal{O}(m + n\log n)$	$\mathcal{O}(n\lambda\delta)$
Send all APSP	$\mathcal{O}(n^2 d\lambda)$	$\mathcal{O}(n^2 d\lambda)$	$\mathcal{O}(n^3)$	$\mathcal{O}(1)$	$\mathcal{O}(d\lambda)$
APSP + PIR	$\mathcal{O}(n^2 d\lambda)$	$\mathcal{O}(n\sqrt{d\lambda})$	$\mathcal{O}(n^3 + n^2 d\lambda)$	$\mathcal{O}(n\sqrt{d\lambda})$	$\mathcal{O}(d\lambda)$
APSP + Hub Labelling (HL)	$\mathcal{O}(nhd\lambda)$	$\mathcal{O}(nhd\lambda)$	$\mathcal{O}(n^3 + n^2 \max(d, \log n)\log d)$	$\mathcal{O}(h)$	$\mathcal{O}(hd\lambda)$
Our solution (APSP + PIR + HL)	$\boldsymbol{\mathcal{O}(nhd\lambda)}$	$\boldsymbol{\mathcal{O}(\sqrt{nhd\lambda})}$	$\boldsymbol{\mathcal{O}(n^3 + nhd\lambda + n^2 \max(d, \log n)\log d)}$	$\boldsymbol{\mathcal{O}(h + \sqrt{nhd\lambda})}$	$\boldsymbol{\mathcal{O}(hd\lambda)}$

TABLE I
ASYMPTOTIC SERVER AND CLIENT SIDE STORAGE, COMMUNICATION, AND COMPUTATION COST FOR THE TRIVIAL SOLUTION, THE APSP SOLUTION, THE APSP SOLUTION WITH A PIR, THE APSP SOLUTION WITH HUB LABELLING, AND OUR SOLUTION WHICH IS THE APSP SOLUTION WITH A PIR AND HUB LABELLING. n IS THE NUMBER OF VERTICES IN THE GRAPH, m IS THE NUMBER OF EDGES, δ IS THE AVERAGE DEGREE, λ IS THE LENGTH OF THE BINARY REPRESENTATION OF EACH VERTEX, d IS THE LENGTH OF THE LONGEST SHORTEST PATH IN THE GRAPH, AND h IS THE MAXIMAL HUB SET SIZE.

B. Server Computes All Pairs Shortest Paths

A natural $(1,1)$-private solution which pushes the burden of computation onto the server is for the server to precompute and store all the all pair shortest paths (APSPs). This solution gives a database with $N = n(n-1) \approx n^2$ entries (every pair of nodes) with entries of length $L = \lambda \cdot d$, i.e., the λ bit labels of the $\leq d$ nodes for each path.

Naturally the amount of computation the server would have to do to compute these APSPs ($\mathcal{O}(n^3)$) increases compared to the previous trivial $(1,1)$-private solution where the server does not have to perform any computation. Nevertheless this factor is not as crucial as the stored database size ($\mathcal{O}(n^2 d\lambda)$), which grows quadratically with n. Even for the moderate sized Lightning network with $n \approx 7000$, this gives an almost $2TB$ database (with $d = 9$ and $\lambda = 20$ bit labels).

In addition, to maintain privacy, the client cannot simply query the server for the desired index, as this blatantly reveals the path information. The client would therefore have to download the entire database from the server to hide their query. Thus, the price of constant client-side computation is an additional client-side communication cost of $\mathcal{O}(n^2 d\lambda)$ (see Table I).

C. Trivial PRDM from Private Information Retrieval (PIR) Using APSPs

In a (ℓ, k)-private information retrieval (PIR) scheme we have k servers, each holding a copy of a database DB which contains N entries, each L bits long. A client $\mathcal{C}$ who want to learn DB$[i]$ for some i, computes queries q_i to every $\mathcal{S}_i$, and then can (efficiently) compute DB$[i]$ from the answers $a_1, \dots, a_k$. The security property (ℓ-privacy) requires that any union of ℓ servers cannot learn anything about the query i.

The paper introducing PIR [12, Corollary 4.2.1.] has a particularly simple $(1,2)$-private PIR where the communication complexity is just $4\max\{L, \sqrt{N \cdot L}\}$, in particular, that's $4L$ if $L \geq N$.[2]

We can get a (ℓ, k)-private mechanism using a (ℓ, k)-private information retrieval on top of the APSP solution. Using a PIR protocol similar to [12] and assuming that the storage size of each shortest path is smaller that the total number of entries in the database, i.e. $d \cdot \lambda < n^2$, this solution reduces the communication cost between the client and servers from $\mathcal{O}(n^2 d\lambda)$ to $\mathcal{O}(n\sqrt{d\lambda})$ as shown in Table I. In addition, the client now does not need to download and store the database but just has to perform $\mathcal{O}(n\sqrt{d\lambda})$ XOR operations to get the desired shortest path so the client-side storage cost is just the cost of storing the path (i.e. $\mathcal{O}(d\lambda)$).

From Table I, we see that this solution incurs a larger asymptotic computation cost on both the server and client side compared to the vanilla APSP solution. Specifically, the server has to perform $\mathcal{O}(n^2 d\lambda)$ XOR operations upon each query from the client in addition to the initial $\mathcal{O}(n^3)$ computations to get the APSPs. These cost increases are not too alarming as, firstly, server-side computation costs are not as important as user-side computation and storage costs. More importantly, the length of the longest shortest path in the Lightning network d is very small and unlikely to increase much as the network grows due to the fact that the network has been maintaining a central core of highly connected nodes (see Section IV-B1), and we can reasonably assume $\lambda \ll n$.

The server-side storage requirements of this solution is unfortunately the same as the vanilla APSP solution and is the main drawback of this approach. In addition, the PIR protocol requires the servers to perform XOR computations linear in the size of the entire database in response to every query. Thus it would be highly beneficial if the database could be kept in memory. As outlined above, for Lightning that is already not possible with this approach.

D. Hub Labelling

In a hub labelling scheme [13] one preprocesses a graph such that later, one can efficiently find shortest paths in-between any two nodes. For some graphs, in particular street networks, the preprocessed data is *much* smaller than storing all pairwise shortest paths.

Concretely, given $G = (V, E)$ the idea is to precompute APSPs. Then compute for every vertex $v \in V$ a set of hubs $hub(v) \subset V$ such that for all $u, v \in V$, the shortest path from u to v (if G is directed also from v to u) contains a vertex that lies in the intersection of their hub sets $hub(u) \cap hub(v)$. This is the *covering property* of the hub sets.

[2]In this construction we think of DB as an $L \times N$ matrix when $L \geq N$, $q_1 \in \{0,1\}^N$ is random and $q_2 = q_1 \oplus e_i$ where e_i is the zero vector with the ith position set to 1. From the answers $a_i = \mathsf{DB} \cdot q_i$ the client computes $a_1 \oplus a_2 = \mathsf{DB} \cdot e_i = \underline{\mathsf{DB}[i] \ni \{0,1\}^L}$. If $L < N$ one does almost the same but thinks of DB as a $\sqrt{N \cdot L} \times \sqrt{N \cdot L}$ matrix.

The preprocessed DB now contains, for every $v \in V$ its hub set $hub(v)$ and for every $u \in hub(v)$ the shortest path from v to u (and u to v).

If $h = \max_{v \in V}\{|hub(v)|\}$ is the size of the largest hub, we can think of the preprocessed data DB as a database with $n = |V|$ entries of length $L = h \cdot \lambda \cdot d$ bits, i.e., for every $v \in V$ we have $hub(v) \leq h$ paths, each of length $\leq d$, and thus the asymptotic server-side storage drops from $\mathcal{O}(n^2 d\lambda)$ to $\mathcal{O}(nhd\lambda)$. For $h \ll n$, which is the case for the Lightning network (see Section V), this allows us to achieve at least an order of magnitude reduction in server-side storage costs.

Compared to the vanilla APSP solution, adding hub-labelling increases the asymptotic server-side computation by an additive factor of $n^2 \max(d, \log n) \log d$. The increase in computation cost is due to the hitting set computation for each radius. Nevertheless we note that the increase is not big as $d \ll n$ in the Lightning network and so the main cost is still the $\mathcal{O}(n^3)$ from the APSP computation. In addition, server-side computation is not as important as server-side storage, and these computations are also not costs incurred with every query but need only to be be done relatively infrequently, for instance every two weeks or so to take into account changes in the underlying network topology.

The main drawback of this solution is that the client needs to download the entire database to maintain the privacy guarantee, which incurs an asymptotic communication cost of $\mathcal{O}(nhd\lambda)$.

IV. The LightPIR Approach

Given the above considerations, we are now ready to present our proposed solution, *LightPIR*.

A. Basic Approach

In a nutshell, our construction of a private route discovery mechanism augments the basic APSP solution with both a PIR protocol (specifically a 2-server RAID-PIR protocol [14] which is similar in flavour to [12]) and hub labelling. A client who wants to learn the shortest path from u to v will query the PIR servers for the u, v entries, which will be the hub sets of u and v respectively. Then, the client should perform a set intersection of the two hub sets, which will be non-empty by the covering property of these hubs, and determine the $w \in hub(u) \cap hub(v)$ for which the length of the path $u \rightarrow w$ plus the length of $w \rightarrow v$ is minimized. The path $u \rightarrow w \rightarrow v$ is the sought shortest path from u to v.

LightPIR reaps the benefit of a lower asymptotic storage cost of $\mathcal{O}(nhd\lambda)$ from hub labelling and a lower asymptotic communication cost of $\mathcal{O}(\sqrt{nhd\lambda})$ from the 2-server PIR protocol. The composition of these two techniques also lowers the per query asymptotic server-side computation due to the PIR protocol from $\mathcal{O}(n^2 d\lambda)$ XOR operations to $\mathcal{O}(nhd\lambda)$. Compared to the vanilla APSP solution, the user now has to perform some computation to determine the desired shortest path. This computation is extremely simple, basically $\mathcal{O}(\sqrt{nhd\lambda})$ XOR operations to privately recover the query result from the servers and a set intersection on two moderately sized sets of at most $h \ll n$ to get the shortest path, and thus can be performed by even the weakest clients.

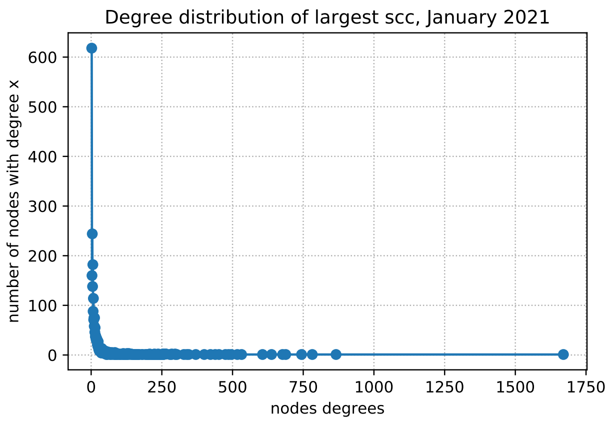

Fig. 1. A plot showing the number of nodes for each degree in the largest strongly component (scc) of the Lightning network on January 2021. The distribution of degrees is quite similar in Lightning network snapshots taken in dates within the past two years.

B. Optimization of Hub Sets

In order to further reduce the database storage, *LightPIR* relies on a heuristic which leverages the specific topology of the Lightning network to optimize the size of hub sets. We first begin with a brief description of the Lightning network topology and then describe the heuristic.

1) Network Topology Characteristics: To understand and exploit the specific topological characteristics of payment channel networks, we downloaded a number of historical snapshots of the Lightning network taken over the past two years from the Lightning github repositories [15], [16]. For instance, a representative network snapshot on January 2021 comprises of 6,376 nodes, 39,993 channels, and 3,650 strongly connected components, although except for the largest one, almost all of the rest components have only a very small number of nodes. We thus focus our analysis on the largest connected component of each snapshot, however our results can be applied to any connected component. The one of January 2021 has 2,707 nodes and comprises a majority (31,350) of the total number of channels.

As can be seen from Figure 1, the majority of the nodes in the largest component of the Lightning network have very low degrees and about 100 nodes have degrees of > 100. The largest component also has a relatively small diameter of 9. We can therefore infer a large degree of centralisation in the Lightning network, with a few central nodes of high degree connected to each other as well as connected to shorter chains of vertices (see Figure 2).

A channel (i.e. an edge) between two nodes in the network is weighted by its channel fee. This usually comprises of a base fee which is independent of the amount one wishes to move through the channel, and an amount-specific rate fee which depends on the amount sent across the channel. Table II

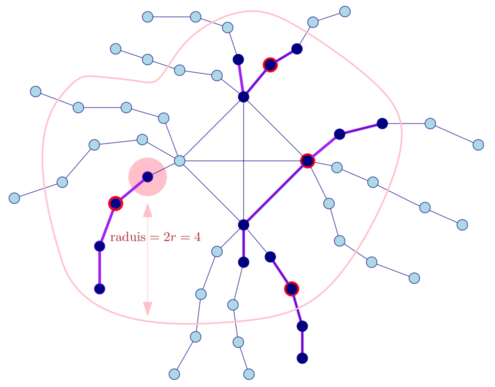

Fig. 2. A graph created from star graphs where the centers of the graphs form a clique. A few paths of length in $[r+1, 2r]$ for $r=2$ with a (red) cover point on each path are shown. A ball (neighborhood) of radius $2r = 4$ is shown, and it contains a substantial fraction of the graph, and thus the cover points.

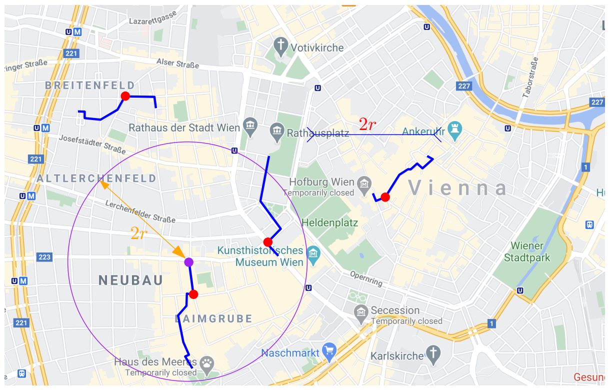

Fig. 3. A street network with 4 shortest paths of length in $(r, 2r]$ (for $r \approx 500m$) and a cover point of those paths (in red) highlighted. A ball (neighborhood) with radius 2r (one with center at the starting point of a path shown in purple) will only contain cover points resulting from relatively few paths "in the neighbourhood".

includes a number of network characteristics for a number of snapshots of the Lightning network over the past two years.

2) Hub Set Computation: Greedy Heuristic: One way of computing the hubs for each node in the network is to make use of the highway dimension (HD) algorithm [17], [18]. Intuitively, HD is a graph metric that captures the size of a subset of nodes that intersects all shortest paths of non-trivial length. In order to compute the HD, we compute sets of nodes that intersect all paths of size $(r, 2r]$ for a small set of radii and measure how sparse these sets are in the network. In the following, we explain how we use these sets to compute the hubs for each network node (Algorithm 1).

In [17], the authors give a definition of HD based on shortest-path covers and also give an approximation algorithm to compute the HD of a network using this definition. In [18], the authors present a modification that improves their earlier work, which also applies to the shortest-path covers algorithm. We further modify this approximation algorithm with a heuristic from observing the topology of the Lightning network, as shown in Algorithm 1, to both return a hub set for each node in the network, and also maintain an upper bound on the HD of the network. That is, we always overapproximate the HD but never underapproximate it.

We present a few necessary ingredients from [18, Section 4], before elaborating on our heuristic. Let $G = (V, E)$ be the network topology. The neighborhood $N_x(u)$ of a node $u \in V$ for a radius x is the set of all nodes that appear in a path P, such that $v \in P$ and $|P| \leq x$, where $|P|$ denotes the length of the path P ($N_x(v)$ contains the same number of nodes as the ball with center u and radius x). Moreover, a set C of nodes is a (k, r)-shortest path cover ((k, r)-SPC) of G if and only if (I) for each shortest path P, such that $r < |P| \leq 2r$, $P \cap C \neq \emptyset$ and (II) $\forall u \in V$, $|C \cap N_{2r}(u)| \leq k$. Then, if G has HD h, then for any $r > 0$ there exists an (h, r)-SPC of G. Thus, the authors claim that by using the greedy $\mathcal{O}(\log n)$ approximation for computing hitting sets, we can combine the above into an algorithm that computes a bound of HD in $\mathcal{O}(h \log n)$, where $h = \mathcal{HD}(G)$ is the highway dimension of G. Specifically, given a hitting set $cover[r]$ of all pairs shortest paths of length in $(r, 2r]$, they make sure that condition I holds and then they can fulfill Condition II by taking all $cover[r] \cap N_{2r}(v)$ intersections for all $v \in V$ and setting k to be the maximum size of those intersections. As mentioned in [13], it suffices to compute the above with the set of radii $r \in \{2^i \mid i \in \mathbb{N} \wedge 2^i \leq diameter(G)\}$. Of course, an exact hitting set algorithm would give a smaller $cover[r]$, but computing it is NP-hard.

Our heuristic is based on the key observation that, due to the centralised nature of the network (low diameter and a core of highly connected nodes), high degree nodes in the Lightning network appear in a vast majority of shortest paths, as opposed to the less dense and centralised road networks for which HD was originally designed (Figure 3). Thus many neighborhoods for center u and radius $2r$ include these high degree nodes, which work as gateways to many remote nodes.

We noticed that for network topologies similar to the one of Lightning removing the base nodes after computing the covers can significantly reduce the HD (Figure 2). Let $base(\ell)$ be the set of the ℓ nodes with the highest degree (in short, $base$). In our heuristic, we adapt the HD computation by first computing the cover sets as in [17], [18], but then removing the base nodes and all the edges connected to the base nodes to obtain the topology $\overline{G} = G \setminus base$, in which we compute the per-node hubs as follows. That is the hub set of node u for radius $2r$ is defined by $\underline{hubs\_in\_neighb}(u, r) = base \cup (N_{2r}^{\overline{G}}(u) \cap cover[r])$, where $N_{2r}^{\overline{G}}(u)$ is the neighborhood $N_{2r}(u)$ computed in $\overline{G}$. Note that [18] (in addition to [17]) requires that the path weights are unique integers, but this is easily achieved with insignificant perturbations of edge weights (in our case we initially set the weights to be the channel base fees and then perturbed them).

We use simple binary search to find the base size ℓ, but more elaborate methods [19] can be used as well. We observed that there's not necessarily only one local minimum (see the

Algorithm 1: *LightPIR*'s hub set optimization

```
Input: $G = (V, E)$, the network topology and $\ell$, the number
       of high degree nodes to use as base
Output: $\overline{\mathcal{HD}}(G)$ (bound on $\mathcal{HD}(G)$) and $hubs(u)$, $\forall u \in V$
1  $APSP \leftarrow allPairsShortestPaths(G)$;
2  $radii \leftarrow \{2^i \mid i \in \mathbb{N} \wedge 2^i \leq diameter(G)\}$;
3  $base \leftarrow highDegreeNodes[0 : \ell - 1]$;
   /* compute shortest path sets                    */
4  for $r \in radii$ do $shortestPaths(r) \leftarrow \emptyset$;     /* init */
5  for $path \in APSP$ do
6      for $r \in radii$ do
7          if $r < |path| \leq 2r \wedge base \cap path = \emptyset$ then
8              $shortestPaths(r).add(path)$;
   /* compute hitting sets and covers               */
9  for $r \in radii$ do
10     $cover[r] \leftarrow base \cup hittingSet(shortestPaths(r))$;
   /* compute $N_{2r}^{\overline{G}}(u) \cap cover[r]$ sizes & hubs    */
11 $\overline{G} \leftarrow G \setminus base$;
12 for $u \in V$ do $hubs(u) \leftarrow \emptyset$;            /* init hubs */
13 for $(u, r) : u \in V \wedge r \in radii$ do
14     $hubs\_in\_neighb(u, r) \leftarrow base \cup (N_{2r}^{\overline{G}}(u) \cap cover[r])$;
15     $hubs(u) \leftarrow hubs(u) \cup (hubs\_in\_neighb(u, r))$;
16 $\overline{\mathcal{HD}}(G) = \max_{u,r} |hubs\_in\_neighb(u, r)|$;
```

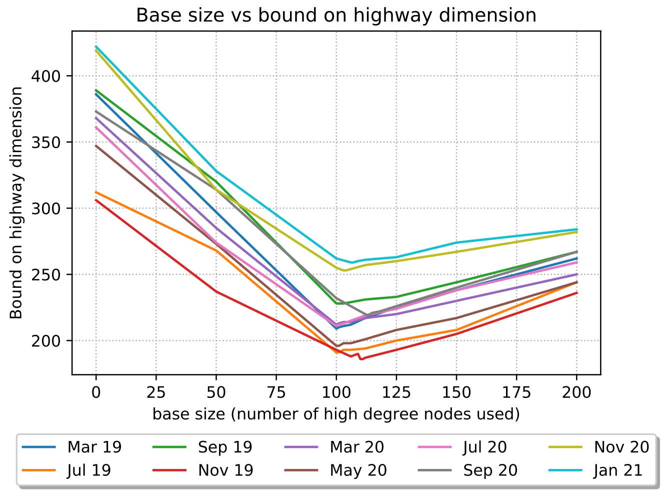

Fig. 4. The x-axis shows the different base set sizes tested (a base with size ℓ includes the ℓ nodes with the highest degree) and the y-axis shows the corresponding highway dimension bound. The baseline computation corresponds to the values for base size 0, which are the highest. We run our heuristic with the base size that minimizes the highway dimension bound (thus also the hub database) and reported the results in Table II.

red curve in Fig. 4), the minima are very close together, so any of them is almost as good as the global one. Moreover, our heuristic can be applied to all connected components. Our general approach works for any payment (or other) network which has small hub sizes. The heuristic of removing a base set of some size ℓ we propose works well for the Lightning network in its current and past form, but for other networks new ideas might be necessary to find small hub sets. We illustrate this heuristic in Algorithm 1 and show that it correctly computes an upper bound $\overline{\mathcal{HD}}(G)$ of $\mathcal{HD}(G)$ in Lemma 1.

Lemma 1. *Algorithm 1 computes an upper bound of $\mathcal{HD}(G)$.*

Proof. We prove that for any two communicating nodes s and d, there exists a hub node in the set of hubs that lies on the shortest path between s and d, $SP(s, d)$. Let $|SP(s, d)| = k$ and let r be the radius for which $r < k \leq 2r$ (there has to be at least one such radius as our set of radii covers all shortest paths). If $SP(s, d) \subset N_{2r}^{\overline{G}}(s)$, i.e. if $SP(s, d)$ entirely appears in the neighborhood with center s and radius $2r$ in the resulting topology *after* removing *base* from G, then the hub for s and d is in $SP(s, d) \cap cover[r]$, as $cover[r]$ hits all shortest paths of length in $(r, 2r]$. Otherwise, if $SP(s, d)$ does not entirely appear in $N_{2r}^{\overline{G}}(s)$, then there exist at least one node in *base* that intersects $SP(s, d)$. Since *base* is included in the set of hubs (Algorithm 1, line 14), then the proof is complete. □

Note that from the proof of Lemma 1, we guarantee the covering property and hence correctness of the hub sets: the hub set of any node covers all shortest paths from that node to any other node in the network. This means that any two hub sets in a connected network must certainly have a non-empty intersection.

Algorithm 1's time complexity is computed as follows. The APSP step takes $\mathcal{O}(n^3)$ and the partitioning of APSP according to their length takes $\mathcal{O}(n^2 \log d)$. The computation of covers, which includes the computation of hitting sets (greedy heuristic costs $\mathcal{O}(n^2 \max(d, \log n))$), takes $\mathcal{O}((n^2 \max(d, \log n)) \log d)$ and the hub computation takes $\mathcal{O}(n \log d)$. Thus, in total $\mathcal{O}(n^3 + n^2 \log d + (n^2 \max(d, \log n)) \log d + n \log d) = \mathcal{O}(n^3 + n^2 \max(d, \log n) \log d)$ (cf. Table I). In practice, the diameters of the Lightning network snapshots that we used to test Algorithm 1 were quite small, thus the running time was dominated by the APSP computation (which was a matter of minutes).

V. Empirical Evaluation

To provide a first empirical evaluation of our approach, we implemented *LightPIR* (the code will be released together with this paper) and ran experiments on historical snapshots of the Lightning network from the Lightning network gossip repository [16]. In general, we find that our approach indeed achieves an improvement by an order of magnitude in terms of storage requirements and a roughly 40% reduction in the highway dimension of each snapshot, which directly translates to a reduced hubs database.

a) Setup: In our experiments, we used a virtual machine with 24 cores (Intel Xeon CPUs E5-2650 v4 at 2.20GHz) and 16 GB RAM, running Ubuntu 18.04.1 LTS. Our experiments were implemented in Python (version 3.7.6) using networkx [20] among other libraries. The implementation of Algorithm 1 and the experiments setup are available in [21].

TABLE II

LIGHTNING NETWORK SNAPSHOT CHARACTERISTICS AND EVALUATION RESULTS (SCC STANDS FOR STRONGLY CONNECTED COMPONENT(S))

date	**nodes**	**channels**	**scc**	**largest scc**								
				nodes	**channels**	**diam.**	**baseline HD bound**	**heuristic HD bound**	**base size**	**max hub set size**	**hub DB size (MB)**	**exec. time (h)**
13-03-2019	3480	45071	994	2480	42025	7	386	209	100	271	1.5	2.03
13-07-2019	4469	53084	1804	2661	46733	7	312	191	100	248	1.4	1.41
13-09-2019	4733	51508	2027	2706	45774	7	389	228	100	304	2.1	2.64
13-11-2019	4598	44784	2158	2422	38011	6	306	186	110	235	1.2	1.31
13-03-2020	5070	48340	2356	2712	41571	6	368	212	100	270	1.7	2.67
13-05-2020	5557	49569	2570	2985	42093	8	347	196	100	257	1.6	1.98
13-07-2020	5888	51932	2748	3136	44005	7	361	211	100	286	1.8	2.27
13-09-2020	5972	51783	2800	3171	43844	7	373	220	112	285	1.8	2.47
13-11-2020	6074	48361	2978	3089	40695	8	419	253	103	315	2.3	2.96
13-01-2021	6376	39993	3650	2707	31350	9	422	259	106	314	1.8	1.83

b) Data: The extracted snapshots range from March 2019 to the current state (January 2021) with the smallest snapshot having 3,480 nodes and the largest having 6,376 nodes. We ignore early or intermediate snapshots that have too few nodes. We performed our empirical evaluation on the largest strongly connected component (SCC) in the snapshot and we found that although the network size almost doubles from March 2019 to January 2021, the number of nodes in the largest SCC remains pretty consistent across all snapshots ($\approx$ 2500 nodes). The first seven columns of Table II give a complete view of the datasets used.

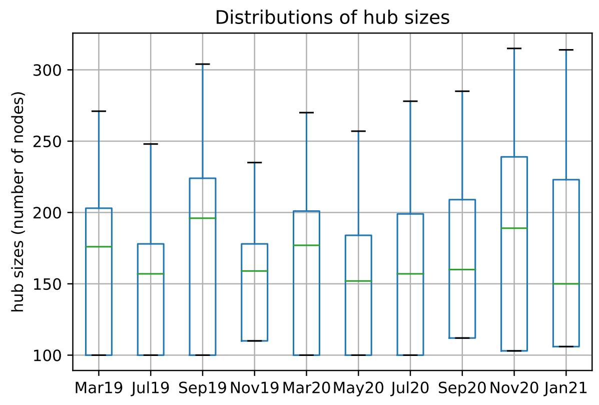

Fig. 5. Box plots of hub sizes (in number of nodes) for a number of snapshots of the Lightning network. The green line shows the second quartile (median), the bottom and top of each box show the first and third quartiles, respectively, and the extended lines out of each box (whiskers) end at the min (bottom) and max (top) values of the datasets.

c) Results: We computed the optimal base size for the largest SCC in each snapshot by running a binary search to find the size of the base set that minimises the approximate highway dimension of the SCC as computed by our algorithm (ℓ in Algorithm 1's input). The baseline algorithm of [17], [18] is the one where $\ell = 0$, i.e., there the base set is empty. As shown in Figure 4, for most Lightning network snapshots a base size of around 100 nodes minimises the approximate highway dimension of the largest SCC. In fact, the reduction in highway dimension is around 40% across all snapshots when compared to the baseline (see Table II baseline HD bound vs heuristic HD bound).

Across all snapshots the median hub set size as computed by our algorithm is between 150 to 200 nodes, with the maximum hub set size being 315 nodes. The distribution of hub sizes for each snapshot is shown in Figure 5. Given that the average size for storing each snapshot is about 13MB, our method provides an overall reduction of an order of magnitude in terms of the hub set sizes (cf. second last column of Table II). Due to the inclusion of the base, the minimum hub set size is at least 100. Specifically, as shown in Table II, the average database size across all snapshots is 1.65MB.

The running time for each snapshot is about two hours (last column of Table II). This includes the search for the base size that minimizes the HD (and thus the hubs database). Thus, it is possible to run our algorithm periodically to update the hub sets as the lightning network topology changes. It would also be useful to see how the run time of our algorithm scales with the size of the lightning network (in particular, the largest SCC). We leave this comparison for future work when the size of the largest SCC significantly increases.

VI. Related Work and Novelty

While the security of blockchains, e.g., related to the underlying consensus protocols [22], [23] but also related to the interconnecting network [24], has been extensively investigated, much less is known about the security of the recent concept of payment channel networks. The security aspects of payment channel networks considered so far mainly revolve around connectivity [25] and the multi-hop routing of payments [8], [26]. We refer the reader to the survey [27] for an overview of specific threats related to hot wallets, mass exits, synchronizing protocols, among others.

Prior work on route discovery primarily focused on efficiency. For example, SpeedyMurmurs [4] (which extends VOUTE [28]) and SilentWhispers [6] rely on clever protocols to speed up route discovery by two orders of magnitude, while maintaining the same success ratio. Another interesting approach is pursued in the SpiderNetwork [7]: the payments are split into units and the route discovery algorithm routes each of them individually (similarly to a packet-switched network). This method however also does not account for privacy and cost effectiveness. To trade off high throughput and probing overhead Flash [29] distinguishes between mice and elephant payments, splitting elephant flows across multiple flows. Flare [3] proposes to improve scalability of route discovery by determining beacon nodes, an approach which also motivates our model in this paper. To overcome the cost overheads of multi-hop routing, thee off-chain channel network Perun [30] introduces a notion of virtual payment channels; while this allows to avoid intermediaries for each payment, it does not solve the discovery problem.

Private route discovery on street networks has been explored in [31]. Their work is similar to ours in that they also use a PIR protocol to privately retrieve shortest path entries in a database. The main difference is the database compression technique: in [31] the authors develop a compression algorithm specific to the grid-like structure of street networks in North America (i.e. each node has degree 4 corresponding to the 4 cardinal directions), whereas we use hub labelling which has good compression guarantees on generic networks and we develop an additional heuristic based on the topology of the Lightning network to get small hubs set sizes.

Our work is motivated by recent empirical work demonstrating issues with confidentiality during the route discovery process in the Lightning network [10], [11]. To our knowledge, the only work suggesting a more secure route discovery mechanism is MAPPCN [32], which however does not account of the information leaking problem addressed in this paper.

Our paper also builds upon classic works on efficient shortest path computations on extremely large networks [33]–[35]. Of particular interest in our context are algorithms based on transit node routing (TNR) [36], which focus on networks in which a small set of vertices covers most shortest paths, i.e., networks with small so-called highway dimension. Recently, Abraham et al. [17], [18] proved theoretical guarantees on the efficiency of common shortest path search algorithms for graphs which satisfy this property. The hub-labelling algorithm [13], [37] is a variant of TNR and it was shown to be significantly faster compared to other state of the art shortest path algorithms when tested on real world road networks [13]. In this paper, we build upon this approach to scalable routing, tailoring it to the specific topological properties of payment channel networks.

Information theoretic PIR protocols were first introduced by Chor et al. [12]. With 2 servers and a simple linear bit summation scheme, they achieve perfect privacy with a communication complexity of $\mathcal{O}(n^{1/3})$. Since then, follow up work in this area has mainly focused on lowering the communication complexity [14], [38]–[41]. Recently, Beimel et al. [42] achieved a communication complexity of $n^{\mathcal{O}(\frac{\log\log k}{k\log k})}$ with k servers using locally decodable codes. Under the conjecture that there are infinitely many Mersenne primes, Yekhanin [43] removed the dependency on large k in the communication complexity bound and achieved a communication complexity of $n^{\mathcal{O}(\frac{1}{\log\log n})}$ for a 3 server PIR protocol. Although these newer protocols achieve a much lower communication complexity compared to the original protocol, the techniques used in these protocols are also much more complicated. As these benefits from a lower communication complexity are not that pertinent to us, we focus on simpler PIR protocols which are similar to [12] in this paper.

PIR protocols have already been successfully deployed in many contexts, such as DNS [44], private presence service [45], and private retrieval of security updates [46]. However, we are not aware of any applications of PIRs in the context of payment channel networks.

VII. Future Work

We understand our work as a first step and would like to highlight three interesting directions of future work.

Firstly, when shortest paths are not unique, most available algorithms for selecting a shortest path out of a set of paths of equal length just select a random one from the set. In terms of reducing the size of the hub set, one can design smarter algorithms that do not select the path randomly, but optimise for paths with a larger vertex overlap with the set of other paths already selected. This ensures that when we greedily select vertices to be in the hub set, we would select vertices that cover a larger set of paths and so could potentially decrease the size of the hub set.

Secondly, our current heuristics to calculate the highway dimension of the network and to reduce the size of the hub sets are heavily inspired by the topology of the Lightning network. It would be interesting to track the topology of the Lightning network as the network grows in size, to see if there are significant changes and to think about how to incorporate these changes in the network topology to further optimise the hub set sizes.

Lastly, we note that the network topology and thus the hub sets change with the amount of Bitcoins a user wants to send, e.g., due to the capacity limitation of channels in the network (i.e. the amount of Bitcoins each channel can forward). For instance the shortest path from Alice to Bob might go through Charlie when Alice wants to send 1 Btc but the path might have to go through Dave when Alice wants to send 10 Btc because Charlie cannot forward more than 1 Btc. Additionally, the topology could also change due to the way the fees are calculated. The fees on channels comprise of a base fee and a rate fee which depends on the amount sent over the channel and thus the shortest path information could change drastically between sending small or large amounts of Bitcoin. Since we do not yet take into account fees, our current work can be seen as optimised for sending a fixed amount of Bitcoin and thus an interesting future direction would be to take into account the amount of Bitcoins a user wants to send and design a similarly efficient and private method of extracting shortest paths.

References

[1] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais, "Sok: Off the chain transactions." *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 360, 2019.

[2] J. Poon and T. Dryja, “The bitcoin lightning network: Scalable off-chain instant payments,” 2016.

[3] P. Prihodko, S. Zhigulin, M. Sahno, A. Ostrovskiy, and O. Osuntokun, “Flare: An approach to routing in lightning network,” *White Paper*, 2016.

[4] S. Roos, P. Moreno-Sanchez, A. Kate, and I. Goldberg, “Settling payments fast and private: Efficient decentralized routing for path-based transactions,” *arXiv preprint arXiv:1709.05748*, 2017.

[5] B. Weekly, “Outsourcing route computation with trampoline payments,” in *https://bitcointechweekly.com/front/outsourcing-route-computation-with-trampoline-payments/*, 2019.

[6] G. Malavolta, P. Moreno-Sanchez, A. Kate, and M. Maffei, “Silentwhispers: Enforcing security and privacy in decentralized credit networks.” in *NDSS*, 2017.

[7] V. Sivaraman, S. B. Venkatakrishnan, M. Alizadeh, G. Fanti, and P. Viswanath, “Routing cryptocurrency with the spider network,” in *Proc. 17th ACM Workshop on Hot Topics in Networks*, 2018, pp. 29–35.

[8] S. Tochner, A. Zohar, and S. Schmid, “Route hijacking and dos in off-chain networks,” in *Proc. ACM Conference on Advances in Financial Technologies (AFT)*, 2020.

[9] R. Dingledine, N. Mathewson, and P. Syverson, “Tor: The second-generation onion router,” Naval Research Lab Washington DC, Tech. Rep., 2004.

[10] S. Tochner and S. Schmid, “On search friction of route discovery in offchain networks,” in *Proc. IEEE International Conference on Blockchain (Blockchain)*, 2020.

[11] U. Nisslmueller, K.-T. Foerster, S. Schmid, and 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.

[12] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, “Private information retrieval,” *J. ACM*, vol. 45, no. 6, pp. 965–981, 1998. [Online]. Available: https://doi.org/10.1145/293347.293350

[13] I. Abraham, D. Delling, A. Goldberg, and R. Werneck, “A hub-based labeling algorithm for shortest paths in road networks,” 01 2011, pp. 230–241.

[14] D. Demmler, A. Herzberg, and T. Schneider, “RAID-PIR: practical multi-server PIR,” in *Proc. 6th edition of the ACM Workshop on Cloud Computing Security, CCSW ’14, Scottsdale, Arizona, USA, November 7, 2014*, G. Ahn, A. Oprea, and R. Safavi-Naini, Eds. ACM, 2014, pp. 45–56. [Online]. Available: https://doi.org/10.1145/2664168.2664181

[15] “Lightning network daemon,” https://github.com/lightningnetwork/lnd.

[16] C. Decker, “Lightning network research; topology, datasets,” https://github.com/lnresearch/topology, accessed: 2020-10-01.

[17] I. Abraham, A. Fiat, A. Goldberg, and R. Werneck, “Highway dimension, shortest paths, and provably efficient algorithms,” in *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA10)*. Society for Industrial and Applied Mathematics, January 2010.

[18] I. Abraham, D. Delling, A. Fiat, A. V. Goldberg, and R. F. Werneck, “Vc-dimension and shortest path algorithms,” in *International Colloquium on Automata, Languages, and Programming*. Springer, 2011, pp. 690–699.

[19] C. Audet and W. Hare, “Derivative-free and blackbox optimization,” 2017.

[20] A. A. Hagberg, D. A. Schult, and P. J. Swart, “Exploring network structure, dynamics, and function using networkx,” in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11 – 15.

[21] “LightPIR Bitbucket repository,” https://bitbucket.org/iosifsalem/lightpir/.

[22] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, “On the security and performance of proof of work blockchains,” in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 3–16.

[23] K. Wüst and A. Gervais, “Ethereum eclipse attacks,” ETH Zurich, Tech. Rep., 2016.

[24] A. Gervais, H. Ritzdorf, G. O. Karame, and S. Capkun, “Tampering with the delivery of blocks and transactions in bitcoin,” in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 692–705.

[25] E. Rohrer, J. Malliaris, and 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.

[26] G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, and 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.

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

[28] S. Roos, M. Beck, and T. Strufe, “Voute-virtual overlays using tree embeddings,” *arXiv preprint arXiv:1601.06119*, 2016.

[29] P. Wang, H. Xu, X. Jin, and T. Wang, “Flash: efficient dynamic routing for offchain networks,” in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, 2019, pp. 370–381.

[30] S. Dziembowski, L. Eckey, S. Faust, and D. Malinowski, “Perun: Virtual payment hubs over cryptocurrencies,” in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 106–123.

[31] D. J. Wu, J. Zimmerman, J. Planul, and J. C. Mitchell, “Privacy-preserving shortest path computation,” *CoRR*, vol. abs/1601.02281, 2016. [Online]. Available: http://arxiv.org/abs/1601.02281

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

[33] A. V. Goldberg and C. Harrelson, “Computing the shortest path: *A* search meets graph theory,” in *Proc. Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005*.

[34] U. Lauther, “An experimental evaluation of point-to-point shortest path calculation on road networks with precalculated edge-flags,” in *The Shortest Path Problem, Proc. of a DIMACS Workshop, 2006*, C. Demetrescu, A. V. Goldberg, and D. S. Johnson, Eds., 2006.

[35] R. J. Gutman, “Reach-based routing: A new approach to shortest path algorithms optimized for road networks,” in *Proceedings of the Sixth Workshop on Algorithm Engineering and Experiments and the First Workshop on Analytic Algorithmics and Combinatorics, New Orleans, LA, USA, January 10, 2004*, L. Arge, G. F. Italiano, and R. Sedgewick, Eds. SIAM, 2004, pp. 100–111.

[36] H. Bast, S. Funke, D. Matijevic, P. Sanders, and D. Schultes, “In transit to constant time shortest-path queries in road networks,” in *Proceedings of the Nine Workshop on Algorithm Engineering and Experiments, ALENEX 2007, New Orleans, Louisiana, USA, January 6, 2007*. SIAM, 2007. [Online]. Available: https://doi.org/10.1137/1.9781611972870.5

[37] I. Abraham, D. Delling, A. V. Goldberg, and R. F. F. Werneck, “Hierarchical hub labelings for shortest paths,” in *Algorithms - ESA 2012 - 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings*, ser. Lecture Notes in Computer Science, L. Epstein and P. Ferragina, Eds., vol. 7501. Springer, 2012, pp. 24–35. [Online]. Available: https://doi.org/10.1007/978-3-642-33090-2_4

[38] A. Beimel and Y. Ishai, “Information-theoretic private information retrieval: A unified construction,” in *In Proceedings of 28th ICALP*, 2001, pp. 912–926.

[39] A. Beimel, Y. Ishai, E. Kushilevitz, and J. Raymond, “Breaking the o(n1/(2k-1)) barrier for information-theoretic private information retrieval,” in *43rd Symposium on Foundations of Computer Science (FOCS 2002)*. IEEE Computer Society, 2002, pp. 261–270.

[40] Z. Dvir and S. Gopi, “2-server pir with subpolynomial communication,” *Journal of the ACM (JACM)*, vol. 63, no. 4, pp. 1–15, 2016.

[41] K. Efremenko, “3-query locally decodable codes of subexponential length,” *SIAM J. Comput.*, vol. 41, no. 6, pp. 1694–1703, 2012. [Online]. Available: https://doi.org/10.1137/090772721

[42] A. Beimel, Y. Ishai, E. Kushilevitz, and J. Raymond, “Breaking the o(n/sup 1/(2k-1)/) barrier for information-theoretic private information retrieval,” *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pp. 261–270, 2002.

[43] S. Yekhanin, “Towards 3-query locally decodable codes of subexponential length,” *J. ACM*, vol. 55, no. 1, Feb. 2008. [Online]. Available: https://doi.org/10.1145/1326554.1326555

[44] F. Zhao, Y. Hori, and K. Sakurai, “Two-servers PIR based DNS query scheme with privacy-preserving,” in *The 2007 International Conference on Intelligent Pervasive Computing, IPC 2007*. IEEE Computer Society, 2007, pp. 299–302.

[45] N. Borisov, G. Danezis, and I. Goldberg, “Dp5: A private presence service,” *Proceedings on Privacy Enhancing Technologies*, vol. 2015, pp. 24 – 4, 2015.

[46] J. Cappos, “Avoiding theoretical optimality to efficiently and privately retrieve security updates,” vol. 7859, 04 2013, pp. 386–394.