Cross-Layer Deanonymization Methods in the Lightning Protocol

Matteo Romiti
Austrian Institute of Technology
matteo.romiti@ait.ac.at

Friedhelm Victor
Technische Universität Berlin
friedhelm.victor@tu-berlin.de

Pedro Moreno-Sanchez
Technische Universität Wien
pedro.sanchez@tuwien.ac.at

Bernhard Haslhofer
Austrian Institute of Technology
bernhard.haslhofer@ait.ac.at

Matteo Maffei
Technische Universität Wien
matteo.maffei@tuwien.ac.at

Abstract

Payment channel networks (PCNs) have emerged as a promising alternative to mitigate the scalability issues inherent to cryptocurrencies like Bitcoin and are often assumed to improve privacy, as payments are not stored on chain. However, a systematic analysis of possible deanonymization attacks is still missing. In this paper, we focus on the Bitcoin Lightning Network (LN), which is the most widespread implementation of PCNs to date. We present clustering heuristics that group Bitcoin addresses, based on their interaction with the LN, and LN nodes, based on shared naming and hosting information. We also present cross-layer linking heuristics that can, with our dataset, link 43.7% of all LN nodes to 26.3% Bitcoin addresses interacting with the LN. These cross-layer links allow us to attribute information (e.g., aliases, IP addresses) to 17% of the Bitcoin addresses contributing to their deanonymization. Further, we find the security and privacy of the LN are at the mercy of as few as five actors that control 34 nodes and over 44% of the total capacity. Overall, we present the first quantitative analysis of the security and privacy issues opened up by cross-layer interactions, demonstrating their impact and proposing suitable mitigation strategies.

1 Introduction

Payment channel-networks (PCNs) have emerged as a promising alternative to mitigate the scalability issues with current cryptocurrencies. These *layer-2* protocols, built on-top of *layer-1* blockchains, allow users to perform transactions without storing them on the blockchain. The idea is that two users create a funding transaction that locks funds (e.g., Bitcoins), thereby creating a payment channel between them [5]. Further payments no longer require on-chain transactions but rather peer-to-peer mutual agreements on how to distribute the coins locked in the channel. At any point, both users can decide to close the channel by creating a settlement transaction that unlocks the coins and distributes them according to the last agreed balance. Interestingly, two users that do not share a payment channel between them can route a transaction through a path of open payment channels.

While there are different payment channel designs, the Bitcoin Lightning Network (LN) [20] is the most widespread PCN implementation to date. At the time of writing, according to 1ml.com, the LN features a network of $7,248$ public active nodes, $36,794$ channels and a total capacity of more than 953.06 Bitcoins, worth $8,980,445$ USD.

Apart from scalability, PCNs are considered beneficial to improve the well-known lack of privacy of cryptocurrencies, where the anonymity claim stemming from the usage of pseudonyms in on-chain transactions has been largely refuted from both academia and industry [12]. The key to an effective deanonymization of Bitcoin pseudonyms lies in heuristic methods, which cluster addresses that are likely controlled by the same entity [16]. In practice, entities correspond to user wallets or software services (e.g., hosted wallet, exchange) that control private keys on behalf of their users. Depending on how a Bitcoin user is spending her funds from a software *wallet*, she can end up controlling several Bitcoin entities.

As only funding transactions and settlement transactions used to open and close channels are available on-chain, the LN is considered one of the technologies that *greatly improves privacy*. In this work, we refute this belief spread in the blockchain community by identifying and quantitatively analyzing the security and privacy leakages affecting the LN and consequently the privacy of Bitcoin addresses.

Recent research [7, 15, 23, 24] focused mostly on the PCN layer and showed that Denial of Service attacks targeted to specific nodes in the LN can effectively block the capacity at their channels and correspondingly isolate them from the network. Also, the closely related work by [19] investigated the links between on-chain transactions and LN channels. However, we notice the lack of research on how to cluster LN nodes controlled by the same user as well as how to unequivocally link such off-chain nodes with Bitcoin addresses. This is a challenging task as such links are not provided in the PCN protocol and they would severely affect the privacy of node operators.

Our Contributions. This work presents a systematic privacy analysis that seeks to better understand the traceability of *cross-layer* information by combining data from the Bitcoin blockchain and the LN. Our methodology is structured in two main strategies: (i) heuristics to create, on layer 1, clusters of Bitcoin entities controlled by the same actor, on layer 2, clusters of off-chain Lightning nodes; and (ii) heuristics to link these clusters across layers.

Within the first strategy, described in Section 4, we present four novel on-chain clustering heuristics (star, snake, collector, proxy), which group Bitcoin addresses based on their interaction patterns with the LN. With this heuristic, we are able to cluster 23% of all Bitcoin entities funding a Lightning channel, and 15% of all entities closing a channel. We also present a Lightning node clustering heuristic that leverages public announcements of aliases and IP addresses. We are able to group 589 nodes into 233 clusters.

Within the second strategy, described in Section 5, we present two novel linking heuristics. The first exploits the fact that the same Bitcoin entity can close one channel and then re-use the coins to open a new channel. In this case, we are able to link 19.44% of the nodes to 15.16% addresses in our dataset, when combined with the previous on- and off-chain clustering heuristics. The second heuristic exploits the reuse of a single Bitcoin entity for opening several channels to different nodes. This allows us to link 26.3% of the addresses to 43.7% of nodes.

We finally assess the impact of our deanonymization techniques on the privacy of Bitcoin entities as well as the security and privacy of the LN. First, when combining the results from the aforementioned two strategies, we observe that 17% of the Bitcoin addresses in our dataset can be attributed with information from the LN such as aliases or IP addresses associated to the linked Lightning nodes. Second, from the LN point of view, we observe that as few as five entities control 35% of the capacity in the LN. This centralization of the capacity has several implications in security and privacy. From the security point of view, a single entity with high capacity is at the position to block the capacity of over 40% of the channels in the LN, affecting over 15% of the capacity. Moreover, this centralization reduces the burden for targeted denial of service attacks, allowing high impact even for low budget attackers. From the privacy point of view, we observe that entities with high capacity are also highly connected in the LN so that can potentially stop about 25% of the payments as well as breach the privacy guarantees for payments routed through a number of paths between 5% and 40% depending on payment value.

To our knowledge, this paper is the first to present a method for linking Lightning nodes with Bitcoin addresses and to discuss privacy and security implications. For the reproducibility of the results, we make our data set and our implementation openly available at `https://github.com/MatteoRomiti/Lightning-Network-Deanonymization`.

2 Background and Problem Statement

In this section, we first present a simplified model of the Bitcoin blockchain and payment channels in the LN with the notions and notations relevant for this paper. Then we discuss and formalize the cross-layer linkage problem, as well as related work in this area. For further details on PCNs, we refer the reader to recent surveys [5, 8].

2.1 Bitcoin Blockchain

The Bitcoin blockchain is an immutable, append-only and publicly available ledger that contains the complete set of addresses and transactions from the Bitcoin genesis.

A Bitcoin **address** a is a tuple $a := (\mathsf{cash}, \mathsf{cond})$ where cash is a positive integer that denotes the number of coins (in Satoshis) associated to this address, and cond is an excerpt of the Bitcoin script language that denotes the (cryptographic) conditions under which a can be used in a transaction. Although in principle it is possible that cond encodes any condition that can be expressed in the Bitcoin script language, in practice most of the addresses share a few conditions: (i) $\mathsf{cond} := pk$ denoting that using this address requires a signature σ on the transaction verifiable under the public key pk; (ii) $\mathsf{cond} := \{pk_1, pk_2\}$ denoting that using this so-called multisig address requires two signatures $\{\sigma_1, \sigma_2\}$ verifiable with pk_1 and pk_2. We hereby say that an address a is owned by a user if she can fulfill the condition $a.\mathsf{cond}$ on her own.

A **Bitcoin wallet** is the software used by a Bitcoin user to handle Bitcoin addresses owned by her. Similar to the notion of saving and checking accounts in banking, a Bitcoin user is advised to have (at least) two wallets[1]: A *cold wallet* storing larger sums and a *hot wallet* with the addresses from where the user spends more frequently. A hot wallet is expected to hold a small number of coins and be periodically topped up from the cold wallet. This separation aims to reduce theft by hackers and malware possibly available in the computer or phone that holds the private keys required to spend addresses in the hot wallet.

A Bitcoin **transaction** tx is a tuple $tx :=$ (txid, Input, Output, $\mathsf{Witness}$) where txid is the identifier of the transaction calculated as the hash of the *body* of the transaction, i.e., $H(\mathsf{Input}, \mathsf{Output})$; Input denotes the set of addresses set as input and thus being spent in this transaction, Output is the set of addresses set as output; and $\mathsf{Witness}$ contains the witness of the transaction allowing to fulfill the conditions $a.\mathsf{cond}$ of each input address a.

Related work has shown that it is possible [16] and effective [6] to cluster Bitcoin addresses that belong to the same user. We capture that by defining a Bitcoin **entity** e as a set $e := \{a_i\}$ of addresses controlled by the same user as clustered with the co-spending heuristic.

[1] `https://en.bitcoin.it/wiki/Cold_storage`

2.2 Payment Channels in Lightning Network

A **node** n in the Lightning Network (LN) is a tuple $n := (\mathsf{nid}, \mathsf{IP}, \mathsf{Alias})$, where nid is the identifier of the node computed as the hexadecimal representation of a public key from a digital signature scheme; IP denotes the IP address associated to the node, and Alias the associated lexical label.

A **payment channel** c is then created between two nodes and denoted by the tuple $c := (\mathsf{chpoint}, n_1, n_2)$, where $\mathsf{chpoint}$ denotes the channel's endpoint that is set to the identifier $tx.\mathsf{txid}$ of the funding transaction tx that created the channel. As the transaction may have several outputs, $\mathsf{chpoint}$ also contains the output index of the multisig address that locks the funds in the channel (e.g., $\mathsf{chpoint{:}choutindex}$); while n_1 and n_2 are the nodes of the channel.

A **Lightning wallet** is the software used by a Lightning user to manage her node, as well as the channels of this node. In practice, a Lightning wallet comes with an integrated Bitcoin (hot) wallet to open and close channels in the LN. Recent releases of two Lightning wallet implementations (*lnd* and *c-lightning*) [2, 26] enable opening and closing a channel using an external Bitcoin wallet.

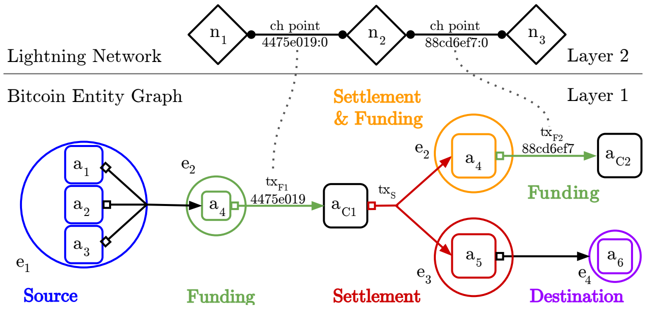

Figure 1: **Life cycle of a Lightning channel.** At layer 1, a source entity e_1 tops up entity e_2 that is then used in tx_{F1} as funding entity of the channel c_1 represented by multisig address a_{C1}. The channel c_1, referenced by a channel point at layer 2, is established between the nodes n_1 and n_2. The channel c_1 is then closed with the settlement transaction tx_S sending the funds back to two settlement entities, e_2 and e_3. The former, e_2, reuses these coins in tx_{F2} to fund another channel (c_2) between n_2 and n_3 represented at layer 1 by the multisig address a_{C2}. The coins in the other settlement entity, e_3, are instead collected into a destination entity e_4, not directly involved in the LN.

2.3 Cross-Layer Interaction

A payment channel enables multiple Bitcoin transactions between two users without adding each of them to the Bitcoin blockchain. The interaction between the blockchain and the Lightning Network is illustrated in Figure 1 and works as follows: assume that Alice has a cold Bitcoin wallet with coins in address a_1 and she wants to open a payment channel with Bob. Moreover, assume that Alice has a Lightning wallet that handles a_4. In this setting, the lifetime of the payment channel between Alice and Bob is divided into the following phases:

Replenishment. Alice first transfers coins from her entity e_1 (a_1, a_2, a_3) to her second entity e_2 (a_4), to top up the Lightning wallet from the Bitcoin cold wallet. We call e_1 the **source** entity as it is used as the source of funds to be later used in the LN.

Funding. Alice can now open a channel with Bob by first computing a *deposit* address a_{C1} such that $a_{C1}.\mathsf{cash} := x_1$ and $a_{C1}.\mathsf{cond} := \{pk'_A, pk_B\}$, where x_1 is fewer than the coins received at a_4 in the previous step and cond reflects that ownership of a_{C1} is shared between Alice and Bob. Second, Alice creates a **funding transaction** tx_{F1} where $tx_{F1}.\mathsf{Input} := a_4$,[2] $tx_{F1}.\mathsf{Output} := a_{C1}$, $tx_{F1}.\mathsf{txid} := H(tx_{F1}.\mathsf{Input}, tx_{F1}.\mathsf{Output})$ and $tx_{F1}.\mathsf{Witness} := \sigma_A$, where σ_A is a digital signature on $tx_{F1}.\mathsf{txid}$ verifiable under pk_A. After the transaction tx_{F1} appears on the Bitcoin blockchain, the payment channel c_1 between Alice and Bob is effectively open. The channel c_1 is then represented in the payment channel system as the tuple $(c_1.\mathsf{chpoint}, n_1, n_2)$, where n_1 and n_2 are (possibly new) nodes belonging to Alice and Bob.

Payment. After the channel c_1 is open, during the *payment* phase, both Alice and Bob can pay each other by exchanging authenticated transactions in a peer-to-peer manner authorizing the updates of the balance in the channel. Following our example, let's assume that Alice wants to use channel c_1 with Bob to pay $\alpha < x_1$ to his address. To do so, they create a **settlement transaction** tx_S where $tx_S.\mathsf{Input} := a_{C1}$, $tx_S.\mathsf{Output} := \{a_4, a_5\}$ so that a_4 belongs to Alice, a_5 belongs to Bob, $tx_S.tx := H(tx_S.\mathsf{Input}, tx_S.\mathsf{Output})$, and $tx_S.\mathsf{Witness} := \{\sigma_A, \sigma_B\}$, where both σ_A and σ_B are signatures on $tx_S.tx$ verifiable by the public keys included in $a_{C1}.\mathsf{cond}$.

The cornerstone of payment channels is that Alice and Bob do not publish the settlement transaction tx_S in the Bitcoin blockchain. Instead, they keep it in their memory (i.e., off-chain) and locally update the balances in their channel c_1. Both Alice and Bob can repeat this process several times to pay each other.

Settlement. When the channel is no longer needed, Alice and Bob can close the channel by submitting the last agreed settlement transaction into the Bitcoin blockchain, thereby unlocking the coins from a_{C1} into two Bitcoin addresses, each belonging to one of them with a number of coins equal to the last balance they agreed off-chain. In practice, the settlement transaction may have more than two outputs: Alice can pay

[2] Although theoretically a payment channel can be dual-funded (i.e., Bob also contributes x_1 to the funding transaction), this feature is under discussion in the community [1] and currently only single-funded channels are implemented in practice.

Bob to a third address where Bob needs to provide cond as input data other than a signature to redeem the coins (e.g., the valid preimage of a hash value before a certain timeout as defined in the Hash-Time Lock Contract[3]).

Collection. After the settlement transaction appears in the Bitcoin blockchain, Bob gets the coins in his Lightning wallet. As a final step, Bob might want to get his coins into a different Bitcoin wallet of his own (e.g., cold wallet). For that, Bob performs a transaction that transfers funds from a_5 to a_6, which we call **destination** address.

We note several points here. First, the addresses involved in the lifetime of payment channels could have been clustered into entities. In such case, we refer to the source/funding/settlement/destination entity involved in the steps instead of the particular address itself. In our example, Alice owns entity e_1 that controls (among others) a_1 and we thus say that entity e_1 is the *source* entity in the replenishment step. Second, the same entity can be used at the same time for settlement and funding. Finally, Alice gets the coins from the channel with Bob in entity e_2 that is then reused later to open a new payment channel.

2.4 The Cross-Layer Linking Problem

A starting point, as shown in Figure 1, is to identify the funding transaction tx_{F1} corresponding to the payment channel $c_1 := (\mathsf{chpoint}, n_1, n_2)$, by finding the transaction (and the output index) that fulfills the condition $tx_{F1}.\mathsf{txid} = c_1.\mathsf{chpoint}$. But we cannot assert that the entity e_2 in $tx_{F1}.\mathsf{Input}$ also controls n_1, as it could also be that e_2 controls n_2. Similarly, while we can deterministically get the settlement transaction tx_S used to close the channel c_1, we cannot unambiguously link each settlement entity to the corresponding node.

The goal of this work is two-fold: (i) cluster Lightning nodes on the LN (layer 2) as well as Bitcoin entities (layer 1); and (ii) unambiguously link sets of Lightning nodes to the sets of Bitcoin entities that control them. Technically, this corresponds to finding a function that takes a set of Lightning channels as input and returns tuples of the form (entity, node) for which it can be asserted that the Lightning node is controlled by the Bitcoin entity linked to it. We state our problem a bit more formally in Definition 1.

Definition 1 (Cross-Layer Linking Algorithm) *Let $\mathcal{C}$ be the set of channels in the LN. Let $\mathcal{E}$ be the set of Bitcoin entities and let $\mathcal{N}$ be the set of nodes in the LN. Let $\mathcal{P}(\mathcal{C})$ be the power set of $\mathcal{C}$, let $\mathcal{P}(\mathcal{E})$ be the power set of $\mathcal{E}$ and let $\mathcal{P}(\mathcal{E} \times \mathcal{N})$ be the power set of the cross product $\mathcal{E} \times \mathcal{N}$. A* linking algorithm *is an implementation of a function $\mathcal{P}(\mathcal{C}) \times \mathcal{P}(\mathcal{E}) \rightarrow \mathcal{P}(\mathcal{E} \times \mathcal{N})$. Each tuple $(e, n) \in \mathcal{P}(\mathcal{E} \times \mathcal{N})$ denotes that the entity e controls the node n.*

2.5 Related Work

On-Chain Clustering and Linking. Several address clustering heuristics have been proposed for Bitcoin, the safest, most effective, and most studied one being the previously mentioned co-spending heuristic [16,21], also known as multiple-input or common-input-ownership heuristic. It assumes that if two addresses (i.e. a_1 and a_2) are used as inputs in the same transaction while one of these addresses along with another address (i.e. a_2 and a_3) are used as inputs in another transaction, then the three addresses (a_1, a_2, a_3) are likely controlled by the same actor.

Recent studies investigated linking in privacy-centric currencies such as Monero [18] or Zcash [10], in account-model currencies such as Ethereum [25] and also in the Ripple network [17]. However, all these methods rely on on-chain transactional evidence and do not take into account off-chain data.

Off-Chain Payment Channel Analysis. Single-layer security attacks on the LN topology were the focus of many recent studies: Rohrer et al. [23] measured the LN topology and found that the LN is highly centralized. An attacker who can remove a certain number of nodes (e.g., through a DoS attack) should focus on nodes the with highest centrality, an attacker with limited resources should target the highest-ranked minimum cut set, which is the set of edges with minimal accumulated capacity that, when removed, partitions the graph. Similarly, Seres et al. [24] found that the LN provides topological stability under random failures, but is structurally weak against rational adversaries targeting network hubs. Also, Martinazzi and Flori [15] have shown that the LN is resilient against random attacks, but very exposed to targeted attacks, e.g., against central players. Lin et al. [7] inspected the resilience of the LN and showed that removing hubs leads to the collapse of the network into many components, an evidence suggesting that this network may be a target for the so-called split attacks.

Single-layer privacy in the LN has recently been studied by Kappos et al. [11], who focused on balance discovery and showed that an attacker running an active attack can easily infer the balance by running nodes and sending forged payments to target nodes. When combined with network snapshots, an attacker can discover all balances in the network.

Nowostawski and Ton [19] conducted an initial cross-layer analysis and investigated footprints of the LN on the public Bitcoin blockchain. They linked channels with their corresponding funding transactions. Our work instead investigates the link between Lightning nodes and Bitcoin entities and the associated security and privacy implications.

In summary, we can state that most existing studies consider active, single-layer attacks on either the LN or the blockchain, however, no study has yet investigated the linkage of Bitcoin entities with Lightning nodes based on publicly available information, which is the focus of this paper.

[3] https://en.bitcoin.it/wiki/Hash_Time_Locked_Contracts

3 Dataset

For our analysis we rely on off-chain (LN) as well as on-chain data (Bitcoin blockchain).

3.1 Off-chain Data: LN

We used the LN Daemon (LND) software and captured a copy of the LN topology at regular intervals (30 min) via the *describegraph* command since May 21 2019. The off-chain part of our dataset contains $70,783$ channels, $37,280$ of which were still open on March 3, 2020. The most recent channel in our dataset was opened on February 24, 2020, while the oldest was opened on January 12, 2018. We stored each channel between two nodes as a tuple $(\mathsf{chpoint}, n_1, n_2, \mathsf{cap})$, where chpoint denotes the channel identifier, and the pair n_1, n_2 denotes the nodes sharing the channel, as described in Section 2. Additionally, we store the capacity of the channel in cap. Our dataset does not contain the balance of each node because this information is kept private at the nodes themselves. We also define the *activity period* of a node as the time that starts with the funding transaction that opened the first public channel in which the node appeared and ends either with the settlement transaction of its last public channel or with 2020-05-18 (the time of preparing the dataset), if the nodes had still public channels open.

These channels were established between $8,267$ distinct nodes, each of which we stored as a tuple $(\mathsf{nid}, \mathsf{IP}, \mathsf{Alias})$, where nid denotes a node identifier as described in Section 2. If available, IP refers to the publicly announced IP or Tor address of a node, and Alias to the human-readable alias assigned by the node operator.

3.2 On-chain Data: Bitcoin Blockchain

First, for each channel in our off-chain dataset, we used the transaction hash included in the channel's field chpoint for retrieving the *funding transaction*. Next, we checked whether the coins sent to the multisig address were spent or not. If a coin was spent, we fetched the *settlement transaction*, that uses that multisig address as input. We obtained this data by querying the GraphSense API[4] and the Blockstream API[5]. We thereby extracted $70,782$ funding transactions[6] and $33,503$ settlement transactions.

Next, we extracted the input addresses of all funding transactions and the output addresses of all settlement transactions and mapped them to funding and settlement entities, as defined in Section 2.1. Before clustering entities, we used BlockSci [9] to filter CoinJoin transactions because they would merge addresses of unrelated users. In addition, we made sure that no CoinJoins from Wasabi or Samourai[7] wallets were in our dataset. On the funding side, we also extracted the *source entities* that were sending coins to funding entities; on the settlement side, we retrieved *destination entities* that received coins from settlement entities. For that purpose, we implemented a dedicated data extraction and analytics job for the GraphSense Platform and executed it on a snapshot of the Bitcoin blockchain up to block $618,857$ (2020-02-25 00:59), amounting for a total of $506,395,375$ transactions and $616,401,800$ addresses clustered into $298,406,250$ entities.

Table 1: On-chain Dataset Summary.

	Source	Funding	Settlement	Destination
# Addr		117,800	46,782	
# Entities	130,828	69,501	28,088	25,014
# Addr (Exp.)	57,245,617	128,421	80,795	63,742,876
# Services	2,515	2	32	251
# Relations		134,477	31,619	

After having extracted the Bitcoin entities that were involved in opening and closing payment channels, we attributed them using the Chainalysis API[8] and assigned service categories (e.g., exchange, hosted wallet) to entities.

Table 1 summarizes the number of addresses (*# Addr*) found in funding and settlement transactions as well as the number of resulting entities after applying the co-spending heuristics on these addresses (*# Entities*). We can clearly observe that the number of distinct source entities (130,828) is higher than the number of destination entities (25,014), which is also reflected in the number of relations (*# Relations*) representing monetary flows from source to funding entities and from settlement to destination entities, respectively. We think these unbalanced numbers are due to many channels still open.

Since the co-spending heuristic also groups addresses which were not part of our dataset snapshot, we also added the number of expanded addresses (*# Addr (Exp.)*). The difference between the number of addresses and entities on both the source and destination side can be explained by the presence of super-clusters, which are responsible for large transaction inputs and outputs and typically represent service entities such as cryptocurrency exchanges [6].

Finally, this table also lists the number of identified service entities (*# Services*). This is generally low ($< 2\%$), reflecting that non-custodial, possibly hosted wallets (we identified 34) are used when opening and closing channels. Roughly 2% of all source entities were categorized, with the majority (1,8%) being exchanges. On the settlement side, we identified 1% of all destination entities as wallets being controlled by services, again the majority (0,6%) being exchanges. We also briefly examined one-hop-out data from the sources and destinations

[4] https://api.graphsense.info/

[5] https://github.com/Blockstream/esplora/blob/master/API.md

[6] Two channels were opened with one funding transaction.

[7] https://github.com/nopara73/WasabiVsSamourai

[8] https://www.chainalysis.com/

and found a similar picture: a fairly large fraction of entities is uncategorized, with most of the others being exchanges.

4 Single-Layer Clustering

In this section, we present clustering heuristics that can group Lightning nodes based on public announcements of aliases and IP addresses (Section 4.1) and Bitcoin entities based on their interaction with the LN (Section 4.2). A challenge, which has also been pointed out in previous works on clustering heuristics (see Section 2.5), lies in the lack of ground-truth data available for quantifying their effectiveness. Acknowledging this issue, we discuss each heuristic in terms of possible false positives and identify countermeasures.

4.1 Off-Chain Lightning Node Clustering

The operator of a Lightning node can announce custom node features such as an alias, which was added to the LN to improve the usability of the system. The alias can be changed by the operator at any time without affecting the operation of open channels, as those are only tied to a node's private and public key pair. We observed that when a user is operating multiple nodes, it is likely that she will name her nodes in a similar fashion, or along a common theme. For example, the operator LNBIG.com enumerates its nodes on their website[9], with aliases such as *LNBIG.com [lnd-25]*, *LNBIG.com [lnd-34]*. Via public chat, the developers confirmed that *LNBIG.com Billing* also belongs to them. Strong alias similarities are most likely intentional, for example, to make it easier for users to identify a service, or the operators may want to use it to achieve a reputation or branding effect.

In order to find nodes under the control of the same entity, we can exploit the alias information and measure similarity. We evaluated popular string similarity metrics (cf. [4]) such as the Levenshtein, Hamming and Jaro-Winkler distances. Naturally, however, aliases can be similar, but not belong to the same entity. Examples include node aliases such as *WilderLightning* and *GopherLightning*, which overlap textually but are not controlled by the same entity.

Apart from the alias, nodes advertise their IP address (or an address within the Tor network) and a port. We can use this additional public information to filter the clusters obtained through alias similarity, increasing the confidence that the nodes are operated by the same entity.

Each IP address is part of a Classless Inter-Domain Routing (CIDR) [3] prefix that is under the control of one or multiple network operators. An Internet Service Provider (ISP) may operate a collection of such CIDRs, and their grouping is called autonomous system (AS), each of which is identified by an autonomous system number (ASN). By performing WHOIS queries, we can obtain the ASN's of each Lightning node IP address. If an alias-based node cluster consists only of IP addresses associated to a single ASN, we conclude that the Lightning nodes are hosted by the same network operator and is, therefore, more likely operated by a single entity. In addition, we also cluster Lightning nodes that are (or have been in the past) reachable via the same IP or Tor address.

Technically, we first determine *pairwise alias distances* by computing a distance matrix between all lightning node aliases using different distance metrics. Then we perform *agglomerative hierarchical clustering* to avoid early cluster merging due to single aliases being similar to two distinct clusters. For *threshold identification*, we evaluate the full range of thresholds by counting the number of lightning nodes that remain when pruning clusters that are not pure with respect to their ASNs. We then choose the threshold that results in the largest number of clustered nodes, while ensuring the LNBIG.com cluster is identified as a single cluster of at least 26 nodes, as we have ground truth from their website. In parallel, we perform *IP-based clustering* by grouping all lightning nodes that have been seen to be reachable via the same IP or Tor address. Finally, we *join alias and IP-based clustering* and merge the resulting alias and IP-based clusters if there is an overlap. This results in the final off-chain-based lightning node clusters.

In our analysis, we considered all nodes with their history of aliases and valid addresses. IPs within address ranges reserved for special purposes such as private networks are excluded. We compared the performance of ten different string distance measures (see Appendix B) and concluded that the relative longest common substring measure yields the best results. In particular, 314 lightning nodes have been grouped into 115 clusters. The IP-based clustering yields 466 clustered lightning nodes, 185 of which are already part of the alias-based clusters. By merging these clusters, the final cluster count is 233, with a total of 589 lightning nodes clustered. The two largest clusters are **lnd-gar-nodl-it* (42 nodes) and *LNBIG.com* (26 nodes).

Discussion. Alias/ASN and IP-based clustering can yield some false positives. For example, if two nodes have very similar aliases, and are coincidentally hosted on the same AS, they would be recognized as one entity. This could happen with LN-specific hosting services or widespread services such as Amazon. In general, however, filtering alias clusters to those running on the same AS should result in fewer false positives. For the entity *LNBig.com*, we had ground truth which we used to optimize the alias similarity threshold. By reaching out to one operator, we were able to validate one cluster of lightning nodes. For privacy reasons, we refrain from naming the operator.

Countermeasures. While the use of aliases supports the usability of the system, the way some users choose them clearly hinders their privacy. For more privacy, aliases should be sufficiently different from one another. While the public announcement of IP addresses may be unavoidable for

[9] `https://lnbig.com/#/our-nodes`

those nodes that wish to have incoming channels in the LN, linkability across nodes of the same user can be mitigated if the clients for each node are hosted with different service providers (and thus ASNs and IP addresses).

4.2 On-Chain Bitcoin Entity Clustering

LN-blockchain interactions are reflected in the Bitcoin entity graph (see Section 2): opening a channel causes a monetary flow (relation) from a *source entity* to a *funding entity*; closing a channel causes a flow from a *settlement entity* to a *destination entity*. When inspecting the resulting graph abstractions, we observed four patterns (see Appendix A) that inform our heuristics.

First, several funding entities received funds from the same source entities. Looking at the interaction graph among entities, this forms a *star-shaped pattern* with one source entity transferring coins to several funding entities. This reflects a current requirement in the Lightning wallet[10], which requires users to transfer funds from an external wallet (source entity) to an internal wallet (funding entity) before opening a channel. Under the assumption that the source entities do not represent services like an exchange, which occurs less than 1% (see Section 3), we can assume that the receiving funding entities in this pattern are likely controlled by the same user who funded several channels using the same external wallet.

Second, again on the funding side, we observed a *snake-like pattern* in which source entities transfer coins to a funding entity, which then opens a channel and the change from the funding transaction is used to fund another channel, and so on (analogous to the well-known Bitcoin Change-Heuristic [16]). We assume that the funding entities in this pattern are likely controlled by the same user.

Third, we identified a so-called *collector pattern*, which mirrors the previously described star pattern on the settlement side: a user forwards funds from several settlement entities, which hold the unlocked coins of closed channels in an internal wallet, to the same *destination entity*, which serves as an external *collector* wallet of funds and therefore fulfills a convenience function for the user.

Fourth, we also found a refined version of the collector pattern, which we call *proxy pattern*: a user first aggregates funds from several settlement transactions in a single settlement entity and then forwards them to a single destination entity. Again, we can assume that all settlement entities and the proxy entities are controlled by the same user.

Based on these observations we can define four heuristics, which are computed as follows: first, we construct 1-hop ego-networks for the funding and settlement entities. We thereby extract funding relations connecting source and funding entities, as well as settlement relations connecting settlement and destination entities (see Section 3). In these ego-networks, the entities are represented as nodes and monetary flows as edges between entities.

Table 2: On-chain clustering results.

	Star (F)	Snake (F)	Collector (S)	Proxy (S)
# Components	32 (<1%)	4,910 (32%)	875 (15%)	539 (9%)
# Entities	84 (<1%)	16,051 (23%)	2,289 (8%)	1,734 (6%)
# Addresses	87 (<1%)	16,077 (13%)	3,839 (5%)	7,657 (9%)

Next, we compute all strongly-connected components in these graphs and filter them by the following conditions:

- *Star Heuristic (Funding)*: if a component contains one source entity that forwards funds to one or more funding entities, then these funding entities are likely controlled by the same user.
- *Snake Heuristic (Funding)*: if a component contains one source entity that forwards funds to one or more entities, which themselves are used as source and funding entities, then all these entities are likely controlled by the same user.
- *Collector Heuristic (Settlement)*: if a component contains one destination entity that receives funds from one or more settlement entities, then these settlement entities are likely controlled by the same user.
- *Proxy Heuristic (Settlement)*: if a component contains one destination entity that receives funds from one or more entities, which themselves are used as settlement and destination entities, then these entities are likely controlled by the same user.

Table 2 shows the number of Bitcoin entities we were able to cluster with each heuristic. When regarding the connected components, we can clearly see the rare occurrence of the star patterns and the dominance of the snake pattern, which represents 32% of all funding components. On the settlement side, 24% of all components either match the collector or the proxy pattern. Consequently, we were able to group 23% (16,135) of all funding entities and 14% (4,023) of all settlement entities. This corresponds to 16,164 funding addresses and 11,496 settlement addresses.

Discussion. Our heuristic can, by definition, also yield false positives for two main reasons: first, an entity could represent several users if clustered addresses are controlled by a service (e.g., exchange) on behalf of their users (custodial wallet) or if transactions of several unrelated users are combined in a CoinJoin transaction. Second, users could transfer ownership of Bitcoin wallets off-chain, e.g., by passing a paper wallet. While the second case is hard to filter automatically, we applied countermeasures to the first case: first, we filtered known CoinJoin transactions (see Section 3), and second, we filtered all components containing service entities by using one of the most comprehensive attribution dataset available.

Countermeasures. We suspect that the above patterns reflect a user behavior that is already known to compromise the privacy of transactions: reuse of addresses. If outputs of

[10] We note that this requirement may no longer be there if the functionality available in the recent release [26] is widely adopted among Lightning users

funding transactions are not reused for opening other channels, the snake heuristic would not work; if users refrain from funding channels from a single external source and avoid collecting funds in a single external destination entity, the other heuristics would not yield any significant results.

5 Cross-Layer Linking: Lightning Nodes and Bitcoin Entities

In this section, we present two algorithms that link Lightning nodes to the Bitcoin entities that control them. In both of these heuristics, we do not consider settlement transactions with more than two output entities (1.7% of the settlement transactions), as they are not a cooperative close and do not allow us to unambiguously link nodes and output entities. An assumption that we make in both of the following linking algorithms is that if one node in a channel has been linked to a settlement entity and the settlement transaction has two output entities, then the other node can be linked to the other settlement entity.

5.1 Linking Algorithm 1: Coin Reuse

Although a payment channel may remain open during the whole lifetime of the LN, payment channels are often closed in practice so that the two users of the closed channel get back their coins in their corresponding Bitcoin addresses. Among other reasons, the users may close a channel because it is depleted (i.e., all the balance in the channel is with one user) or simply because they no longer find economic incentive in keeping their coins locked at the channel to transact between each other. After a channel is closed and the two users get their corresponding coins back, they can open new payment channels with other users that better represent their updated business relations.

Our linking algorithm builds upon this usage pattern. An illustrative example of this linking algorithm is included in Figure 1 where a funding entity e_2 has been used to open a channel c_1 between nodes n_1 and n_2 with the funding transaction tx_{F1}. Later, this channel has been closed in the settlement transaction tx_S, releasing the coins in the channel to the entities e_2 (i.e., the same that was used as input in tx_{F1}) and e_3. Finally, assume that the owner of entity e_2 decides to open a new channel reusing the coins from tx_S performing a new funding transaction tx_{F2} which results in the payment channel c_2 between the aforementioned node n_2 and the node n_3.

In this situation, given that the entity e_2 has appeared in the settlement transaction of c_1 and has been reused to open a new channel in the funding transaction c_2, our heuristic concludes that the entity e_1 controls node n_1.

Definition 2 (Linking Algorithm: Coin Reuse) *Assume that a Bitcoin entity e opens a Lightning channel* $c_1 := (\mathsf{chpoint}_1, n_1, n_2)$. *If e is used as settlement entity to close the Lightning channel* c_1 *and also as funding entity to open a new Lightning channel* $c_2 := (\mathsf{chpoint}_2, n_1, n_3)$, *the user controlling entity e also controls the Lightning node* n_1 *in common to both channels* c_1 *and* c_2.

We applied the linking algorithm based on coin reuse which resulted in 43 tuples of (funding transaction, settlement transaction, settlement transaction) and 8 entities reusing their addresses for opening and closing channels. Once these 8 entities are linked to Lightning nodes, all the other output entities in the settlement transactions of these 8 entities can be linked to the counter-party nodes in the channels as mentioned earlier. Finally, after these new links are created, our heuristic can iteratively go over the settlement transactions that involve these newly linked entities to find other entity-node pairs.

After 7 iterations, the heuristic yielded $4,218$ entities linked to $1,265$ nodes, having thus cases where a node is linked to multiple entities. In total, if we consider the number of entities we have in our dataset ($92,544$ overall, both funding and settlement side[11]) the heuristic is able to link 4.56% of them. This result is a lower bound on the possible number of linked entity-node pairs because the linking algorithm mainly relies on channels to be closed (in our dataset only half of them are) and on a specific subset of entities, namely the output entities of settlement transactions with exactly two outputs, one per node. In fact, if we focus only on settlement transactions with two output entities, we have $15,895$ entities, 26.54% of which can be linked, showing thereby that this linking algorithm has a targeted but effective linking effect. Regarding the nodes percentages, we can link 15.30% of the total ($8,267$ overall) and 42.84% of the nodes for which it has existed at least one channel that has been closed using a 2-output-entity settlement transaction, confirming thus the trend we observed with entities.

Discussion. We note that requiring that the same entity is used for all three transactions (i.e., funding and settlement of first channel as well as funding of the second channel) may be too restrictive and leave out further links of entities and nodes. However, we enforce this restriction to avoid false positives that could be otherwise introduced as we describe next.

Assume we control a LN node, n_2, with an associated Bitcoin entity e_1 that funds channel c_1 between node n_2 and n_1 through tx_{F1}. Furthermore, we have a LN wallet with an associated Bitcoin entity, e_3, on our phone provided by a third-party app. This means that there must be another node in the LN, n_3, managed by this third-party app. When we decide to close channel c_1, we specify an address provided by our third-party app, hence belonging to entity e_3, as settlement address to receive the funds back. We finally proceed to use these funds to open a new channel, c_2, again with node n_1 but from node n_3, the third-party node. Without the requirement on the same funding entity, the heuristic would link the

[11] Here we do not consider source and destination entities as they do not directly interact with the LN.

node n_1, in common between the two channels, to the entity e_3 reusing the funds, which is false. With the same funding address requirement, instead, this case is ignored.

A further condition that needs to be satisfied to strengthen this heuristic is that the nodes not common to the two channels (nodes n_1 and n_3 in Figure 1) have a time overlap in their activity period. This excludes the unlikely, but not impossible case that one node changes its ID (public key) from n_2 to n_3 keeping the same Bitcoin wallet (and thus entity), which could allow one to open two channels from two different nodes, but to the same node, using the same Bitcoin entity, creating a false-positive case for the heuristic.

Countermeasures. The default functionality of Lightning wallets followed thus far by virtually all users consists in having a single wallet per node from where to extract the funds to open channels and where to send the coins after channels connected to such node are closed. We conjecture that this setting favors the usage pattern leveraged in the linking algorithm described in this section. As a countermeasure, we advocate for the support of funding and settlement channels of a single node from different (external) Bitcoin wallets, helping thus to diversify the source of funds. We observe that recent versions of the Lightning wallet *lnd* and *c-lightning* have started to support this functionality [2, 26].

5.2 Linking Algorithm 2: Entity Reuse

Assume that a user, say Alice, has a certain number of coins in her wallet and she wants to participate in the LN. For that, she needs to open payment channels with other users by creating the corresponding funding transactions. In practice, Alice might have used the same Bitcoin wallet to open all the channels, which translates in using the same Bitcoin entity as input for all funding transactions. We leverage this usage pattern in our second linking algorithm.

In a bit more detail, in this linking algorithm we assume that an entity e opened several payment channels with other entities. This common usage pattern in practice can be detected at the blockchain by finding the set of N_C funding transactions that have e in common as the funding entity. We can thus say that e has opened N_C channels. At the LN, if there is only one node n common to all the N_C channels funded by e, we say that e controls n.

An illustrative example of this linking algorithm is shown in Figure 2, where entity e_1 funds $N_C := 3$ channels and a node n_1 is common to all those channels. Then we can say that entity e_1 controls n_1.

Definition 3 (Linking Algorithm 2) *If there are N_C channels opened by one single funding entity e that have only one Lightning node n in common, then the user controlling entity e also controls the node n.*

The minimum value of N_c that allows us to link entities to nodes is 2. We can link 4,747 entities to 1,541 nodes which correspond to 5.13% of all the entities and 18.64% of all the nodes respectively.

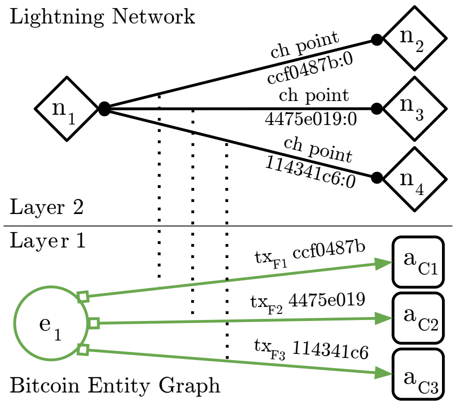

Figure 2: **Linking Algorithm 2: Entity reuse example.** At layer 1, the funding entity e_1 is reused to perform $N_C = 3$ funding transactions. At layer 2, the corresponding channels are opened and there is one node, n_1, common to all the N_C channels and it can be linked to the funding entity e_1.

Discussion. The way this linking algorithm has been described and implemented so far might yield false entity-node links. As discussed in section 5.1, a user can open a channel from its node n_2 to another node n_1, then close the channel, change its node ID to n_3 keeping the same Bitcoin wallet and finally open a second channel to n_1. For this linking algorithm, this example would cause a false positive because n_1 would be linked to the Bitcoin entity of this user. To prevent this from happening, we add the following condition. Consider the set of nodes appearing in the channels funded by a single funding entity e and exclude from this set the node that has been linked to e with this heuristic. Now, if there is at least one pair of nodes (n_2, n_3 from the example above) in this set that have an activity period overlap, then we discard the false-positive risk as it is not possible for node n_2 to change to n_3 keeping two channels open. When implementing this additional requirement, we discovered that our results do not contain any false positive as there is at least one pair of nodes with an activity period overlap for each entity-node link.

To further validate the results of this second linking algorithm, we checked whether it provides the same entity-node links as in the linking algorithm presented in Section 5.1. To do this, we verified that all of the 4,218 entities that appear in both linking algorithms, are associated to the same nodes.

Countermeasures. A countermeasure to this heuristic is to not reuse the same funding entity to open multiple channels. This can be achieved either by having multiple unclustered addresses in a wallet or to rely on external wallets [2, 26].

Table 3: Summary results

Linking + Clustering	% addresses linked	% entities linked	% nodes linked
heuristic 1	13.46	4.56	15.30
heuristic 1 + stars	13.46	4.56	15.30
heuristic 1 + snakes	13.48	4.56	15.30
heuristic 1 + proxies	15.14	5.26	17.13
heuristic 1 + all on-chain	15.16	5.26	19.44
heuristic 1 + all on/off-chain	15.16	5.26	17.13
heuristic 2	14.41	5.13	18.64
heuristic 2 + stars	14.45	5.15	18.75
heuristic 2 + snakes	24.55	10.32	39.03
heuristic 2 + proxies	16.09	5.83	20.33
heuristic 2 + all on-chain	26.3	11.04	40.16
heuristic 2 + all on/off-chain	26.3	11.04	43.7

6 Combining Heuristics

In this section, we merge the results of our clustering algorithms (as described in Section 4) and our linking algorithms (as described in Section 5), thereby increasing the linking between entities and nodes. In the best case (last entry in Table 3) we get to 26.3% of linked addresses and 43.7% of linked Lightning Network nodes.

The reason why on-chain clustering algorithms should improve the linking algorithms is that they better represent a user's behavior, just like the co-spend heuristic does. If we think about the on-chain patterns that we introduced, they all group together entities that, based on their interaction with the LN, are controlled by one single actor.

Table 3 shows the percentage of addresses, entities and nodes that can be linked together when adding the clustering algorithms in the linking process. Comparing these results with the ones from the basic implementation of the linking algorithms, we notice that the first linking algorithm improves only by few percentage points, while the second linking algorithm improves roughly by a factor of 2. However, not every clustering algorithm contributes the same to the overall results. We discuss each of them next except for the collector pattern, which, despite being a useful on-chain clustering heuristic, does not contribute significantly to the linking.

Star-pattern contribution. The behavior that can be modeled when combining the star pattern and the linking algorithms can be described with the following example. A user owns a cold wallet and additionally controls one LN node n which runs its own LN wallet. Anytime the LN wallet needs to be replenished, it generates a different address a_i (corresponding to an entity e_i of size 1) and the cold wallet sends coins to it. After this, e_i can be used to open a new channel from the node n. At this point, the node n can be linked to the star that is formed by the set of entities $\{e_i\}$.

Unfortunately, this pattern, as reported in Table 2, occurs less often than the others, a possible reason why it has no contribution for the linking algorithm 1 and an impact of less than a percentage point in linking algorithm 2.

Snake-pattern contribution. As already described in Section 4.2 the snake pattern follows the concept of reusing the change address to fund a new channel. Due to the frequent creation of a change in Bitcoin, this pattern occurs much more often than the star pattern and the proxy pattern (two and one order of magnitudes more respectively). This also the reason why its contribution to the linking is the most significant one for linking algorithm 2. Unfortunately, it is not so effective with the linking algorithm 1, probably because the coin-reuse heuristic is a stricter version of the entity-reuse heuristic.

Proxy-pattern contribution. The proxy pattern models the behavior of a LN user that decides to merge the coins from different settlement transactions into one single entity to avoid keeping track of funds, possibly on different wallets. This pattern seems to have a stable contribution (around 2% for linked nodes) for both linking algorithms when applied without the other patterns.

Off-chain node clustering contribution. Assume there is a cluster of nodes obtained with the heuristic presented in Section 4.1 and one of these nodes has been linked to one entity. At this point, since the nodes in the cluster are supposed to be controlled by the same LN user, we can indirectly link all the other nodes in the cluster to the entity. We refer to these nodes as *indirectly-linked nodes*. Even though we enforced strict conditions in the clustering algorithm based on alias/IP information, we are aware of the fact that this type of linking may be considered as weaker as it relies on one additional assumption (nodes in an alias-based cluster are correctly attributed to one actor). In total, for the linking algorithm 1 we find 191 indirectly-linked nodes, which corresponds to an additional 2.91% of nodes linked, while for the linking algorithm 2 we find 292 indirectly-linked nodes which correspond to an additional 3.53% of nodes linked.

7 Security and Privacy Implications

In this section, we evaluate the security and privacy implications of our clustering and linking algorithms in two ways: (i) privacy impact on Bitcoin entities (Section 7.1) and (ii) security and privacy impact on the LN (Sections 7.2 to 7.6).

7.1 Privacy Impact on Bitcoin Entities

The linking algorithms and clustering algorithms described in this work allow attributing activity to Bitcoin entities derived from their interaction with the LN. Assuming a cluster is formed by a certain number of Bitcoin entities and Lightning nodes. If any of the Lightning nodes has publicly identifiable information (e.g., alias or IP address), this information can be attributed to the Bitcoin entities as well. In total, we can attribute tagging information (aliases or IP addresses) to 10,064 different entities that in total account for 26,343 different addresses, which represent 16.9% of our dataset.

This deanonymization is based purely on publicly available data and can be carried out by a low budget, passive adversary that simply downloads the Bitcoin blockchain and the information from the LN. Moreover, the possible deanonymization of Bitcoin entities hereby presented shows that it is crucial to consider the privacy of both layers *simultaneously* instead of one of them at a time as largely done so far in the literature.

7.2 Distribution of Wealth and Impact in Griefing Attacks in LN

In this section, we first evaluate how wealth is distributed in the LN, that is, how much capacity is controlled by each of the entities found during our analysis. For that, we take a recent snapshot of the LN from 2020-02-24 and extract the capacity controlled by each entity. If a channel has been created by an entity that has been linked to a node, we can attribute the full capacity of the channel to that node. For channels where only one side belongs to a clustered entity, we attribute the capacity to that entity. For all other channels, we assume that each entity controls half of the capacity. Under these assumptions, we observe that the overall capacity of the LN is distributed as shown in Table 4. In particular, a single entity controls over 36% of the overall capacity in the LN and as few as 156 nodes (2.4%) control over 80% of the capacity. This result refines the previous study in [7] where they find that 80% of the capacity is controlled by 10% of the nodes.

This result shows that few Lightning entities are in a privileged situation that they can potentially use to selectively prevent other Lightning nodes from transacting in the network, for instance, launching a *griefing attack* [22] against the victim nodes. In the griefing attack, the attacker finds a path of the form $n_1 \to n_2 \to \ldots \to n_k$ where n_1 and n_k belong to the attacker. Using that path, the attacker routes a payment from n_1 to n_k, thereby allocating funds at each channel to support the payment transfer. However, this payment is never accepted by n_k, forcing the intermediary channels to wait to release the funds locked for the payment until a certain timeout expires. In the current LN implementation, this timeout is in the order of several days.

For this attack to be effective, the attacker needs to perform and lock a payment for an amount at least the capacity available at the channel of the victim. However, as shown in Table 4, the uneven distribution of wealth in the LN makes this a small investment if the attacker is one of the entities with high capacity. In fact, we evaluated the possible damage that each entity in the LN can infringe by launching this griefing attack with the results shown in Figure 3. As expected from the wealth distribution, the entity with the highest capacity is the one that can infringe the most devastating attack, being able to render useless for a period of time over 40% of the channels in the Lightning Network, which amount for over 15% of the total capacity.

Table 4: Lightning entities controlling most capacity

Entity	Node count	Share of total capacity contributed
LNBig.com *	26	36.35%
BitRefill.com, ...	2	3.90%
CoinGate	2	2.06%
rompert.com	2	0.91%
Breez	2	0.84%

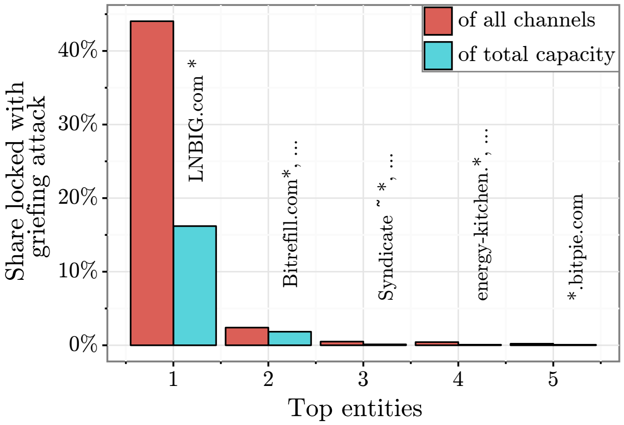

Figure 3: Fraction of all LN channels/capacity vulnerable to griefing attack launched by each entity.

7.3 Vulnerability to DoS attacks in LN

The growing monetary value of the LN and the existence of competitor business within the network as well as from other available payment networks open the door for DoS attacks. In fact, there have already been DoS attacks against the LN reported. For instance, in March 2018, it was reportedly hit by a distributed DoS attack that took 20% of the nodes offline[12]. In this state of affairs, we study here the effect of DoS attacks targeted at the Lightning entities found in this work.

Based on the LN snapshot we iteratively remove the nodes and channels corresponding to a given entity, starting with the entities that control the most capacity. We then compare the resulting graph with the original one to evaluate the *adversary's advantage* (i.e., attack's success) attributed to a DoS targeted to such entity. Following [23], we characterize the notion of adversary's advantage as $\Delta_m := \left|1 - \frac{m'}{m}\right|$ where m is the a priori measurement and m' is the a posterior one. The higher Δ_m becomes, the higher the success of the attack according to the metric m. We consider the three metrics as defined in [23]: (i) Δ_r defined as the number of nodes within the biggest component in the graph, representing thereby the

[12] https://www.trustnodes.com/2018/03/21/lightning-network-ddos-sends-20-nodes

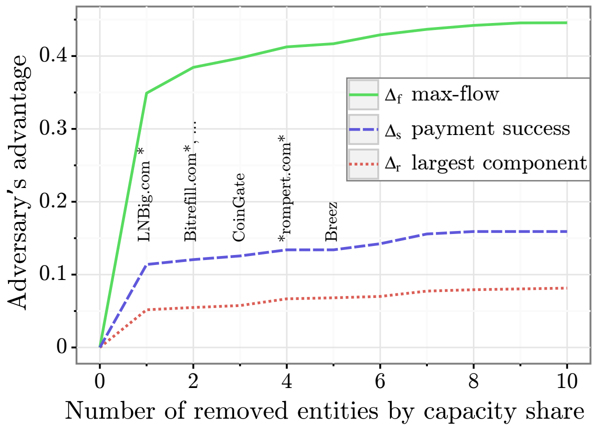

Figure 4: Adversary's advantage in doing DoS attacks in LN.

effect on the number of reachable nodes; (ii) Δ_f defined as the average maximum flow between every two nodes in the graph, representing thereby the effect of the attack on liquidity; and (iii) Δ_s defined as the payment success ratio, representing thereby the effect of the attack on the payments. Following their approach for estimation, we perform a uniform random sampling of 1000 pairs of nodes to compute Δ_f and Δ_s.

We obtain the results shown in Figure 4. We observe that a possibly low resource adversary that carry out a DoS attack targeted to a single Lightning entity (26 nodes in total) already gets an advantage that is only slightly improved when targeting more entities. As each entity is hosted on a single autonomous system, it could be sufficient to attack a single hosting provider. In this regard, our results differ from those in [23]. Multiple high degree nodes are likely using several hosting providers, increasing the attack's cost. Second, even with a lower budget requirement, our DoS attack targeted at entities yields a similar adversary's advantage as in [23] for all the metrics when only considering one entity with 26 nodes.

7.4 Lightning Entities Across Paths

In this section, we study to what degree the security and privacy of individual payments between any two nodes in the LN are affected by our clustered entities. In the LN, a payment between two nodes is typically routed through the cheapest path between them, where the cost associated to the path is calculated by the sum of fees charged by each intermediary node. An intermediary node charges a fee composed of a rate fee proportional to the payment amount, and a base fee that is independent. We computed the cheapest paths between all node pairs for a varying payment amount. This allows us to study the following properties, that are visualized in Figure 5.

Value privacy. The payment value is observed by every single intermediary in the path. Thus, according to our results, a reduced number of entities know how many coins are being transferred in the LN, giving them undue advantage over competitors (e.g., to set the fees or target products to users accordingly). Being an intermediary also has a second implication, from a security point of view: a payment between any two nodes can be aborted by a *single* intermediary node that simply drops it. A few entities can thus stop between 25% and 40% of the payments in the LN, and this fraction grows significantly if multiple entities were to collude. Given the decentralized payment protocol used currently in the LN, it is not possible for the sender to pinpoint which intermediary node has stopped the payment. Therefore the sender needs to blindly guess what node is the malicious one and possibly pay higher fees to circumvent it.

7.5 Lightning Entities Within a Path

From the results in the previous section, we observe that a few entities are frequently intermediary nodes for many paths used for payments in the LN. In this section, we are interested in studying whether a single entity has *more than one* node as an intermediary in a single path. This setting has further security and privacy implications in practice.

Relationship anonymity. Assume a path where an entity has two nodes, one of which is the immediate successor of the sender and the other is the immediate predecessor of the receiver. In such a setting, the fact that information uniquely identifying a payment is sent across the path (e.g., a hash value used to cryptographically secure the payment), allows the entity to learn the sender and receiver for such payment, even when other simultaneous payments may be using part of the path. This privacy attack breaks the notion of *relationship anonymity* as described in [13].

We evaluated the presence of such a threat in the Lightning Network with the results shown in Figure 5. We observe that there are between 5% and 20% of the paths prone to this privacy issue even when as little as one entity behaves adversarial. The reason why relationship anonymity is much more vulnerable for higher payment amounts is straightforward: Only a few channels have sufficient capacity, and several of them are operated by the same entity (i.e. LNBig.com), forcing more payments to go through them.

Wormhole attack. Assume now a path where an entity has two intermediary nodes at any position in the path with the condition that there are other honest nodes between them. The latter are at risk of becoming a victim of the *wormhole attack* as described in [14]. They are tricked into locking capacity at their channels to facilitate the payment but never contacted again to release those funds so that channels get locked for a certain timeout period established as system parameter which is on the order of days in the current implementation. While similar in spirit to the griefing attack, the wormhole attack differs in two main points: (i) the attacker entity does not need to be the sender and receiver of the payment; and (ii) the attacker entity can successfully settle the payment at the channels in the path other than those being attacked (i.e., channels

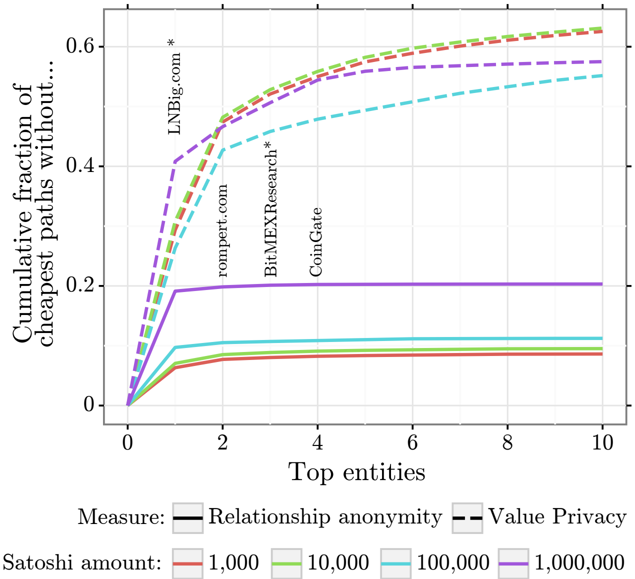

Figure 5: Fraction of cheapest paths without value privacy and relationship anonymity by different amounts to be transferred.

between two nodes of the attacker), so that the attacker also gets the fees for providing an apparently successful payment at the eyes of the sender and the receiver.

As shown in Figure 6, surprisingly the entity with the highest impact in this attack is not LNBIG.com as in the previous attacks. In this case, the entity associated to rompert.com can perform the wormhole attack for about 0.5% of all cheapest paths in the LN. While this number is lower than in previous attacks, the effect of this attack actively disrupts users in the path (i.e., their coins get locked), different to privacy-based attacks where the payment finishes successfully and the privacy breach is computed locally and passively at the attacker node.

7.6 The Good and the Bad for Routing in LN

The possibility of deanonymization, which opens up with the cluster and linking algorithms proposed in this work, has the following implications arising from the security and privacy issues in the routing of payments discussed so far.

Honest users can use the knowledge about entities to search for payment paths that avoid them. However, this may not always be possible, especially for users who control a node with only a few channels. In addition, alternative paths circumventing these entities may be more expensive, which represents a trade-off between security/privacy and transaction fees.

On the other hand, the fact that honest users can learn about entities and avoid them may have a negative impact on the business model of these entities. The business incentive for the LN nodes is to offer many channels and to set their fees so that as many payments as possible are routed through them. More payments are also associated with higher revenue potential. From this point of view, the deanonymization techniques presented in this work are not beneficial for routing entities.

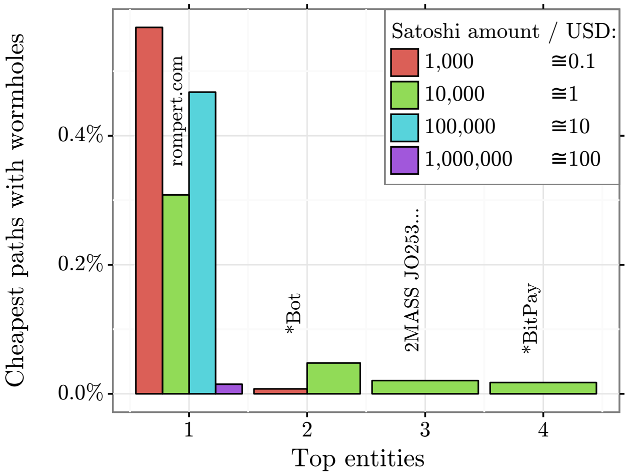

Figure 6: Cheapest paths prone to the wormhole attack.

8 Conclusion and Future Work

In this paper, we presented two novel linking algorithms to reveal the ownership of Bitcoin addresses that are controlled by LN nodes using publicly-available data. We also developed four Bitcoin address clustering algorithms and one LN node clustering algorithm that allowed us to link 26.3% of the addresses in our dataset to 43.7% of the public nodes, and cluster 589 Lightning nodes into 233 entities. Finally, we evaluated the security and privacy implications of our findings in the LN, where we find that a single entity controls 36% of the overall capacity and a few entities have a large impact on value privacy and payment relationship anonymity. These few entities also have a large overlap with those entities that would be candidates for high-impact attacks, the success of which can have significant negative effects on payment success and throughput for the entire Lightning network. With the new knowledge at hand, users can make better privacy and routing decisions.

Scalability issues appear in a broad range of blockchain applications and layer-2 protocols are increasingly considered as possible solutions. In light of these developments, we find an interesting venue for future work to evaluate whether our heuristics apply to layer-2 protocols other than Lightning Network such as the Raiden Network for Ethereum.

Acknowledgments

This work is partially funded by the European Research Council (ERC) under the European Unions Horizon 2020 research (grant agreement No 771527-BROWSEC), by PROFET (grant agreement P31621), by the Austrian Research Promotion Agency through the Bridge-1 project PR4DLT (grant agreement 13808694); by COMET K1 SBA, ABC, by CoBloX Labs and by the Austrian Science Fund (FWF) through the Meitner program (project agreement M 2608-G27). The authors would also like to thank Peter Holzer and Marcel Müller for setting up and starting the LN data collection process and Peter Sebastian Nordholt and Jacob Illum for providing the attribution data and valuable feedback.

References

[1] Community, L.N.: Wip: Dual funding (v2 channel establishment protocol). Github Issue, `https://github.com/lightningnetwork/lightning-rfc/pull/524`

[2] Decker, L.N.C.: New release: c-lightning 0.7.1. Blostream Blog Post, `https://medium.com/blockstream/new-release-c-lightning-0-7-1-9fca65debeb2`

[3] Fuller, V., Li, T., Yu, J., Varadhan, K.: Classless inter-domain routing (cidr): An address assignment and aggregation strategy. IETF RFC1519 (1993)

[4] Gomaa, W., Fahmy, A.: A survey of text similarity approaches. International Journal of Computer Applications **68**, 13–18 (04 2013). https://doi.org/10.5120/11638-7118

[5] Gudgeon, L., Moreno-Sanchez, P., Roos, S., McCorry, P., Gervais, A.: Sok: Off the chain transactions. IACR Cryptology ePrint Archive **2019**, 360 (2019), `https://eprint.iacr.org/2019/360`

[6] Harrigan, M., Fretter, C.: The unreasonable effectiveness of address clustering. In: 2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld). pp. 368–373. IEEE (2016)

[7] Jian-Hong, L., Kevin, P., Tiziano, S., Christian, D., J, T.C.: Lightning network: a second path towards centralisation of the bitcoin economy. arXiv preprint arXiv:2002.02819 (2020)

[8] Jourenko, M., Kurazumi, K., Larangeira, M., Tanaka, K.: Sok: A taxonomy for layer-2 scalability related protocols for cryptocurrencies. Cryptology ePrint Archive, Report 2019/352 (2019), `https://eprint.iacr.org/2019/352`

[9] Kalodner, H., Goldfeder, S., Chator, A., Möser, M., Narayanan, A.: Blocksci: Design and applications of a blockchain analysis platform. arXiv preprint arXiv:1709.02489 (2017)

[10] Kappos, G., Yousaf, H., Maller, M., Meiklejohn, S.: An Empirical Analysis of Anonymity in Zcash. In: 27th USENIX Security Symposium (USENIX Security 18). pp. 463–477. USENIX Association, Baltimore, MD (2018), `https://www.usenix.org/conference/usenixsecurity18/presentation/kappos`

[11] Kappos, G., Yousaf, H., Piotrowska, A., Kanjalkar, S., Delgado-Segura, S., Miller, A., Meiklejohn, S.: An empirical analysis of privacy in the lightning network (2020)

[12] Kus Khalilov, M.C., Levi, A.: A survey on anonymity and privacy in bitcoin-like digital cash systems. IEEE Communications Surveys Tutorials **20**(3), 2543–2585 (2018)

[13] Malavolta, G., Moreno-Sanchez, P., Kate, A., Maffei, M., Ravi, S.: Concurrency and privacy with payment-channel networks. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017. pp. 455–471. ACM (2017). https://doi.org/10.1145/3133956.3134096, `https://doi.org/10.1145/3133956.3134096`

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

[15] Martinazzi, S., Flori, A.: The evolving topology of the lightning network: Centralization, efficiency, robustness, synchronization, and anonymity. PLOS ONE **15**(1), 1–18 (01 2020). https://doi.org/10.1371/journal.pone.0225966, `https://doi.org/10.1371/journal.pone.0225966`

[16] Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G.M., Savage, S.: A fistful of bitcoins: Characterizing payments among men with no names. In: Proceedings of the 2013 Conference on Internet Measurement Conference. p. 127–140. IMC '13, Association for Computing Machinery, New York, NY, USA (2013). https://doi.org/10.1145/2504730.2504747, `https://doi.org/10.1145/2504730.2504747`

[17] Moreno-Sanchez, P., Zafar, M.B., Kate, A.: Listening to whispers of ripple: Linking wallets and deanonymizing transactions in the ripple network. PoPETs **2016**(4), 436–453 (2016). https://doi.org/10.1515/popets-2016-0049, https://doi.org/10.1515/popets-2016-0049

[18] Möser, M., Soska, K., Heilman, E., Lee, K., Heffan, H., Srivastava, S., Hogan, K., Hennessey, J., Miller, A., Narayanan, A., Christin, N.: An empirical analysis of traceability in the monero blockchain. Proceedings on Privacy Enhancing Technologies **2018**, 143–163 (06 2018). https://doi.org/10.1515/popets-2018-0025

[19] Nowostawski, M., Jardar, T.: Evaluating Methods for the Identification of Off-Chain Transactions in the Lightning Network. Applied Sciences **9**(12), 2519 (2019). https://doi.org/10.3390/app9122519, https://www.mdpi.com/2076-3417/9/12/2519

[20] Poon, J., Dryja, T.: The Bitcoin Lightning Network:. lightning.network p. 59 (2016)

[21] Reid, F., Harrigan, M.: An analysis of anonymity in the bitcoin system. In: 2011 IEEE International Conference on Privacy, Security, Risk, and Trust, and IEEE International Conference on Social Computing (2011), http://arxiv.org/pdf/1107.4524

[22] Robinson, D.: Htlcs considered harmful, stanford Blockchain Conference, Stanford, CA, USA, January 2019. http://diyhpl.us/wiki/transcripts/stanford-blockchain-conference/2019/htlcs-considered-harmful/

[23] Rohrer, E., Malliaris, J., Tschorsch, F.: Discharged payment channels: Quantifying the lightning network's resilience to topology-based attacks. In: 2019 IEEE European Symposium on Security and Privacy Workshops, EuroS&P Workshops 2019, Stockholm, Sweden, June 17-19, 2019. pp. 347–356. IEEE (2019). https://doi.org/10.1109/EuroSPW.2019.00045, https://doi.org/10.1109/EuroSPW.2019.00045

[24] Seres, I.A., Gulyás, L., Nagy, D., Burcsi, P.: Topological analysis of bitcoin's lightning network. In: Panos, P., Kotsireas, I., Yike, G., William, K. (eds.) Mathematical Research for Blockchain Economy. pp. 1–12. Springer International Publishing, Cham (2020)

[25] Victor, F.: Address clustering heuristics for ethereum. In: Financial Cryptography and Data Security 2020 (2020)

[26] Vu, B.: Announcing lnd v0.10-beta! Lightning Labs Blog Post, https://lightning.engineering/posts/2020-04-30-lnd-v0.10/

A On-Chain Bitcoin Entity Clustering

In Figure 7, we depict an illustrative example of the patterns that we use to cluster Bitcoin entities. Following the same notation of Figure 1, the star pattern clusters the entities e_1, e_2, e_3 and e_4 into one entity, the snake pattern clusters the entities e_1, e_2 and e_3 into one entity, the collector and the proxy pattern cluster the entites e_1, e_2, e_3 and e_4 into one entity.

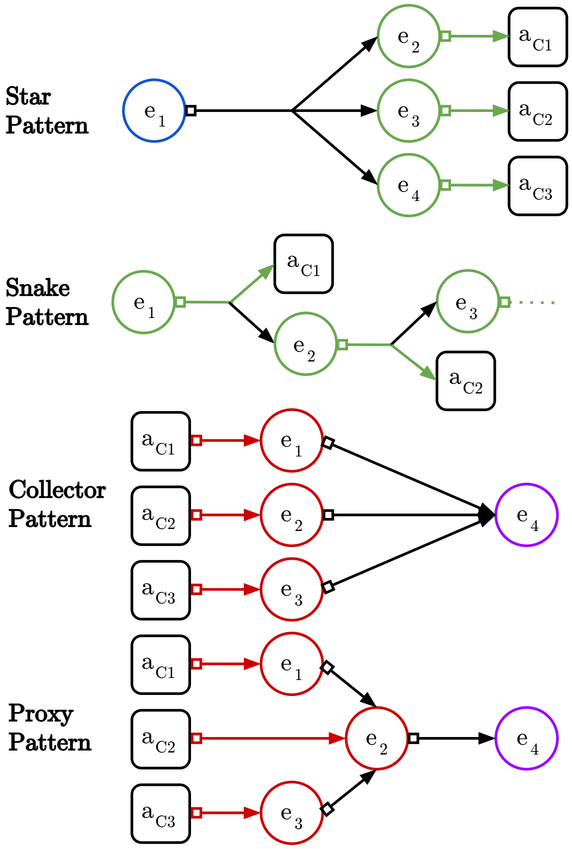

Figure 7: On-chain clustering heuristics. In the star patter, a source entity replenish different funding entities that can be clustered together. In the snake pattern, a series of funding transactions are performed using the change address of a previous funding transaction as input and these funding entities can be clustered. In the collector and proxy pattern, multple settlmenet entities merge their coins in one single entity and these settlement entities can be clustered.

B Evaluation of Different String Distance Measures

As illustrated in Figure 8, we compare the string measures lcs (longest common substring), Jaro, Jaro-Winkler, Levenshtein, Damerau-Levensthein and Hamming distance. For those dis-

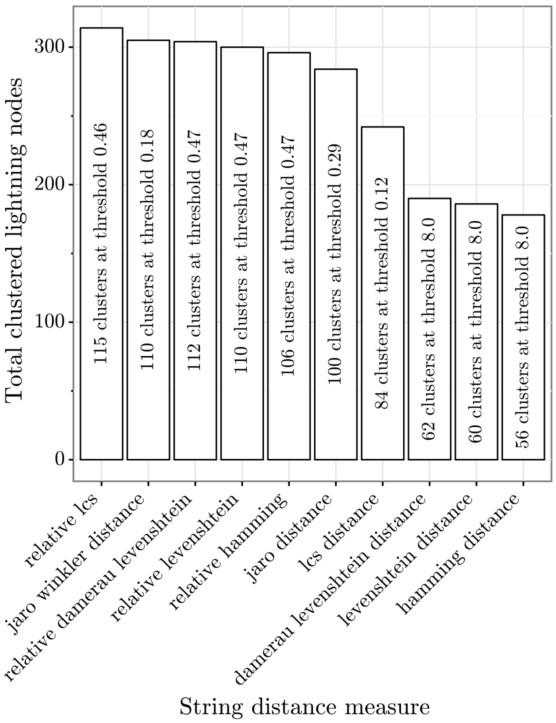

Figure 8: Comparison of string distance measures for alias clustering. The relative longest common substring (lcs) measure performs best. It grouped 314 Lightning nodes into 115 clusters. The threshold of 0.46 implies that for two aliases to be clustered, their longest common substring needs to account for about half of the longer alias.

tances where the result is not already between 0 and 1, we normalize the distance by dividing by the longer one of the two aliases to be compared.

For example, a popular string edit distance, the Levensthein distance, measures the minimum number of single character edits that are needed to transform one string into another. Here an edit, refers to replacement, insertion or deletion. For a detailed overview on text similarity approches we refer the reader to [4].

The results indicate that several normalized string distances exhibit similar performance, while the relative longest common substring yields the best performance. The optimal threshold of 0.46 can be interpreted as follows: if a common substring is identified between two aliases, it needs to account for about half of the length of the longer alias. A practical similarity comparison is illustrated in Figure 9. We have chosen a subset of aliases that contains all observed aliases of LNBig.com, multiple nodes containing the substring textitLightning, and some randomly selected alises. At the threshold, 3 clusters are identified. In two of them, all nodes are hosted on the same AS. So the inital result would be 2 identified clusters. As the cluster consisting of *LNBIG.com [lnd-27/old-lnd-19]* and *LNBIG.com [lnd-28/old-lnd-22]* are just additional aliases that have been seen over time, but actually belong to some of the same public keys of the other LNBig cluster, the two clusters are joined.

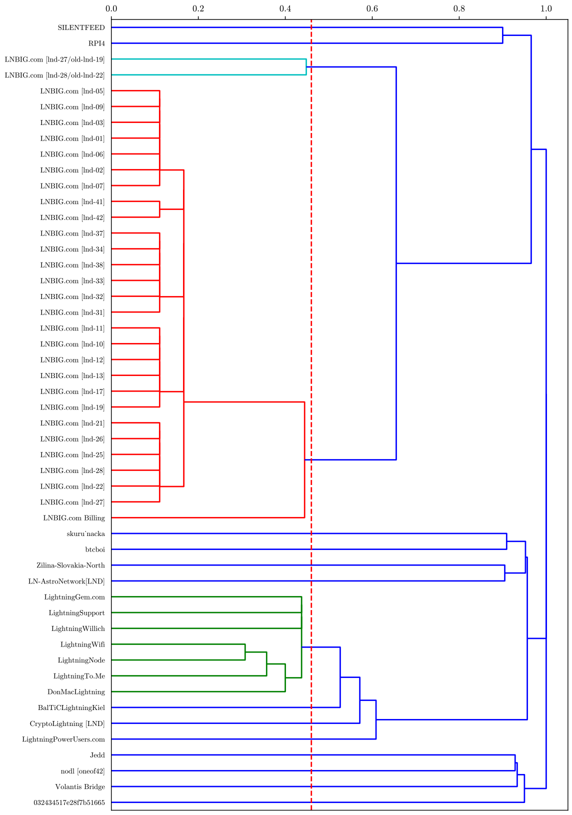

Figure 9: Example of a dendrogram illustrating alias similarity. Here, the relative lcs distance metric has been used along with the optimal threshold of 0.46 (red vertical dashed line). As a result, 3 clusters are found. However, only in the LNBig clusters, all nodes are hosted on the same AS. In addition, two nodes behind these aliases appear in both clusters. Therefore, in this example, they are merged and one LNBig.com cluster is the result.