Cross-Layer Deanonymization Methods in the Lightning Protocol

Matteo Romiti[1], Friedhelm Victor[2], Pedro Moreno-Sanchez[3], Peter Sebastian Nordholt[5], Bernhard Haslhofer[1], and Matteo Maffei[4]

[1] Austrian Institute of Technology {matteo.romiti, bernhard.haslhofer}@ait.ac.at
[2] Technische Universität Berlin friedhelm.victor@tu-berlin.de
[3] IMDEA Software Institute pedro.moreno@imdea.org
[4] Technische Universität Wien matteo.maffei@tuwien.ac.at
[5] Chainalysis psn@chainalysis.com

Abstract. Bitcoin (BTC) pseudonyms (layer 1) can effectively be deanonymized using heuristic clustering techniques. However, while performing transactions off-chain (layer 2) in the Lightning Network (LN) seems to enhance privacy, a systematic analysis of the anonymity and privacy leakages due to the interaction between the two layers is missing. We present clustering heuristics that group BTC addresses, based on their interaction with the LN, as well as LN nodes, based on shared naming and hosting information. We also present linking heuristics that link 45.97% of all LN nodes to 29.61% BTC addresses interacting with the LN. These links allow us to attribute information (e.g., aliases, IP addresses) to 21.19% of the BTC addresses contributing to their deanonymization. Further, these deanonymization results suggest that the security and privacy of LN payments are weaker than commonly believed, with LN users being at the mercy of as few as five actors that control 36 nodes and over 33% of the total capacity. Overall, this is the first paper to present a method for linking LN nodes with BTC addresses across layers and to discuss privacy and security implications.

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 Bitcoin (BTC) blockchain. The idea is that two users create a funding transaction that locks coins, thereby creating a payment channel between them [9]. 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.

While there are different payment channel designs, the BTC Lightning Network (LN) [22] is the most widespread PCN implementation to date. At the time of writing (September 2020), according to 1ml.com, the LN features a network of 13,902 public active nodes, 37,003 channels and a total capacity of more than 1,108.70 BTC, worth 11,569,618 USD.

Apart from scalability, PCNs are considered beneficial to improve the well-known lack of privacy of cryptocurrencies [4], where the anonymity claim stemming from the usage of pseudonyms in on-chain transactions has been largely refuted from both academia and industry [15]. The key to an effective deanonymization of BTC pseudonyms lies in heuristic methods, which cluster addresses that are likely controlled by the same entity [19]. In practice, entities correspond to user wallets or software services (e.g., hosted wallet, exchange) that control private keys on behalf of their users.

In this work, we challenge the widespread belief that the LN greatly improves privacy by showing for the first time how LN nodes can be linked to BTC addresses, which results in a bi-directional privacy leakage affecting LN and BTC itself. Related research [25, 26, 18, 11, 21] already focused on security and privacy aspects on the PCN layer, but, so far, none of them focused on linking off-chain LN nodes to on-chain BTC addresses. This is a challenging task because such links are not provided explicitly in the LN protocol as they would severely affect the privacy of node operators (e.g., revealing their business to competitors).

Our Contributions. Our methodology is structured in two main strategies: (i) heuristics on layer 1, to create clusters of BTC addresses controlled by the same actor, and on layer 2, clusters of LN nodes; and (ii) heuristics to link these clusters across layers. In Section 4, we present four novel on-chain clustering heuristics (star, snake, collector, proxy), which group BTC addresses based on their interaction patterns with the LN. With these heuristics, we can cluster 19.39% of all BTC entities funding an LN channel, and 13.40% of all entities closing a channel. We also present an LN node clustering heuristic leveraging public announcements of aliases and IP addresses, which allows us to group 1,251 nodes into 301 clusters. In Section 5, we present two novel cross-layer linking algorithms. One exploits that the same BTC address can be used to close one channel and then re-use the coins to open a new channel, which allows us to link 26.48% of the LN nodes to 20.96% BTC addresses in our dataset, when combined with the previous on- and off-chain clustering heuristics. The other algorithm exploits the reuse of a single BTC entity for opening several channels to different LN nodes and it allows us to link 29.61% of the addresses to 45.97% of nodes.

Given these results, we finally discuss the impact of our deanonymization techniques on the privacy of BTC entities as well as the security and privacy of the LN. In a nutshell, we are able to (i) attribute 21.19% of the BTC addresses with information from the LN (e.g., IP addresses); (ii) measure the centralized control of the capacity in the LN and observe that as few as five actors consisting of 36 nodes control over 33% of the total capacity; (iii) show that as few as five users can threaten the security of the LN by means of (possibly targeted) DoS attacks and violate the privacy of over 60% of the cheapest payment paths because they are routed through them.

For the reproducibility of the results, we make our dataset and our implementation openly available at `https://github.com/MatteoRomiti/lightning_study`[6].

2 Background and Problem Statement

We now define the simplified model and terminology used throughout this paper, elaborating then on the cross-layer linkage problem, as well as on related work in this area. For further details on PCNs, we refer to recent surveys [9, 12].

2.1 BTC Blockchain (Layer 1)

A BTC **address** a is a tuple containing (i) a number of coins (in Satoshis) associated to this address; and (ii) an excerpt of the BTC script language that denotes the (cryptographic) conditions under which a can be used in a transaction. Although in principle it is possible that a can be spent under any condition that can be expressed in the BTC script language, in practice most of the addresses share a few conditions: (i) requiring a signature σ on the transaction verifiable under a given public key pk; and (ii) requiring two signatures $\{\sigma_1, \sigma_2\}$ verifiable with two given public keys pk_1 and pk_2 (i.e., multisig address). We say that an address a is owned by a user if she can produce the required signature/s.

A BTC **transaction** tx is identified by txid computed as the hash of the *body* of tx, i.e., $H(\mathsf{Input}, \mathsf{Output})$. Input denotes the set of addresses set as input and being spent in tx; and Output is the set of addresses set as output. A transaction can have also a change output, where coins and address are owned by the same user controlling the inputs.

We define a BTC **entity** e as a set $e := \{a_i\}$ of addresses controlled by the same user as clustered with the well-known and effective [10] co-spending heuristic [19]. This heuristic 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.

A **BTC wallet** is the software used by a BTC user to handle BTC addresses owned by her. A wallet may correspond to a BTC entity, if addresses are reused.

2.2 Nodes and Payment Channels in the LN (Layer 2)

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; IP denotes the IP address associated with 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

[6] The proprietary attribution data from Chainalysis is not included in the published dataset. The reader can contact the company for further inquiry.

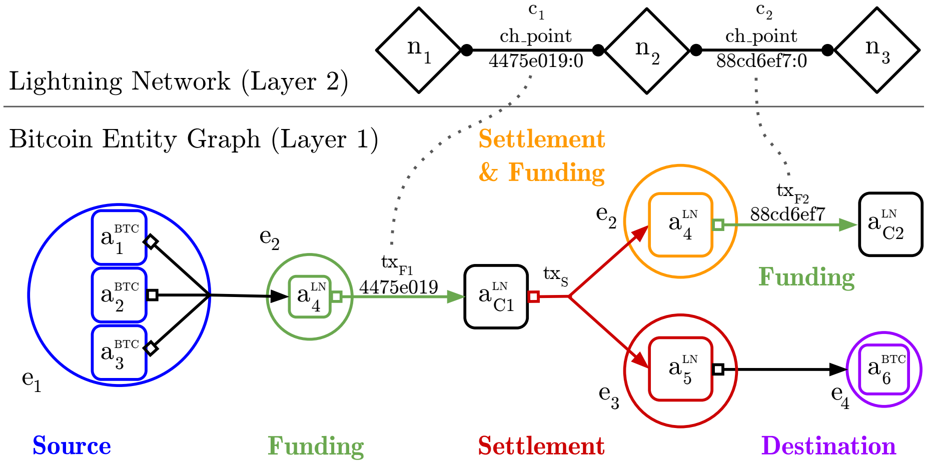

Fig. 1. Life cycle of an LN channel. At layer 1, a source entity e_1 tops up entity e_2 that is then used in tx_{F1} as funding entity of the channel c_1 represented by multisig address a_{C1}^{LN}. The channel c_1 is established at layer 2 between the nodes n_1 and n_2. The channel c_1 is then closed 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}^{LN}. The coins in the other settlement entity, e_3, are instead collected into a destination entity e_4, not directly involved in the LN.

that is set to the identifier tx.txid of the funding transaction tx that created the channel. As the transaction may have several outputs, chpoint also contains the output index of the multisig address that locks the funds in the channel (e.g., chpoint:choutindex); while n_1 and n_2 are the nodes of the channel.

An **LN wallet** is the software used by an LN user to manage her node, as well as the channels of this node. In practice, an LN wallet comes with an integrated BTC wallet to open and close channels in the LN. Recent releases of two LN wallet implementations (*lnd* and *c-lightning*) [27, 5] enable opening/closing a channel using an external BTC wallet.

2.3 Cross-Layer Interaction

In this section, we describe the interaction between BTC and the LN by means of the example illustrated in Figure 1. Assume Alice wants to open a payment channel with Bob. Further, assume that Alice has a BTC wallet with coins in address a_1^{BTC} and she wants to open a payment channel with Bob. Additionally assume that Alice has never interacted with the LN before and only has an LN wallet, whose integrated BTC wallet handles a_4^{LN}. 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 BTC wallet (represented by entity $e_1 := \{a_1^{BTC}, a_2^{BTC}, a_3^{BTC}\}$) to her LN wallet (entity $e_2 := \{a_4^{LN}\}$), to

top up the LN wallet from the BTC 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^{LN}_{C1} shared between Alice and Bob. In the next step, Alice creates a **funding transaction** tx_{F1} where $tx_{F1}.\mathsf{Input} := a^{BTC}_4$, $tx_{F1}.\mathsf{Output} := a^{LN}_{C1}$, and $tx_{F1}.\mathsf{txid} := H(tx_{F1}.\mathsf{Input}, tx_{F1}.\mathsf{Output})$.[7] After tx_{F1} appears on the BTC blockchain, the payment channel c_1 between Alice and Bob is effectively open. The channel c_1 is then represented in the payment channel network as the tuple $(c_1.\mathsf{chpoint}, n_1, n_2)$, where n_1 and n_2 are 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, Alice and Bob create a **settlement transaction** tx_S where $tx_S.\mathsf{Input} := a^{LN}_{C1}$, $tx_S.\mathsf{Output} := \{a^{LN}_4, a^{LN}_5\}$ so that a^{LN}_4 belongs to Alice, and a^{LN}_5 belongs to Bob. The cornerstone of payment channels is that Alice and Bob do not publish tx_S in the BTC 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 BTC blockchain, thereby unlocking the coins from a^{LN}_{C1} into two BTC 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 Bob to a third address where Bob needs to provide 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 [1]).

Collection. After the settlement transaction appears in the BTC blockchain, Bob gets the coins in his LN wallet. As a final step, Bob might want to get his coins into a different BTC wallet of his own. For that, Bob transfers funds from a^{LN}_5 to a^{BTC}_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 a 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^{BTC}_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

[7] 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 [3] and currently only single-funded channels are implemented in practice.

the transaction (and the output index) that fulfills the condition $tx_{F1}.\mathsf{txid} = c_1.\mathsf{chpoint}$. While this is trivial, 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 to cluster BTC entities based on their interactions with the LN and then unambiguously link these clusters to LN nodes that are under their control. Technically, this corresponds to finding a function that takes a set of LN channels as input and returns tuples of the form (entity, node) for which it can be asserted that the LN node is controlled by the linked BTC entity.

2.5 Related Work

Single-layer security attacks on the LN topology were the focus of many recent studies: Rohrer et al. [25] measured the LN topology and found that the LN is highly centralized and vulnerable to targeted (e.g., DoS) attacks. Similarly, Seres et al. [26] found that the LN provides topological stability under random failures, but is structurally weak against rational adversaries targeting network hubs. Also, Martinazzi and Flori [18] have shown that the LN is resilient against random attacks, but very exposed to targeted attacks, e.g., against central players. Lin et al. [11] inspected the resilience of the LN and showed that removing hubs leads to the collapse of the network into many components, evidence suggesting that this network may be a target for the so-called split attacks. Single-layer LN privacy has recently been studied by Kappos et al. [14], 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. Nowostawski and Ton [21] conducted an initial cross-layer analysis and investigated footprints of the LN on the public BTC blockchain in order to find which transactions in the BTC blockchain are used to open and close LN channels. Our work instead uses the funding and settlement transactions (and more) as input data to investigate for the first time: (i) the link between LN nodes and BTC entities; (ii) clustering of BTC entities allowed by blockchain footprint for the interaction of these entities with the LN; and (iii) the associated security and privacy implications.

3 Dataset

In this section, we present the data we collected for our analysis.

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 $98,431$ channels, $37,996$ of which were still open on September 9, 2020. The most recent channel in our dataset was opened on September 9, 2020, while the oldest was opened on January 12, 2018. 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 2020-09-09 (the time of preparing the dataset), if the nodes had still public channels open. Finally, we observe that channels in our dataset were established between $10,910$ distinct nodes.

3.2 On-chain Data: BTC 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.* Then, 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 open-source GraphSense API[8] and the Blockstream API[9]. We thereby extracted $98,240$ funding transactions[10] and $60,447$ 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 [13] to filter CoinJoin transactions because they would merge addresses of unrelated users. For the same reason, we also made sure that no CoinJoins from Wasabi nor Samourai[11] 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 BTC blockchain up to block $647,529$ (2020-09-09 23:06), amounting for a total of $566,776,778$ transactions and $703,443,739$ addresses clustered into $336,847,691$ entities. After having extracted the BTC entities that were involved in opening and closing payment channels, we attributed them using the Chainalysis API[12] 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 heuristic on these addresses (*# Entities*). We can clearly observe that the number of distinct source entities ($196,131$) is lower than the number of destination entities ($424,732$), 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. These unbalanced numbers might be due to funds going from settlement entities to mixing services, as we

[8] `https://api.graphsense.info/`

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

[10] Some channels were opened with the same funding transaction.

[11] `https://github.com/nopara73/WasabiVsSamourai`

[12] `https://www.chainalysis.com/`

Table 1. On-chain Dataset Summary.

	Source	Funding	Settlement	Destination
# Addr		170,777	88,166	
# Entities	196,131	96,838	53,371	424,732
# Addr (Exp.)	70,638,581	196,818	2,243,525	107,474,279
# Services	5,812	1	5	67,969
# Relations		203,328	438,725	

discuss later. 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 [10]. Finally, this table also lists the number of identified service entities (*# Services*). We only found them in few cases for funding (1) and settlement (5) entities, probably because mostly non-custodial wallets are used when opening and closing channels and known services in our dataset behave only as source and destination entities. Roughly 0.9% of all source entities were categorized, with the majority (0.8%) being exchanges. On the settlement side, we identified 10% of all destination entities as wallets being controlled by services, with the majority (8%) being mixing services. We can not fully account for this strong connection to mixing, but it does suggest that many LN users are privacy-aware. Indeed, there is evidence that the LN is recognized as a privacy technology complementary to mixing. e.g., the well-known mixing wallet Wasabi suggests LN as one way to enhance privacy when using the wallet[13].

3.3 Ground Truth Data: LN Payments

We devised and implemented a simple process that allows us to create a ground truth dataset of entity-node pairs that can then be compared with our linking results as a validation step. We first run our linking algorithms resulting in an initial set of entity-node pairs. We then found a trade-off for selecting the target nodes: some randomly-selected linked nodes for generality purposes and some other nodes with the highest number of settlement transactions as a sign of being very active on the network and reusing funds, a useful aspect for the next steps. Next, we managed to open channels, perform payments and close channels with 52 of them. For these nodes that received coins from us, we are able to see their settlement entity, but only 11 nodes further spent the settlement funds in other transactions, necessary for us to capture their spending behaviors with our heuristics. We additionally managed to have channels open to us from 3 LN nodes that provide inbound channels as a service, revealing their funding

[13] https://docs.wasabiwallet.io/using-wasabi

entities. We performed this activity at the beginning of September 2020 (block 646559) and after waiting some days to let the nodes spend our coins, we run the linking algorithms again on our latest dataset (until block $647,529$) so that for these targeted nodes we have both ground truth and heuristically-obtained links to entities. In Section 5.3, we compare this ground truth data with our linking results, while a more detailed explanation of the methodology to extract this data is presented in Appendix A.

4 Clustering Heuristics

In this section we introduce the on-chain and off-chain clustering heuristics.

4.1 On-Chain BTC Entity Clustering (Layer 1)

LN-blockchain interactions result in monetary flows from source to funding and from destination to settlement entities (see Figure 1). When analyzing the resulting entity graph abstraction, we observed four patterns (see Figure 2).

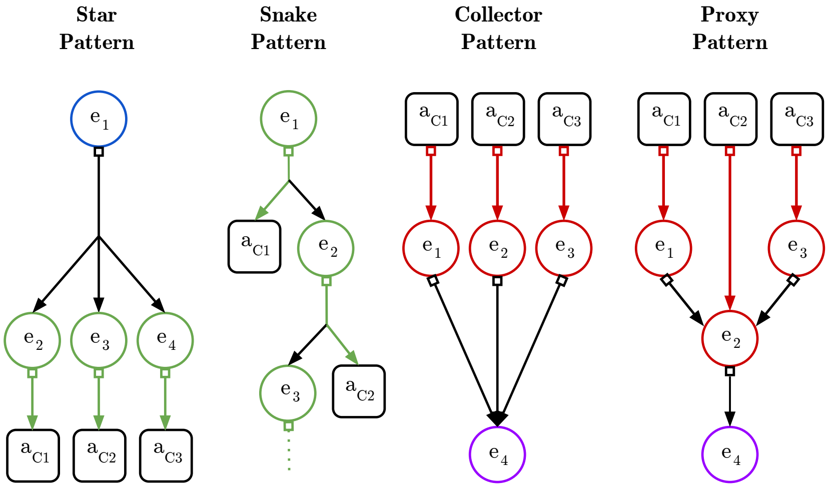

Fig. 2. On-chain clustering heuristics. Following the same notation of Figure 1, in the star pattern, a source entity e_1 replenishes three different funding entities creating a single cluster (e_1, e_2, e_3, e_4). In the snake pattern, a series of funding transactions are performed using the change address of a previous funding transaction as input and the funding entities can be clustered (e_1, e_2, e_3). In the collector and proxy pattern, multiple settlement entities merge their coins in one single entity and these settlement entities can be clustered (e_1, e_2, e_3, e_4).

First, several funding entities received funds from the same source entities with one source entity transferring coins to several funding entities. This forms a *star-shaped pattern* and reflects a current LN wallet feature, which requires[14] users to transfer funds from an external wallet (source entity) to an internal wallet (funding entity) before opening a channel. If these source entities are not services, which is rarely the case (see Section 3), then we can assume:

Definition 1 (Star Heuristic). *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.*

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 Bitcoin Change-Heuristic [19]).

Definition 2 (Snake Heuristic). *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.*

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.

Definition 3 (Collector Heuristic). *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.*

Fourth, we found a refined 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.

Definition 4 (Proxy Heuristic). *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.*

We compute the above heuristics as follows: we construct 1-hop ego-networks for the funding and settlement entities and extract funding relations and settlement relations (see Section 3). Next, we compute all weakly-connected components in these graphs and filter them by the conditions defined above.

Table 2 shows the number of BTC 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 31% of all funding components. On the settlement side, 23% of all

[14] We note that this requirement may no longer be there if the "fund-from-external-wallet" functionality, already available in the recent release [27], is widely adopted.

Table 2. On-chain clustering results.

	Star (F)	Snake (F)	Collector (S)	Proxy (S)
# Components	52 (0.3%)	5,638 (31%)	1,476 (14%)	989 (9%)
# Entities	139	18,512 (19%)	3,923 (7%)	3,229 (6%)
# Addresses	144	18,556	6,146	12,292

components either match the collector or the proxy pattern. Consequently, we were able to group 19.39% (18,651) of all funding entities and 13.40% (7,152) of all settlement entities. This corresponds to 18,700 funding addresses and 18,438 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 BTC 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 Chainalysis, 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 TXOs (transaction outputs). If outputs of 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. Despite not pervasive on the network, Coinjoins and similar solutions could, in theory (e.g., if used as funding transactions), obfuscate the entity linked to an LN node behind a set of unrelated addresses.

4.2 Off-Chain LN Nodes Clustering (Layer 2)

We have also designed an algorithm to cluster LN nodes based on aliases and IPs reported in the LN, along with their corresponding autonomous systems (AS). If a set of node aliases share a common substring, and they are hosted on the same AS, we cluster them. Similarly, if a set of nodes report the same IP or onion address, we cluster them assuming they are controlled by the same entity. This allows us to cluster 1,251 nodes into 301 clusters. Due to space constraints, we defer the description of this clustering to Appendix B.

5 Linking LN Nodes and BTC Entities

In this section, we present two algorithms that link LN nodes to the BTC entities that control them. In both of these heuristics, we do not consider settlement transactions with more than two output entities (1.9% of the settlement transactions), as they are not a cooperative close and do not allow us to unambiguously link nodes and output entities. Furthermore, we ignore settlement transactions that involve punishment transactions [2]. Finally, 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

Our linking algorithm builds upon the usage pattern that appears when a payment channel is closed and the user that receives the coins from such channel reuses them to open a new payment channel. 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 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_2 controls node n_2.

Definition 5 (Linking Algorithm 1: Coin Reuse). *Assume that a BTC entity e opens an LN channel* $c_1 := (\mathsf{chpoint}_1, n_1, n_2)$*. If e is used as settlement entity to close the LN channel* c_1 *and also as funding entity to open a new LN channel* $c_2 := (\mathsf{chpoint}_2, n_1, n_3)$*, and the nodes* n_2 *and* n_3 *have activity period overlap, then the user controlling entity e also controls the LN 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 83 tuples of (funding transaction, settlement transaction, funding transaction) and 22 entities reusing their addresses for opening and closing channels. Once these 22 entities are linked to LN nodes, all the other output entities in the settlement transactions of these 22 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 9,042 entities linked to 2,170 nodes, thus having cases where a node is linked to multiple entities. In total, considering the number of entities we have in our dataset (138,457 overall, both funding and settlement side[15]) the heuristic is able to link 6.53% of them. This result is a lower bound on the possible number of linked entity-node pairs because the linking

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

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 $32,321$ entities, 27.98% 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 19.89% of the total ($10,910$ overall) and 46.91% of the nodes for which there exists at least one channel that has been closed using a 2-output-entity settlement transaction, confirming 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 the 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 an LN node, n_2, with an associated BTC entity e_1 that funds channel c_1 between node n_2 and n_1 through tx_{F1}. Furthermore, we have an LN wallet with an associated BTC 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 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 BTC wallet (and thus entity), which could allow one to open two channels from two different nodes, but to the same node, using the same BTC entity, creating a false-positive case for the heuristic.

Countermeasures. The default functionality of LN wallets followed thus far by virtually all users consists of 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) BTC wallets, helping thus to diversify the source of funds. We observe that recent versions of the LN wallet *lnd* and *c-lightning* have started to support this functionality [27, 5].

5.2 Linking Algorithm 2: Entity Reuse

In this linking algorithm we leverage the usage pattern that appears when a user reuses the same BTC wallet (e.g., the one integrated within the LN wallet) to open several payment channels. Thus, 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 Appendix C.

Definition 6 (Linking Algorithm 2: Entity Reuse). *If there are $N_C > 1$ channels opened by one single funding entity e that have only one LN node n in common, and there are at least two nodes n_x and n_y in these channels with activity period overlap, then the user controlling entity e controls node n too.*

We can link $9,904$ entities to $2,170$ nodes which correspond to 7.15% of all the entities and 22.84% of all the nodes respectively.

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 BTC 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 BTC 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 report that it provides the same entity-node links that are in common with the linking algorithm presented in Section 5.1.

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 [27, 5].

5.3 Validation

For the validation of the heuristics presented in this work we use the ground truth dataset presented in Section 3. For each of the 11 nodes that received funds from us, we compare their set of ground truth settlement entities with their set of linked entities from our linking algorithms. If there is an intersection between these two sets, we say that the link is validated. In total, we find that 7 nodes (i.e., 63%) are validated. The validation for the 3 nodes that opened channels to us is the same, but uses their ground truth funding entities as set for comparison with the set of linked entities from our linking algorithms. In this case, we can validate 2 nodes. The lack of validation for the other nodes can have several reasons: i) as reported in Section 3, we notice that only 11 out of the 52 nodes receiving our coins (by default on newly-generated BTC addresses) also spent them, ii) the coins spent are not merged with funds from other channels or iii) the coins are spent and merged with funds from channels missing in our dataset. Nevertheless, one should note that over time our ground truth data will increase and more nodes could be validated as soon as they spend our funds.

We believe that our small ground truth dataset is a reasonable trade-off between obtaining a representative picture of the LN main net and a responsible and ethical behavior that does not alter the LN properties significantly. We also see our methodology to gather ground truth data as an interesting contribution due to its scalability features: costs are relatively low (two on-chain transactions and LN routing fees for each targeted node) and executable in a programmatic way. We defer a more detailed description of this methodology to Appendix A.

6 Assessing Security and Privacy Impact

We merged the results of our clustering algorithms (Section 4) and our linking algorithms (Section 5), thereby increasing the linking between entities and nodes as shown in Table 3. We defer to Appendix D a detailed description of the contribution for each heuristic individually.

6.1 Privacy Impact on BTC Entities (Layer 1)

The linking algorithms and clustering algorithms described in this work allow attributing activity to BTC entities derived from their interaction with the LN. Assume that a cluster is formed by a certain number of BTC entities and LN nodes, then if any of the LN nodes has publicly identifiable information (e.g., alias or IP address), this information can be attributed to the BTC entities as well. In total, we can attribute tagging information to $17,260$ different entities that in total account for $50,456$ different addresses, which represent 21.19% of our dataset.

This deanonymization is based purely on publicly available data[16] and can be carried out by a low budget, passive adversary that simply downloads the BTC blockchain and the information from the LN. We envision that further impact can be achieved by a more powerful adversary (e.g., a BTC miner). Moreover, the possible deanonymization of BTC 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.

6.2 Security and Privacy Impact on the LN (Layer 2)

We have evaluated the implications of our clustering and linking algorithms in the security and privacy of the LN. In summary, we studied how the capacity of

[16] We note that Chainalysis attribution data is not strictly necessary for the linking algorithms.

Table 3. Summary results

Linking + Clustering	% addresses linked	% entities linked	% nodes linked
Linking Algorithm 1	18.16	6.53	19.89
Linking Algorithm 1 + all on/off-chain	20.96	8.14	23.64
Linking Algorithm 2	19.19	7.15	22.84
Linking Algorithm 2 + all on/off chain	29.61	12.72	45.97

the LN is distributed across actors and found that a single actor controls over 24% of the total LN capacity and as few as five actors consisting of 36 nodes control over 33% of the total capacity. Few LN actors are thus in a privileged situation that can be used to diminish the security and privacy of the LN. For instance, we observed that the entity with the highest capacity can render useless over 40% of the channels for a period of time by means of DoS attacks. Similar issues appear from the privacy point of view, where just 5 actors can learn the payment amounts used in up to 60% of the cheapest paths in the LN and determine who pays to whom in up to 16% of the cheapest paths. Due to space constraints, we defer a detailed discussion of our security and privacy assessment to Appendix E.

7 Conclusion and Future Work

In this paper, we presented two novel linking algorithms to reveal the ownership of BTC addresses that are controlled by LN nodes using publicly-available data. We also developed four BTC address clustering algorithms and one LN node clustering algorithm that allowed us to link 29.61% of the BTC addresses in our dataset to 45.97% of the public LN nodes, and cluster 1,251 LN nodes into 301 actors. Finally, we discussed the security and privacy implications of our findings in the LN, where we find that a single actor controls 24% of the overall capacity and a few actors have a large impact on value privacy and payment relationship anonymity. These few actors also have a large overlap with those 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 LN.

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 the LN 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, by the Austrian Science Fund (FWF) through the Meitner program (project agreement M 2608-G27) and by the Austrian security research programme KIRAS of the Federal Ministry of Agriculture, Regions and Tourism (BMLRT) under the project KRYPTOMONITOR (879686). The authors would also like to thank Peter Holzer and Marcel Müller for setting up and starting the LN data collection process.

References

1. Hash time locked contracts. Wiki post, `https://en.bitcoin.it/wiki/Hash\_Time\_Locked\_Contracts`
2. Community, L.N.: Bitcoin transaction and script formats, `https://github.com/lightningnetwork/lightning-rfc/blob/master/03-transactions.md`
3. Community, L.N.: Wip: Dual funding (v2 channel establishment protocol). Github Issue, `https://github.com/lightningnetwork/lightning-rfc/pull/524`
4. Decker, C.: Privacy in lightning. Blog post (2018), `https://snyke.net/post/privacy-in-lightning/`
5. 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`
6. Egger, C., Moreno-Sanchez, P., Maffei, M.: Atomic Multi-Channel Updates with Constant Collateral in Bitcoin-Compatible Payment-Channel Networks. In: Computer and Communications Security (2019)
7. Fuller, V., Li, T., Yu, J., Varadhan, K.: Classless inter-domain routing (CIDR): An address assignment and aggregation strategy. IETF RFC1519 (1993)
8. Gomaa, W., Fahmy, A.: A survey of text similarity approaches. International Journal of Computer Applications **68**, 13–18 (04 2013)
9. Gudgeon, L., Moreno-Sanchez, P., Roos, S., McCorry, P., Gervais, A.: SoK: Layer-Two Blockchain Protocols. In: Financial Cryptography and Data Security (2020)
10. Harrigan, M., Fretter, C.: The unreasonable effectiveness of address clustering. In: 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) (2016)
11. Jian-Hong, L., Kevin, P., Tiziano, S., Christian, D., J, T.C.: Lightning network: a second path towards centralisation of the bitcoin economy (2020), `https://arxiv.org/abs/2002.02819`
12. 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`
13. Kalodner, H., Goldfeder, S., Chator, A., Möser, M., Narayanan, A.: BlockSci: Design and applications of a blockchain analysis platform (2017), `https://arxiv.org/abs/`
14. 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), `https://arxiv.org/abs/2003.12470`
15. Kus Khalilov, M.C., Levi, A.: A Survey on Anonymity and Privacy in Bitcoin-Like Digital Cash Systems. Communications Surveys Tutorials **20**(3), 2543–2585 (2018)

16. Malavolta, G., Moreno-Sanchez, P., Kate, A., Maffei, M., Ravi, S.: Concurrency and Privacy with Payment-Channel Networks. In: Conference on Computer and Communications Security (2017)
17. Malavolta, G., Moreno-Sanchez, P., Schneidewind, C., Kate, A., Maffei, M.: Anonymous Multi-Hop Locks for Blockchain Scalability and Interoperability. In: Network and Distributed System Security Symposium (2019)
18. 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)
19. 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: Internet Measurement Conference (2013)
20. Mizrahi, A., Zohar, A.: Congestion attacks in payment channel networks (2020), `https://arxiv.org/abs/2002.06564`
21. Nowostawski, M., Jardar, T.: Evaluating Methods for the Identification of Off-Chain Transactions in the Lightning Network. Applied Sciences **9**(12) (2019)
22. Poon, J., Dryja, T.: The Bitcoin Lightning Network (2016), `lightning.network`
23. Pérez-Solà, C., Ranchal-Pedrosa, A., Herrera-Joancomartí, J., Navarro-Arribas, G., Garcia-Alfaro, J.: Lockdown: Balance availability attack against lightning network channels. Cryptology ePrint Archive, Report 2019/1149 (2019), `https://eprint.iacr.org/2019/1149`
24. 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/`
25. Rohrer, E., Malliaris, J., Tschorsch, F.: Discharged Payment Channels: Quantifying the Lightning Network's Resilience to Topology-Based Attacks. In: European Symposium on Security and Privacy Workshops (2019)
26. 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 (2020)
27. Vu, B.: Announcing lnd v0.10-beta! Lightning Labs Blog Post, `https://lightning.engineering/posts/2020-04-30-lnd-v0.10/`
28. Yu, B., Kermanshahi, S.K., Sakzad, A., Nepal, S.: Chameleon Hash Time-Lock Contract for Privacy Preserving Payment Channel Networks. In: Steinfeld, R., Yuen, T.H. (eds.) International Conference Provable Security (2019)

A Ground Truth Data Collection

To send coins to LN nodes and discover their settlement entities, we run two nodes in the LN. One is the *sending* node, n_s, and the other the *receiving* node, n_r. We make sure that n_r is connected to the LN by a channel with a good amount of incoming capacity, so that it can receive a number of LN payments. We then have n_s open a channel, c_t, to a given *target* node, n_t. Once c_t is open, we route a payment of amount a from n_s to n_r over c_t. On successful payment, we close the channel c_t. We then repeat this experiment for a number of target nodes n_t.

The purpose of making a payment in this way is to increase the balance of n_t on channel c_t to a before the channel is closed. This ensures that when c_t

is closed, the entity of n_t receives an on-chain payment of a BTC. If the entity further spends this amount of BTC we can apply our heuristics and attempt to link entity to LN node. An advantage of routing the payment over n_t rather than making the payment to n_t directly, is that we will not have to request an invoice from each n_t, or rely on the currently experimental *key send* mechanism. Instead, we just have n_r generate a set of invoices we can use for the experiment.

For each target node, we attempted to open a channel of 100,000 satoshis capacity and make a payment of 1,000 satoshis to the receiving node. We chose these amounts as they allow us to perform the experiments with a relatively low amount of capital. While the payment is low, it is above the BTC dust limit of 546 satoshis, ensuring that the target node will receive the funds.

Unfortunately, in some cases the experiment would fail, either because the channel would fail to open, or the payment was unsuccessful. When the channel fails to open most often this is due to the target node not responding to the channel opening request (presumably as the node is no longer on-line). Occasionally, our requests to open channels will fail due to the requested channel not meeting a policy set by the target node for opening new channels (e.g., some nodes will only accept channels above a certain capacity). Once a channel was established, the payment could fail because a suitable route could not be found between receiving and sending node, although this was rare.

B Off-Chain LN Node Clustering

The operator of an LN 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[17], 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 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. [8]) such as the Levenshtein, Hamming and Jaro-Winkler distances. Naturally, however, aliases can be similar, but do 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.

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

Each IP address is part of a Classless Inter-Domain Routing (CIDR) [7] 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 LN node IP address. If an alias-based node cluster consists only of IP addresses associated to a single ASN, we conclude that the LN nodes are hosted by the same network operator and is, therefore, more likely operated by a single entity. In addition, we also cluster LN 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 LN 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 LN 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 LN 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 LN 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.1) and concluded that the relative longest common substring measure yields the best results. In particular, 363 LN nodes have been grouped into 126 clusters. The IP-based clustering yields 1135 clustered LN nodes, 241 of which are already part of the alias-based clusters. By merging these clusters, the final cluster count is 301, with a total of $1,251$ LN nodes clustered. The two largest clusters are nodl-lnd-s007-* (88 nodes) and *-lnd-gar-nodl-it (65 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. Of the 301 identified clusters, 20 are hosted on Amazon servers, but are identified as distinct clusters due to their different naming schemes. Within the overall time frame of our dataset, 313 (2.9%) different lightning nodes have at some point in time been hosted with Amazon. In general, however, filtering alias clusters to those running on the same AS should result in few 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 LN 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 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).

B.1 Evaluation of Different String Distance Measures

As illustrated in Figure 3, we compare the string measures lcs (longest common substring), Jaro, Jaro-Winkler, Levenshtein, Damerau-Levensthein and Hamming distance. For those distances 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 [8].

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 4. We have chosen a subset of aliases that contains all observed aliases of LNBig.com, multiple nodes containing the substring Lightning, and some randomly selected aliases. At the threshold, 3 clusters are identified. In two of them, all nodes are hosted on the same AS. So the initial result would be

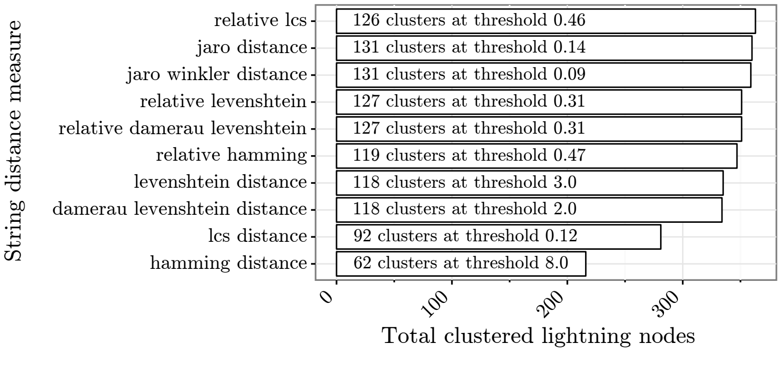

Fig. 3. Comparison of string distance measures for alias clustering. The relative longest common substring (lcs) measure performs best. It grouped 363 LN nodes into 126 clusters. The threshold of 0.46 implies that for two aliases to be clustered, their longest common substring needs to account for about half of the longer alias.

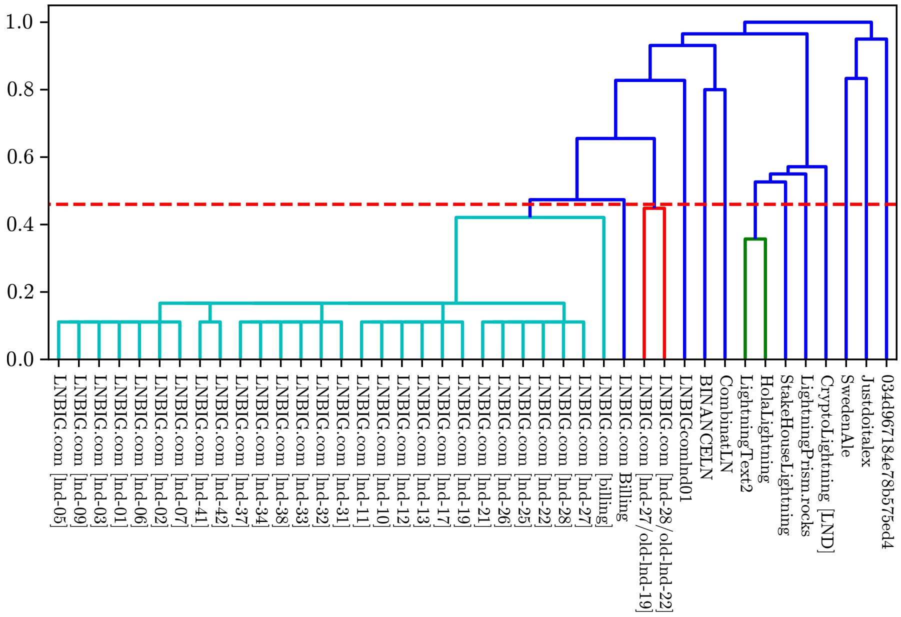

Fig. 4. Example of a dendrogram illustrating alias similarity. Here, the relative lcs distance metric has been used along with the optimal threshold of 0.46 (red vertical dashed line). As a result, 3 clusters are found. However, only in the LNBig clusters, all nodes are hosted on the same AS. Node ids behind the 2 LNBig clusters overlap. Therefore, in this example, one LNBig.com cluster is the result.

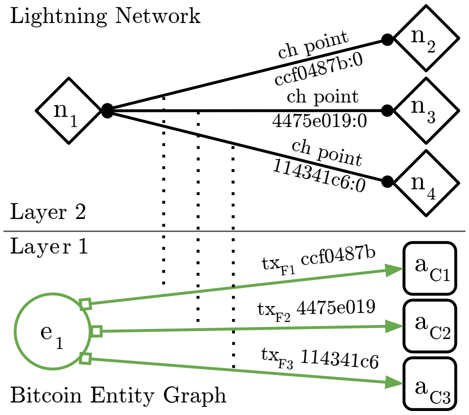

Fig. 5. Linking Algorithm 2: Entity reuse example. At layer 1, the funding entity e_1 is reused to perform $N_C = 3$ funding transactions. At layer 2, the corresponding channels are opened and there is one node, n_1, common to all the N_C channels and it can be linked to the funding entity e_1.

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.

C Illustrative Example Linking Algorithm

An illustrative example of the linking algorithm described in Section 5.2 is shown in Figure 5, 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.

D Details for Combining Heuristics

In the best case (last entry in Table 4) we get to 29.61% of linked addresses and 45.97% of linked LN 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 4 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

Table 4. Summary results

Linking + Clustering	% addresses linked	% entities linked	% nodes linked
Linking Algorithm 1	18.16	6.53	19.89
Linking Algorithm 1 + stars	18.16	6.53	19.89
Linking Algorithm 1 + snakes	18.18	6.53	19.89
Linking Algorithm 1 + collectors	18.36	6.64	19.97
Linking Algorithm 1 + proxies	20.12	7.64	22.81
Linking Algorithm 1 + all on-chain	20.96	8.14	26.48
Linking Algorithm 1 + all on/off-chain	20.96	8.14	23.64
Linking Algorithm 2	19.19	7.15	22.84
Linking Algorithm 2 + stars	19.22	7.17	22.96
Linking Algorithm 2 + snakes	26.8	11.09	39.84
Linking Algorithm 2 + collectors	19.39	7.26	22.97
Linking Algorithm 2 + proxies	21.16	8.27	25.6
Linking Algorithm 2 + all on-chain	29.61	12.72	42.16
Linking Algorithm 2 + all on/off-chain	29.61	12.72	45.97

a factor of 2. However, not every clustering algorithm contributes the same to the overall results. We discuss each of them next.

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 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 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 4.1 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 BTC, 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 an 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 3% for linked nodes) for both linking algorithms when applied without the other patterns.

Table 5. LN users controlling most capacity

User	Node count	Share of total capacity contributed
LNBig.com *	26	24.07%
bfx-lnd*	2	4.20%
BitRefill.com, ...	3	2.37%
CoinGate	2	1.98%
Breez	3	0.52%

Collector-pattern contribution. Similar to the proxy pattern, this behavior merges the coins from different settlement transactions into one single entity, with the difference that this last one is not directly involved in the LN settlements. This pattern appears to be less common and powerful compared to the proxy pattern.

Off-chain node clustering contribution. Assume there is a cluster of nodes obtained with the heuristic presented in Appendix B 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 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 310 indirectly-linked nodes, which corresponds to an additional 2.84% of nodes linked, while for the linking algorithm 2 we find 416 indirectly-linked nodes which correspond to an additional 3.81% of nodes linked.

E Security and Privacy Implications

In this section, we evaluate the security and privacy implications of our clustering and linking algorithms in the security and privacy impact on the LN.

E.1 Wealth Distribution and Impact of Griefing Attacks in the 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 users found during our analysis. For that, we take a recent snapshot of the LN from 2020-09-09 and extract the capacity controlled by each user. If a channel has been created by a user that has been linked to a node, we can attribute the full capacity of the channel to that node. For all other channels, we assume that each user controls half of the capacity. Under these assumptions, we observe that the overall capacity of the LN is distributed as shown in Table 5. In particular, a single user controls over 24% of the overall capacity in the LN and as few as 237 nodes (3.2%) control over 80% of the capacity. This result refines the previous study in [11] where they find that 80% of the capacity is controlled by 10% of the nodes.

This result shows that few LN users are in a privileged situation that they can potentially use to selectively prevent other LN nodes from transacting in the network, for instance, launching a *griefing attack* [24] against the victim nodes. In the griefing attack, the attacker finds a path of the form $n_1 \rightarrow n_2 \rightarrow \ldots \rightarrow n_k$ 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 corresponding with the capacity available at the channel of the victim. However, as shown in Table 5, the uneven distribution of wealth in the LN makes this a small investment if the attacker is one of the users with high capacity. In fact, we evaluated the possible damage that each user in the LN can infringe by launching this griefing attack with the results shown in Figure 6. As expected from the wealth distribution, the user 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 LN, which amount for about 14% of the total capacity.

We remark that although griefing attacks have a cost for the adversary (i.e., the adversary needs to lock some of its own channels), the adversary can still benefit from griefing other nodes, as studied in the literature. For instance, Pérez-Solà et al. [23] show that the adversary can launch a griefing attack to block LN middle nodes in multi-path payments. Mizrahi and Zohar [20] as well as Rohrer et al. [25] show how similar attacks can be used to block as many high liquidity channels as possible, disconnect channels from the LN and isolate individual nodes from the LN. If the adversary is successful, the attack gives the adversary a dominant position in the LN, which can be later exploited either for exploiting privacy (e.g., off-chain payment data gathering) or for economic rewards (e.g., increasing the benefits in term of fees reducing the number of competing LN gateway nodes).

E.2 Vulnerability to DoS attacks in the 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[18]. In this state of affairs, we study here the effect of DoS attacks targeted at the LN users found in this work.

Based on the LN snapshot we iteratively remove the nodes and channels corresponding to a given user, starting with the users that control the most capacity. We then compare the resulting graph with the original one to evaluate

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

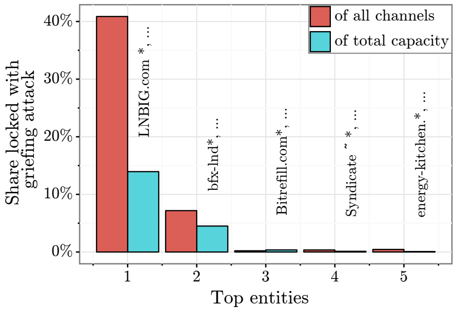

Fig. 6. Fraction of all LN channels and capacity vulnerable to a griefing attack launched by most effective entity.

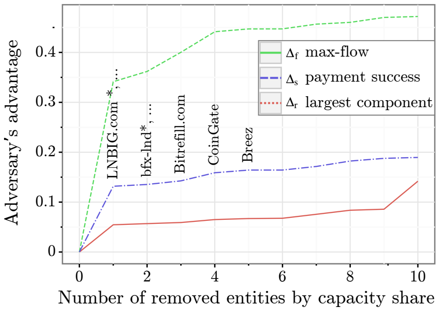

Fig. 7. Adversary's advantage in impeding max-flow, payment success and largest component size — given an entity DoS.

the *adversary's advantage* (i.e., attack's success) attributed to a DoS targeted to such user. Following [25], we characterize the notion of adversary's advantage as $\mathit{\Delta}_m := \left|1 - \frac{m'}{m}\right|$ where m is the a priori measurement and m' is the a posterior one. The higher $\mathit{\Delta}_m$ becomes, the higher the success of the attack according to the metric m. We consider the three metrics as defined in [25]: (i) $\mathit{\Delta}_r$ defined as the number of nodes within the biggest component in the graph, representing thereby the effect on the number of reachable nodes; (ii) $\mathit{\Delta}_f$ defined as the average maximum flow between every two nodes in the graph, representing thereby the effect of the attack on liquidity; and (iii) $\mathit{\Delta}_s$ 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 $\mathit{\Delta}_f$ and $\mathit{\Delta}_s$.

We obtain the results shown in Figure 7. We observe that a possibly low resource adversary that carry out a DoS attack targeted to a single LN user (LNBig.com, 26 nodes in total) already gets an advantage that is only slightly improved when targeting more users. By attacking this entity, the max-flow of the LN can be reduced by one third, and payment success be reduced by 12%. As each user 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 [25]. 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 users yields a similar adversary's advantage as in [25] for all the metrics when only considering one user with 26 nodes.

E.3 LN Users on Payment 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 users. 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 value privacy property, that is visualized in Figure 8.

Value privacy. The payment value is observed by every single intermediary in the path. Thus, according to our results, a reduced number of users 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 single user can thus stop almost 40% of the payments in the LN, and this fraction grows to 60% if the top 5 users 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.

E.4 LN Users With Multiple Nodes on Payment Paths

From the results in the previous section, we observe that a few users are frequently intermediary nodes for many paths used for payments in the LN. In this section, we are interested in studying whether a single user 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 a user 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 user 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 [16].

We evaluated the presence of such a threat in the LN with the results shown in Figure 8. We observe that there are between 5% and 16% of the paths prone to this privacy issue even when as little as one user 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 user (i.e. LNBig.com), forcing more payments to go through them.

Wormhole attack. Assume now a path where a user 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 [17]. 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

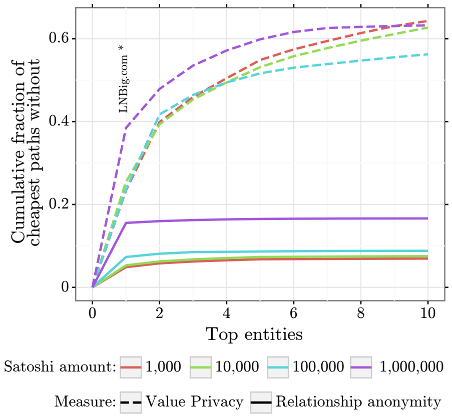

Fig. 8. Fraction of cheapest paths without value privacy and relationship anonymity by different amounts to be transferred.

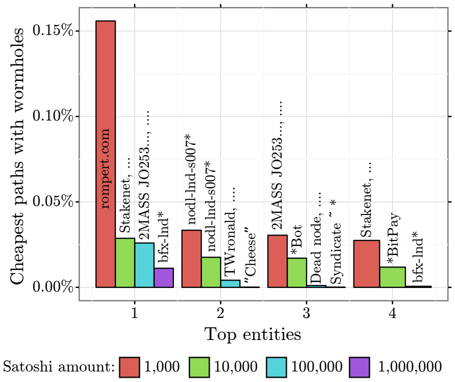

Fig. 9. Cheapest paths prone to the wormhole attack. In theory, the rompert.com entity could exploit the most wormholes.

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 user does not need to be the sender and receiver of the payment; and (ii) the attacker user can successfully settle the payment at the channels in the path other than those being attacked (i.e., channels 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 9, surprisingly the user with the highest impact in this attack is not LNBIG.com as in the previous attacks. In this case, the user associated to rompert.com can perform the wormhole attack for about 0.15% of all cheapest paths in the LN. While this number is much 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.

These results call for the inclusion into the current LN of countermeasures recently proposed in the literature. For instance, Egger et al. [6] and Malavolta et al. [17] have proposed alternative payment schemes for the LN that prevent the worhmole attack. Moreover, the payment schemes proposed by Malavolta et al. [16, 17] and Yu et al. [28] provably prevent the relationship anonymity attack that are otherwise currently feasible in the LN.

E.5 The Good and the Bad for Routing in the 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 users 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 users 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 users and avoid them may have a negative impact on the business model of these users. 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 users.