SoK: A Taxonomy for Layer-2 Scalability Related Protocols for Cryptocurrencies

Preprint · April 2019

CITATIONS
0

READS
138

4 authors, including:

Mario Larangeira
Tokyo Institute of Technology/Input Output HK
9 PUBLICATIONS **13** CITATIONS
SEE PROFILE

Keisuke Tanaka
Tokyo Institute of Technology
149 PUBLICATIONS **971** CITATIONS
SEE PROFILE

SoK: A Taxonomy for Layer-2 Scalability Related Protocols for Cryptocurrencies

Maxim Jourenko
Department of Mathematical and Computing Sciences,
School of Computing,
Tokyo Institute of Technology.
Tokyo-to Meguro-ku Oookayama 2-12-1 W8-55.
Email: jourenko.m.ab@m.titech.ac.jp.

Kanta Kurazumi
Department of Mathematical and Computing Sciences,
School of Computing,
Tokyo Institute of Technology.
Tokyo-to Meguro-ku Oookayama 2-12-1 W8-55.
Email: kurazumi.k.aa@m.titech.ac.jp.

Mario Larangeira
Department of Mathematical and Computing Sciences,
School of Computing,
Tokyo Institute of Technology.
Tokyo-to Meguro-ku Oookayama 2-12-1 W8-55.
Email: mario@c.titech.ac.jp.
Input Output Hong Kong - IOHK.
Email: mario.larangeira@iohk.io.

Keisuke Tanaka
Department of Mathematical and Computing Sciences,
School of Computing,
Tokyo Institute of Technology.
Tokyo-to Meguro-ku Oookayama 2-12-1 W8-55.
Email: keisuke@is.titech.ac.jp.

***Abstract*—Blockchain based systems, in particular cryptocurrencies, face a serious limitation: scalability. This holds, especially, in terms of number of transactions per second. Several alternatives are currently being pursued by both the research and practitioner communities. One venue for exploration is on protocols that do not constantly add transactions on the blockchain and therefore do not consume the blockchain's resources. This is done using off-chain transactions, *i.e.*, protocols that minimize the interaction with the blockchain, also commonly known as *Layer-2* approaches. This work relates several existing off-chain channel methods, also known as payment and state channels, channel network constructions methods, and other components as channel and network management protocols, *e.g.*, routing nodes. All these components are crucial to keep the usability of the channel, and are often overlooked. For the best of our knowledge, this work is the first to propose a taxonomy for all the components of the Layer-2. We provide an extensive coverage on the state-of-art protocols available. We also outline their respective approaches, and discuss their advantages and disadvantages.**

***Index Terms*—Blockchain, Off-chain, Channels, Scalability.**

I. Introduction

Blockchain is the main data-structure behind the successful rebirth of digital cash from its first attempts [1, 2], firstly by Bitcoin [3] and now with several decentralized cryptocurrencies [4–8]. Although Bitcoin's relative success in offering worldwide payment alternatives to the more traditional mechanisms, like, for example, VISA Network [9] and Paypal [10], is undeniable, it still has a long way ahead in terms of handling a larger number of transactions.

The technical challenge of increasing the number of transactions per second (TPS) of a blockchain system is urgent, and it is closely related to the inner workings of the system itself. Namely, to its *consensus protocol*. As a more concrete example, we refer to the Bitcoin network whose consensus protocol depends on the joint hash power of its nodes to perform the block leader election procedure, that is, the selection of the new block issuer, which is calibrated by design to happen every 10 minutes on average. In Proof-of-Stake (PoS) based systems, like Cardano/Ouroboros [7, 11] and EOS [4], analogous election exists also within a carefully designed (and strongly dependent on security guarantees) time slot for the generation of the new block. Let alone that, in order to confirm a transaction, it is required a minimum number of blocks added to be added in the main chain, which gives the *confirmation time*. The confirmation time, despite its central role in the security and stability of the system, imposes severe restrictions to the TPS rate of the overall platform, and therefore alternatives, arguably most of them centered or closely related to the consensus protocol, had been suggested, such as, for example, alternative structures for blockchain [12–16], change in the size of the block [17], signature aggregation [18], different consensus protocol [6, 19–22]. It is common to divide the approaches to increase the TPS rate in three cases:

- **Layer-0** (the network infra-structure): Here, optimization and special servers are employed to decrease the latency of the network, in order for example, to increase the TPS. An example of such approach is given by BloXroute [23];
- **Layer-1** (the consensus protocol): Arguably the layer which have more approaches in literature. They are different consensus architectures [4–7, 11, 20–22], where a complete taxonomy can be found in [24], different data-structure for blockchain [12, 13, 15, 16], valid chain

criteria [14], sharding [25–27], federation [28], and etc;
- **Layer-2** (the off-chain channel/protocol): Briefly protocols that perform with minimal interaction with the blockchain. Typically for opening, paying, closing and disputing a channel. In the realm of cryptocurrencies, concrete examples are given by Decker and Wattenhofer [29] and Lighting Network (LN) [30], and others. This layer encompasses much more protocols and algorithms than the channel construction alone, as we review in later sections. One of the contributions of this work is to present a taxonomy, a coherent list of protocols, for different levels in the stack of protocols in this layer.

Layer-2 channel constructions have few names in literature: off-chain, *payment* or, yet, *state* channels, seemingly without consensus among the research community about their meaning. Here, in the face of the lack of standard terminology, we settle to name *payment channels*, constructions that do not rely on smart-contracts, and *state channels* otherwise. Furthermore, we rely on the generic term *off-chain channel* when referring to either one regardless of its inner workings.

In a nutshell, a payment channel between two players, is when both participants decide to trade several transactions during a period of time, and in the end they settle on a final balance based on the transactions exchanged, then the channel is closed. This transaction method is suitable for very small amounts, *i.e.*, *micropayments*, which has a long history of research [31–38]. A fairly complete survey, which includes centralized solutions, was done by Ali et al. [39]. More recently, micropayments were also studied by Pass and shelat [40] in the setting of decentralized currencies, and for specific applications [41, 42]. The main advantage of such a setting is that for each transaction made during the period of the channel, do not need to be published in the blockchain, which is a clear advantage as concluded by McCorry et al. [43]. That is a set of transactions can be settled independently of the confirmation time, drastically improving the TPS rate.

A. More than Pairwise Channel

The channel constructions themselves are building blocks into a stack of protocols (or algorithms). A simple payment/state channel only paves the way for exchanging funds, which is of limited use since the channel yields a *capacity*, *i.e.*, the sum of funds initially deposited by the two players. A more interesting, and realistic, use is the concatenation of single channels into a *payment-channel network* (PCN). In such a setting, a node A can send payment to C without creating a specific channel from scratch with C, as long as both A and C are connected to a third node B, which relays transactions, typically by collecting fees.

A practical example of such a network is the already mentioned LN [30], which has multiple implementations [44–47]. Although it extends the channel functionality, a lurking problem persists: *how to find suitable routes within the network*? The resemblance with theory of networks is inevitable, naturally similar problems appear. For example, a node sending a payment needs to find a route, similarly to routing problems in networks. On the other hand, payment networks also present differences, for example, the cost of the fees in a particular route. Yet, assuming the mentioned routing problem is solved, another one still exists: *How about the stability of the channel/network? Do all nodes need to be online during all the time of the channel? Is the capacity of each pairwise channel a bottleneck for the whole network?*

All those questions are legit and relevant, and there are more as we will see in later sections. Protocols and ideas to tackle them are currently being considered and studied by the research and practitioner community very often in an independent fashion without a comprehensive framework or compilation. Given that these ideas can be scattered through several journal and conference papers, as well as Internet forums and repositories, compiling them is a major task. Hence, maybe not so surprisingly, at the best of our knowledge, it has not been done yet. This work fulfills this gap.

B. Our Contribution

We provide an extensive coverage on all the Layers-2 solutions for scalability in cryptocurrencies. Namely the off-chain channel, including probabilistic, simplex, and duplex, in addition with payment network constructions and network management protocols, *i.e.*, the upper levels protocols, which, we highlight, off-chain channels are building blocks.

A Taxonomy for the Layer-2 stack. As one of the immediate results of our survey, we devise a taxonomy, with further classification for the three cited levels: off-chain construction, payment-channel network, and network management. The levels compose a stack of protocols for Layer-2, as illustrated by Table I in Section II. As expected different levels, and their respective set of functions, define specific technical challenges, which by the best of our knowledge, were never compiled in a comprehensive framework. A quick look on Table I reveals intense research on areas and few works in others, or even no works for specific problems. We believe these observations are of strong interest by the research community, which, otherwise, would have to collect these works through numerous internet forums and repositories in addition to scientific journals.

Comparison on the body of work in level. Briefly, given the similarities with networks, techniques can be borrowed from currently known network algorithms but need to be adapted, this trend is promoted by Hoenisch and Weber [48], for *routing*, a crucial functionality within the *Network Management* Level in our classification. On the other hand, this trend contrasts with the technicalities of the *off-chain channel* and *network* levels. Our taxonomy decouples these different issues, and we further discuss them in Sections III (channels), IV (networks) and V (management).

As expected the off-chain channels body of work, *i.e.*, described in Section III, presents a greater number of works, in comparison, for example to the *Network* level, *i.e.*, described in Section IV, which is understandable given that, as described later, most of the protocols in Section IV derive from a

single technique to realize *conditional payments*, the *Hash Time Locked Contracts* (HTLC) technique introduced by the LN [30]. An example is Sprites [49] which employs a derived technique but with smart-contracts.

On the other hand, the amount of work on off-chain channels is comparable to that of the *Network Management* level, because, as later will be illustrated by Table I and described in Section V, of the significant amount of effort put on the functionality of *routing* among the nodes of the network by researchers and developers. However, even in this level, there are discrepancies on the attention given by the community, as the single work on re-balancing the network, *i.e.*, the REVIVE protocol [50], shows. Even though this is crucial to avoid the bottleneck on the capacity of the routes.

Security and privacy preserving constructions. In order to illustrate the type of problems between the levels, we observe that privacy and security permeates the layers, and some problems are well known in the literature. For example, exchange of funds, in off-chain channels, need to be consistent with the payments, *i.e.*, they cannot allow for one player to steal the coins from the other, that is they should provide *balance security*, property common to all safe channel constructions. Similar concerns exist for payment networks, but extended to the intermediate nodes, *i.e.*, the nodes that allow the connection between two other nodes with not common channel. Furthermore, PCN should prevent information to be leaked to intermediate nodes. Finally, network management systems should provide safe, in the sense of both balance and information is not leaked, routes for payers and payees as in Table I.

II. A Taxonomy for Layer-2

Here we propose a taxonomy for the existing protocols in Layer-2. We start by presenting an overview of our approach and justification for our choices. Later we present our classification illustrated by Table I.

A. Overview

We organize the different sets of protocols in three levels [1]. Each level embraces functions it performs. In other words, we further subdivide the level into "functions". The intuition is that a protocol deployed on Layer-2, is expected to fulfill a certain function within a level.

In order to illustrate and justify this design, take for example the most popular Layer-2 PCN technique: HTLC [30]. Briefly, for the sake of example, HTLC allows LN nodes to establish connection between themselves in order to transfer funds. A major challenge in this setting is to find a suitable route among the nodes, thus the function required is *Routing*. A routing protocol for the LN fits into the "Routing" function within the "Network Management" level. Later, Table I summarizes our taxonomy and the levels we identify in the literature. However, first, consider the following levels:

- **Off-chain Channels:** Here are the constructions for the channels themselves. In order words, how the nodes, relying on the transaction design of a cryptocurrency, can construct channels. These are the building blocks of the upper families.
- **Network:** Here are the techniques employed to create the networks themselves. Typically by concatenating existing pairwise channels, or establishing a *network* of more than two nodes right from start.
- **Network Management:** This the early mentioned family which embraces the protocols and algorithms that maintain the channels and network of channels. Typically they are responsible for keeping them "alive" and usable.

B. Functions on Each Level

Given the earlier identified levels, here we further break them down into functions.

a) Off-Chain Channels: The terminology in the literature is, in fact, not standard. However it is possible to observe two big functions (1) make payments and (2) keeping state. The first is simplest, it does not need to keep a state. Whereas the second is more general and requires the aid of smart contracts. We further review (1) the three existing types: *Duplex*, *Simplex*, and *Probabilistic*.

b) Network: In order to establish PCN, special techniques are required whilst it is also necessary to carry the payments themselves. Clearly, this level depends on the capabilities of the channels. Some off-chain channels do not support network construction capabilities. We further discuss this topic on Section IV.

c) Network Management: Over the network of channel several functionalities are required. We identified the following main ones in the existing literature:

- **Routing:** The payments can be carried over several nodes performing a protocol accordingly, however it is necessary to select a set of nodes.
- **Re-Balancing:** A difference from regular computer networks is that each pairwise channel has a capacity which may exhaust during the course the transactions exchange, and therefore it needs to be rebalanced.
- **Stability:** Another difference from the regular computer network, is that some constructions require to the nodes to be online in order to perform in the dispute phase of the protocols. Therefore there are protocols that mitigate or solve this requirement.
- **Anonymity/Privacy:** Information of nodes within a network, can be leaked as well as information about the performed transactions. Typically, the nodes in the middle of the channel can see the flow of funds. An alternative is to provide cryptographic machinery over the channels.

A major difficulty on gathering the information on protocols for Layer-2 is that, in contrast to scientific community literature, several protocols had been proposed in forums and internet repositories, and not in conference proceedings or journals.

[1] We justify the choice of the term "level", instead of the more natural "layer" as an alternative to avoid ambiguity with the terms Layer-0, Layer-1 and Layer-2 which permeate this work.

Level	Function	Protocols
Network Management **Section V**	Routing **Section V-B1**	Silent Whispers [51] Speedy Murmur [52] Spider Routing [53] Flare Routing [54] Splitting Payments [55] Hoenisch and Weber [48] Atomic Multi-path [56]
	Re-Balancing Channels **Section V-B2**	REVIVE [50]
	Channel Stability **Section V-B3**	PISA [57] Avarikioti et al. [58]
	Anonymity/ Privacy **Section V-B4**	Fulgor & Rayo [59] Tumblebit [60]
Network **Section IV**	Construction **Section IV-B**	HTLC [30] Sprites [49] State Assertion [61] Virtual Channels [62] Counterfactual [63]
Off-chain Channels **Section III**	State **Section III-C4**	Z-Channel [64] Perun [62] NoCUST [65]
	Duplex Payment **Section III-C3**	Raiden [66] Lightning [30] Decker et. al. [29] BOLT [67] Teechan [68] Burchert et al. [69] TumbleBit [60]
	Simplex Payment **Section III-C3**	Simplex [70] Dimitrienko et al. [71] Takahashi et al. [72]
	Probabilistic Payment **Section III-C3**	Pass et. al. [40] Hu and Zhang [41]

TABLE I: The channels stack and concrete protocols.

A unified compilation of issues and open problems is also, apparently, non existent. We address this with our suggested classification in Table I, which illustrates the protocol stack for Layer-2.

Multiparty computation. The reader should note that our definition of channel does not cover, for example, general secure multiparty computation protocols, where several users interact off-chain and provide *correctness proof* of the computation. Here, we are interested in protocols that realize a channel between two nodes or PCN, possibly through concatenation of channels, for purposes of transferring value. In other words, we leave out of our classification protocols with multiuser distribution of funds, as used in [73–76].

Network construction technique. The reader should note that Table I arranges LN [30] and Decker et. al. [29] in the Off-chain Channels Level, not in the Network Level, and this can be considered unusual, given that LN, in particular, is associated with PCN, and not the channel construction. We highlight that the construction technique used in both cases is the HTLC, therefore a more accurate setting is to locate the technique in the Network level instead. Moreover, this arrangement is consistent with the literature as in [77]. Similarly, we acknowledge that both [62] and [63] could be classified in the *Off-chain channel*, however these works seem to present a more "built in" machinery for networks therefore we classify them this way.

In the next section we detail each level from Table I, and cite concrete protocols that address the described functions.

III. Off-Chain Channel Level

Before reviewing the constructions for off-chain channels, it is convenient to review the structure of transactions and how they can be issued on typical cryptocurrency. That clarifies for the reader how channels can be established.

A. Preliminaries: Transaction Design Review

Payments done on a blockchain are recorded in the form of *transactions*, as in Figure 1. Each transaction consists of n inputs and m outputs, $n \in \mathbb{N}$, $m \in \mathbb{N}_0$. Each output consists of coins as well as a spending requirement. A plain payment would have as requirement a signature with the private key of the payee. Inputs on the other hand consist of a reference to an output of a previous transaction on the blockchain (which was not referenced by another input before) as well as a witness for the spending requirement, *e.g.*, a witness is a signature with the private key of the payee.

Spending requirements can be more complex than simply asking for the payee's signature. A relevant spending requirement for us is the k-out-of-l threshold signatures which are used in Bitcoin with the opcode **CHECKMULTISIG** within its scripting language. A witness to such a requirement need to contain signatures of k out of l specified parties. In our case we will use these spending requirements with $k = l = 2$. Further generalization was introduced in the Ethereum by introducing a full Turing complete programming language. This innovation gave rise to the first practical smart-contract platform. In a nutshell, smart-contracts have addresses with accounts and variables, which are triggered by receiving transactions.

A transaction can specify a *timelock* to enforce that miner will not include a transaction before a specified point in time. Timelocks can either be defined per input or for a whole transaction. Moreover timelocks can specify an absolute time or a point in time relative to when the referenced inputs were included in the blockchain. In Bitcoin an absolute timelock for a whole transaction can be specified with the field nTimelock, a relative timelock can be defined for each input using the field nSequence [78][2]. Lastly, in the following we denote Δ as the maximum time required for a transaction to be committed by a party and subsequently be included in the blockchain, which is critical in the **Dispute** phase of the channel (outlined in SectionIII-C1).

[2]Early protocols make use of a sequence number as a method of transaction replacement, a variant idea can be found in Eltoo protocol [79], where miners are supposed to include transactions with higher sequence numbers. However, the respective field in Bitcoin has been re-purposed to work as a relative timelock.

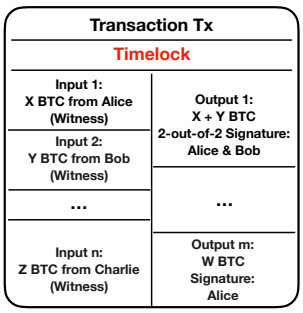


Fig. 1: A simplified illustration of a transaction, with n inputs (left side) and m outputs (right side).

B. Technical Challenges

As already emphasized earlier, the main goal of the channel is to increase the number of transactions while rendering the minimal number of interactions with the blockchain, *i.e.*, committed transactions.

As we will see later, in Section III-C1, most of the time the interaction, in a pairwise channel, happens only between the players in the channel. Therefore, the players need to keep the balance through all the transactions performed in the channel, in other words the protocol should offer *balance security*.

As we see next, very often, in order to have balance security, the nodes need to be online, in order to take action in the case the partner in the channel, misbehaves. In terms of privacy, a payment hub, *i.e.*, the node which controls several channels, a payer could also prefer that its payments are untraceable.

C. Channel Constructions

We start by a general description of a channel, then later we review the simplest forms of channels: *simplex* and *probabilistic simplex*. More complex forms are covered in the *duplex* and *state* channels sections.

1) General Description: The terminology is fuzzy since both terms, "payment" channel and "state" channel are used. Both terms refer to the idea of relying on off-chain direct communication between two players, *i.e.*, a *channel*. That is, initially, both parties open the channel by committing a certain amount of coins, which will be the channel capacity or the maximum amount that is passed from one player to the other.

The state of the channel can be tracked by simple signed transactions with or without the aid of *smart-contracts*, which we conveniently chose to denote them, respectively, *state channels* and *payment channels*. Although the choice may appear arbitrary, we observe that, with smart-contract capabilities, the term "state" seems more accurate given its generality. The scalability naturally derives from the fact that the only interaction with the blockchain is for opening, closing and disputing the channel.

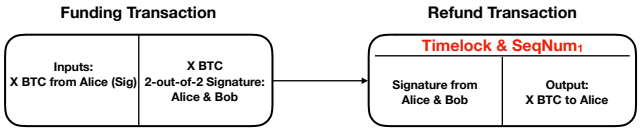
The simplest form of channel relies on instructions for timelocks, *e.g.*, the one described in BIP65 [80], and threshold signatures[3] using the early mentioned **CHECKMULTISIG** opcode. Typically channels have four phases:

- **Setup:** Two parties lock funds into the blockchain which the sum of funds is the capacity of the channel. It is important to note that the parties need to account for the confirmation time of the chain to start the channel.
- **Payment:** This phase the parties can exchange funds without interaction regardless of the confirmation time of the system. During this phase the initially locked funds cannot be used outside of the channel.
- **Dispute:** At any moment, a user can start a dispute in the case the other party do not follow the protocol.
- **Closing:** When the two players decide to close the channel, they publish transactions accordingly in the blockchain. These transactions reflect the exchange of funds that happened in the *Payment* Phase. Again, in this phase the players need to rely on the confirmation time of the blockchain. Furthermore, this phase can also trigger the **Dispute** Phase, whenever the parties do not agree on the current state.

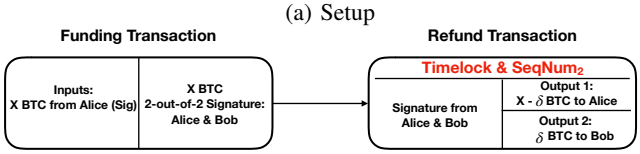
2) Typical Channel: The current balance of the channel is kept between the parties by exchanging directly mutually signed transactions with the amount being passed from one player to the other. When the channel is closed the last pair of exchanged transaction are committed to the blockchain, which makes the coins available for the players to redeem them. A safety mechanism is in place by locally keeping the signed transactions representing the new state of the channel. For example, for a channel between A and B, with capacity C_A and C_B, the coins committed, respectively, by A and B, after a transaction with value δ from A to B, therefore the new state is $C_A - \delta$, for A's balance and $C_B + \delta$ for B's balance. When a player does not correctly perform the protocol, *e.g.*, by not confirming a payment, thus the protocol enters in the *dispute* phase. During this phase, for [30], the partner can publish its local kept (and mutually signed) transaction claiming the coins of the channel. Alternatively, in Decker and Wattenhofer [29], the channel is closed in the previous state. The consistency of all signed transactions are assured to be correct due to the use of timelock operation code in the transaction verification script language [80]. In comparison, smart-contracts offer more complex operations. A contract, by design, can keep track of the balance and transactions made into it. Therefore it can execute payments accordingly based on previously set conditions. In particular, the execution of a protocol can be triggered by presenting proofs of correctness on secure multiparty protocols [73–75, 81, 82]. More formally, given witness that a protocol was correctly performed, the contract can be executed.

Next we detail the two functions *payment* and *state* in the Off-chain Channel Level.

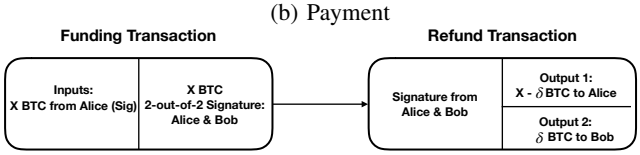
[3]Often denoted *multisig* by the Bitcoin community.



(a) Setup



(b) Payment



(c) Updated version of payment

Fig. 2: Figure 2a shows the initial state of a payment channel upon setup. Figures 2b and 2c show payments. The former makes use of sequence numbers and timelocks to replace previous payments.

3) Payment Channels: Here we describe the main works on payment channels, from simplex, and probabilistic, to duplex constructions. Next, we review state channel works. Finally, we present constructions for privacy preserving channels.

a) Simplex Payment: The earliest proposal of payment channels that we could find was posted on the Bitcoin Wikipedia [70] and is illustrated in Figure 2. Note that the first proposal mentioned here makes use of a sequence number to replace previous transactions, however, the respective, and already cited, Bitcoin field, nSequence has been disabled on 20th August 2010 [70, 78] [4].

Figure 2 illustrates a simplified form of the protocol in [70] between two parties: The payer and payee, and they initiate the *Setup Phase*. In order to do this the payer prepares two transactions as depicted in Figure 2a. The first is the so called Funding Transaction which takes as input funds from the payer and spends them within one output that requires the signatures of both payer and payee to spend. The second is the Refund Transaction which spends the output of the Funding Transaction and pays them back to the payer. Moreover the transaction has the sequence number set to 0, and a timelock in the near future by adjusting the value of nLocktime. Then, without signing the transactions, the payer sends both transactions to the payee who signs them and sends them back. After verifying the payee's signatures, the payer signs both transactions itself and sends them back. Then, one of the parties commits the Funding Transaction to the blockchain.

If the payer wants to pay a certain amount, it happens in the *Payment Phase*, it creates a new version of the Refund Transaction transaction that still spends the output of the funding transaction but instead of giving all funds back to the payer, it contains an output that allocates some funds to the payee as depicted in Figure 2b. Moreover, the sequence number is set to a higher value to replace the previous transaction. The payer signs this transaction and sends it to the payee. This process can be repeated for subsequent payments that all adjust the funds more and more in Payee's favor and increase the sequence number respectively.

Correctly keeping track of balances. The timelock and the sequence numbers prevent the payer to double-spending the funds. If the payer commits, by publishing it to the blockchain its Refund Transaction, created at Setup Phase, which gives all funds back to the payer, the payee could commit any Refund Transaction with a higher sequence number to replace Alice's committed transaction. However, the payee needs to commit his Refund Transaction before expiration of the timelock on payer's Refund Transaction. Since then the protocol has been adjusted on the Bitcoin-development mailing list [83] and the Bitcoin Wikipedia entry—[70].

In 2013 this protocol was updated, as illustrated in Figure 2c. This update removes the use of the sequence number. Moreover the timelock in the initial Refund Transaction is kept, however, updated Refund Transactions do not contain a timelock. Note that the payer only ever sees the initial Refund Transaction fully signed and therefore cannot commit any Refund Transaction without a timelock to the blockchain. As long as the payee commits its latest version of the Refund Transaction, such that it is included in the blockchain before expiration of the timelock on payer's Refund Transaction, he is assured to receive his funds.

Note that this construction is a simplex payment channel only, *i.e.*, only the payer can do payments to the payee, but not the other way around. To illustrate this assume the payee sends a payment to the payer, by having the Refund Transaction allocate more funds to the payer than to itself, compared to a previous Refund Transaction. Then payee can commit any transaction that allocates more funds to him resulting in a loss of funds for the payer.

b) Probabilistic Simplex: Pass et. al. [40] propose a probabilistic payment system in their work to reduce the number of transactions on the blockchain and subsequently reduce the amount of transaction fees enabling micropayments. Moreover their solution provides near-instantaneous payments without requiring confirmation delays. As such it is concurrent work to the LN [30]. Their work is based on the work of Wheeler [84], Rivest [85] as well as subsequent work in [86, 87]. [40] proposes three protocols. The first is a naive, therefore vulnerable, construction for two parties. The second and third protocols are similar, with both relying on verifiable third party with difference that third protocol is optimistic and utilizes the verifiable third party in case of a dispute.

In these protocols the payer sets up an escrow address and put an amount a of coins into it. After the payment is performed

[4]https://github.com/bitcoin/bitcoin/commit/05454818dc7ed92f577a1a1ef6798049f17a52e7#diff-118fcbaaba162ba17933c7893247df3aR522

the payer can potentially use the same escrow to pay another merchant after some time has passed. Moreover the payer sets up a second escrow in parallel such that in case of misbehavior the payment will be burnt by the verifiable third party. However, this approach has a couple of drawbacks, for example, feasibility. The user needs to pay into two escrow addresses, one for paying a merchant and one for punishment in case of misbehavior. The amount paid into the punishment escrow address should be significantly larger than the one paid into the escrow address for payment. The payer can only pay a fraction of the coins they possess at any time since a large amount of coins is locked in the punishment escrow address.

These protocols might be less feasible for small merchants or individual users who only occasionally execute payments. Another issue is that misbehaviour of the verifiable third party can only be observed but the party cannot be punished in any way. The verifiable third party can collude with the merchant to burn the payer's punishment money or it could collude with the user to withhold payment to the merchant. This would result in a loss of trust of the third party, however, there would not be any other drawbacks. It is important to note, that this work was improved by Hu and Zhang [41] using a time-locked deposit.

c) Duplex Payment: In a duplex payment channel both parties can allocate funds into their mutually shared channel and these funds can be redistributed between both parties arbitrarily. The challenge for the construction is to ensure balance security for all honest parties, *i.e.*, any honest party cannot lose funds in the presence of a malicious adversary and independent of the behavior of its counter-party. Both the constructions for duplex channels proposed by Decker et al. [29] and the LN [30] extend the idea of simplex payment channel[5]. That is, first, parties create a funding transactions in which both parties allocate their funds by paying coins into it. Moreover the funding transaction contains one output which requires signatures of both parties to spend. Second, the parties create a refund transaction which spends the funding transaction and has two outputs which pays back the coins to their respective parties. Note that each party needs to hold off signing the funding transaction until it holds a fully signed refund transaction since, otherwise, the other party might hold the funds in the funding transaction as hostage by committing the funding transaction and refusing to sign the refund transaction. After signing the refund transaction both parties sign the funding transaction and commit it to the blockchain to lock their funds into the channel, for the setup phase of the channel.

If the parties want to execute a payment, they create a new refund transaction and exchange signatures for it. The payment is considered executed when both parties hold the new fully signed refund transaction and invalidate all previous refund transactions. If both parties cooperate for channel closure the parties sign and commit a closing transaction that spends the funding transaction and pays each party the amount of coins that are denoted in the most recent refund transaction.

Dispute Phase differences. The way previous refund transactions are invalidated differs in [29] and [30]. While in [29], the dispute is solved using timelocks, to enforce the most recent mutually agreed status, in [30] it is done by punishing the party by giving all channel coins to the honest party. Briefly, the former structures channels within a, so-called, invalidation tree. The funding transaction represents the tree's root and refund transactions are the leaves of the tree. Each payment adds a node in form of a transaction into the tree. There are two methods that enforce that only one path in the tree, and therefore only one refund transaction, is valid. For one, each node in the tree has a relative timelock to its parent. A node in the tree can invalidate all of its siblings by setting its timelock to be Δ less than the minimal timelock on all of its siblings.

Note that refund transactions are always set to the highest locktime possible as agreed by both parties. The other method is to extend the tree's depth by replacing a refund transaction by timelock with an intermediate node that spends the previous node's output. This transaction has exactly one output that requires both party's signatures to be spend as a funding transaction. Then the new set of refund transactions spends this intermediate node. The maximum depth of the tree can be agreed upon before channel creation by both parties. The maximum width of the tree depends on the maximum agreed upon timelock since a replacing transaction needs to have a timelock that is at least Δ smaller then all of its siblings. In case of a dispute each party can commit the most recent path of the invalidation tree onto the blockchain to enforce the most recent refund transaction.

Punishing players. LN approaches invalidation of previous refund transactions by punishing the party that committed an invalid transaction by giving all their coins within the channel to the respective other party. Simplified, this is done as follows. Let Alice and Bob share a channel. First, to be able to punish a party we need to identify which party attempted to commit a possibly invalid transaction. This is done by having each party hold a version of the refund transaction that contains information that uniquely identifies them. So, when Alice and Bob want to execute a payment each of them creates a version of the new transaction that identifies the other party respectively, signs it and sends it to the other party. Then Alice holds the new refund transaction created by Bob that identifies herself, with Bobs signature and Bob holds the transaction of Alice. If Alice, or Bob, wants to submit the refund transaction they have to sign the transaction they hold, which was created by the other party, and commit it to the blockchain. They will not be able to commit the transaction created by themselves and would identify the other party since the other party does not return a signature for their transaction. By doing this we can identify the claiming party.

Next we need to add a method to punish a party in case they submit an invalid refund transaction. This is done by making

[5]A comparison can be found in [88].

the funds of the party that commits the transaction redeemable with two different conditions. In the following assume that Alice is the party that committed the refund transaction. First, in the case Alice is honest the funds are redeemable after a set time has expired using a timelock and with Alice's signature. The timelock is used to give Bob a timeframe in which to claim the output in case of Alice's wrongdoing. The second condition is redeemable with Bob's signature as well as a signature with a private key that Alice created for invalidating the refund transaction. Now to invalidate a refund transaction both parties exchange their respective invalidation keys after creating and exchanging the new payment transactions.

In [29] the amount of possible payments is limited by the maximum depth and timelock used for the invalidation tree, however, it has lower memory requirements compared to the LN since parties do only need to store the currently valid path of the invalidation tree. In contrast nodes in the LN need to store a set of keys linear to the number of payments executed in the payment channel. Moreover, for the construction of Decker et al., the number of transactions that need to be committed onto the blockchain in case of a dispute depends on the maximum depth of the invalidation tree, whereas the number of transactions committed in LN is constant. However, a limitation is that for both constructions nodes need to be online during the dispute, in order to enforce commitment of the currently valid path in the invalidation tree or punishment transaction, respectively.

Hardware based approaches. Alternatives which relies on secure hardware, *i.e.*, trusted execution environments (TEE), exist. Teechan is a protocol introduced by Lind et al. [68] which introduces full duplex payment channel framework between two mutually distrusting parties, assuming that they are equipped with TEEs. In their implementation on the Bitcoin network, they utilizes Intel's Software Guard Extensions (SGX), which provides secure region of memory, *i.e.*, a *enclave*. By making only specific SGX instructions able to execute a code or access to a data in an enclave, confidentiality and integrity for the enclave code and data are guaranteed. Moreover SGX supports *remote attestation* which enables an enclave to acquire a proof that it is executing a particular enclave code. Both private keys of the two parties interacting in Teechan are securely shared by their TEEs. In execution, only each TEE generates every transaction, encrypts it and then gives it to the party. Therefore each party is only responsible to the setup, payment or close requests for its TEE, and forwards them to the counterpart. Lind et al. [68] measured 10 million transactions to be exchanged in optimal network conditions,*i.e.*, network bandwidth and latency, and observed that the protocol achieves 2,480 TPS on average.

A similar work to [68] is given by Takahashi and Otsuka [72] which improves the work of Dmitrienko et al. [71] by selecting a different technique to validate the funds of a hardware secure wallet. Both [71] and [72] relies on loading funds in advance into the wallet. However, whereas in [71] the payer wallet verifies the pre-loaded funds, in [72], the payee performs the verification. In [72], differently from [68], it is not needed the signature of both sides of the channel, *i.e.*, 2-out-of-2 signature between payer and payee.

Channel factories. The general framework for channels, as described in Section III-C1, imposes restrictions on the creation of channels. Namely, the confirmation time for the **Setup** Phase may still be unacceptable for micropayments, without in advance funding transaction. Furthermore, upon channel creation the funds are locked for only that channel. The relaxation in these requirements is the motivation for the creation of *channel factories* proposed by Burchert et al. [69].

The core idea is to introduce a layer between the blockchain and the payments. This translates into a step where a group of collaborators jointly fund a factory. This first step, still requires blockchain interaction, therefore still is subject to the limitations of the consensus algorithm. However any new pairwise channel can be created, among the initial users, from this point, upon communication between the collaborators, hence creating channels. Although the factory creation still requires time and funds locking into the blockchain, the advantage of this design is that it allows reallocation of funds between the pairwise channels.

Embedding script into signature. An interesting idea to preserve space in the blockchains, which is the case for major cryptocurrencies, is to embed more instructions into the signature issuing procedure by relying on a more sophisticated, *i.e.*, one that offers more properties, signature scheme. That is the approach introduced by Poelstra [89, 90] denoted *scriptless script* for the Schnorr signatures and LN, which was recently extended by Malavolta et al. [91] to the more suitable, for cryptocurrencies, ECDSA signature scheme.

Briefly, [89] outlines an approach which relies in the Schnorr signature scheme in order to embed the value of the pre-image x, *i.e.*, for the HTLC hash challenge, into the signature computation algorithm, jointly carried by the payer and the payee of the channel. Although the formalization effort displayed in [91], the authors point out it is not the case for other proposals. Furthermore, main cryptocurrencies, *e.g.*, Bitcoin and Ethereum, may not be compatible to the Schnorr scheme.

Privacy preserving channels. An anonymity focused approach has been proposed by Green et. al. [67], named BOLT. It assumes anonymity of the underlying blockchain, *i.e.*, either use an anonymous blockchain as Zcash [92] or means to anonymize it, *e.g.*, by using a mixer. Then, assuming that one of the parties is well known, *e.g.*, it is a merchant, it provides additional anonymity guarantees for the respectively other party, *e.g.*, a customer. The merchant knows that they received a payment, however, does not know by which customer the payment has been done.

Their duplex payment channel proposal extends the Compact Ecash protoool by Camenisch et al. [93] which is based on the work by Chaum [94]. However their construction does

not consider a definition of balance security when a malicious customer stop cooperating with the merchant after doing a couple of payments [6]. As the customer is required to start channel closure the merchant, *i.e.*, one of the participants of the channel, cannot close it arbitrarily, therefore it cannot claim the funds payed to him. More recently Zhang et al. [64] introduced Z-Channel, which provides anonymity for micropayments adapting the ZeroCash. However it is uncertain if [64] is suitable for payment networks. Similarly BOLT (Blind Off-chain Lightweight Transaction) [67] tackles the *linkability* problem in channels, however [67] does not seem to provide a construction over ZeroCash [95].

Another protocol that minimizes the payer information leakage is TumbleBit [60], which presents an unidirectional unlinkable payment channel, where payer pay to payee via an untrusted payment hub (called Tumbler) that plays a role of a mixer. In TumbleBit, unlinkability is achieved for Tumbler itself. Moreover, the protocol prevents a malicious users theft any other user's funds without any trust. It was also thought to be one of the remarkable features that the protocol is compatible to the Bitcoin protocol before SegWit is adopted, the most recent update in the Bitcoin protocol.

As with the conventional micropayment channels, TumbleBit's users require a channel to be established in advance. Then payer can send payments as much as they like within its deposit. The protocol consists of the three phases: Escrow, Payment and Cash-Out. It needs four on-chain transactions that one pair of payer and payee opens a channel in the Escrow Phase, communicates to transfer multiple unidirectional off-chain payments in the Payment Phase and finally settles the channel in the Cash-Out Phase.

When payer Alice transfers a payment to payee Bob, the overview of the protocol is as follows. First, Alice deposits to the Tumbler and the Tumbler does to Bob. The Tumbler encrypts transactions as *puzzles*, by which Bob can claim the deposit, and sends them to Bob. TumbleBit core is its sub-protocol *puzzle-solver protocol*, where Alice receives the solution to her single puzzle only if she transfers her single payment to the Tumbler. In the Payment Phase, Alice receives a single puzzle from Bob, engages in the puzzle-solver protocol together with the Tumbler and then get the solution to the puzzle and back the solution to Bob. Note that Bob blinds the puzzle before sending it to Alice, thus It makes sure that the Tumbler cannot link Alice and Bob.

4) State Channels: Until now, in this work, the channels are built based on special instructions for transactions, as described in Section III-A. Smart contracts, as in Ethereum, enable more complex offchain structures which has been investigated in [49, 62][7]. A particular use for smart-contracts aims to enforce fairness and correctness on distributed systems.

[6]Although the authors acknowledge it by explicitly assuming the customer follows the protocol.

[7]We discuss [62] and [49] in more detail in Section IV.

The approach of relying in penalties to guarantee correctness and fairness of computation, in particular, on *secure multiparty computation protocols*, has been independently explored for efficient protocols on the top of blockchain. For the general case, this idea has been introduced by Andrychowics et al. [96, 97], and later by Bentov and Kumaresan [98]. For specific purposes, *e.g.*, card-games, it was further explored by Bentov et al. [73] and David et al. [74–76]. The main difference from the first works and the specific purposes protocols is the heavy use of smart-contracts. In particular, to arbitrage disputes among the players of the protocol.

More specifically, smart contracts can be used to store the state of the channel other than just the fund distribution between two parties. Since it is implemented with a Turing complete programming language, it can redistribute funds according to arbitrary states upon closure of the channel. Compared to the previously mentioned constructions state channel can be implemented more straight forwardly. They are opened by committing the respective smart contract onto the block chain, whose state can be changed using a message signed by both parties as well as a sequence number. Parties then can change the state offchain by computing a message to the smart contract that would make it transition into that state including a sequence number and exchanged signatures for this message but without committing the message onto the blockchain. The smart contract can be closed by sending to it the latest message in order to initiate the dispute or close phase, in which one of the parties sends a message with higher sequence number. After that, the smart contract transitions into the state as specified by that message.

NoCUST [65] is another protocol which relies heavily on smart-contracts. Here two parties wishing to exchange small amounts create a channel by making their deposits into a smart-contract, therefore two on-chain operations. All the payments from that point on, are executed off-chain via a third trusted node which intermediates the off-chain operations between the two participants, and each payment requires issuing request payments and receipts. This design has the advantage, in comparison to regular payment hub over off-chain channels, of not requiring the hub node to allocate/lock a large amount of funds, hence *no custodian*, while intermediating payments between large group of pairs of nodes.

IV. Network Level

In this level the concern is how the payment network can be established, in contrast to how it is managed (which is the topic of Section V).

A. Technical Challenges

The main function of this level is the construction of a payment route. More concretely, how the pairwise "inner" channels can be concatenated and used to provide a medium for payments between two nodes that are more than one hop apart. Within this setting, it is important to keep, likewise Section III, *balance security*, specially in the inner nodes. The technical

challenge is significantly bigger than in the early sections, because in the network cases, other nodes in the route could also collude against the honest player.

B. Construction Function

Pairwise payment channels can be concatenated in several nodes to form a *payment network* which can enhance even more the scalability of a system, because two nodes do not necessarily contact each other to open a channel. Instead, they can create a "virtual channel", it the sense of Dziembowski et al. [99] via a mutually connected node. The core technique in these networks is the early cited HTLC [30].

This enhances the scalability of a system even more since it removes the need to create new payment channels between each pair of payer and payee. The concept of PCN and related concepts had been previously studied: trust networks [100–102], credit network [103], path-based transaction (PBT) [104] and privacy in PCN [59]. The technique in [30] is currently being used in the Bitcoin network and paved the way to new propositions for different cryptocurrencies, as in the Raiden Network [66] for the Ethereum [8].

HTLC and conditional transfers. Nodes can execute a payment atomically on a set of channels using HTLC. A payment can be routed across a sequence of channels without payer and payee having to create a channel between themselves that would require committing transactions onto the blockchain. Briefly, in [30], two nodes exchange funds by relying on a middle node, say, B, and the two channels between A and B, and B and C. Node A wants to send coins to C, and, as the first step, C computes a hash value on an arbitrarily chosen random number x and shares it with A via a direct channel, however without establishing a payment channel. The actual transaction is carried by A sending the funds to B, using their mutual channel with the extra condition that node B shows the secret value x. Analogously, node B handles the funds do C in a similar fashion, relying in their mutual channel.

Apart from LN based channels [30], several types of channel constructions, from Section III, support HTLC, *e.g.*, Sprites [49]. They are: Raiden [66], Decker et. al. [29], and Perun [62]. Given the importance of the HTLC and applications on PCN, it is very surprising that apparently there is still no formalization of it in the literature.

Mitigating Dispute Phase costs. In general the payment network protocols are *optimistic*, *i.e.*, they assume the nodes will cooperate, therefore the dispute phase, as outline in Section III-C1, is expected to be rarely triggered. However this phase execution can be costly both in terms of time complexity and financially given that the arbitrage can involve the execution of smart-contract to compute penalties. The Sprites Protocol [49] introduced by Miler et al. is designed for Ethereum's smart contracts, and has the goal of reducing of the time complexity of resolving a dispute. Note that this work is independently done in Raiden [66]. The main observation of [49] is that in the dispute case for the multichannel route, any of the internal pairwise channel will have to solve the dispute in its own respective HTLC agreement, *i.e.*, between each of the two consecutive. Hence, [49] substitutes them for a Ethereum smart-contract which solves the dispute. That differs from the HTLC technique since there is no need to increment the time on each hop of the payment route.

A recent idea introduced by Buckland and McCorry named *State Assertion Channels* [61], analogously to [49], but for computational costs. The goal of [61] is to guarantee that during the Dispute Phase, which can also be triggered by closing a channel, the honest party always can be paid back regardless of the cost of performing the verification of the full state of the application in the blockchain deployed smart contract. What Sprites [49] does for the expiration time of channels, the State Assertion technique does for the cost of computation, which indeed can be expensive if done on chain.

The main setting for state assertion channels is to deploy two smart contracts, the application (AC) and the assertion (SC) ones, with the requirement that the latter has to receive the funds before the creation of the channel itself. The states are not disputed in the AC, but in SC which can pay back the honest user whenever it challenge an invalid state transition proposed by the cheater user. The crucial observation is that SC verifies without the *full state* of the application, but, instead, with a digest of it via hash values. The work in [61] provides a simulation on this novel approach, and it is a first step in the rigorous formalization of the idea.

Generalizing State Channels. There are approaches to construct more general structures over offchain-channels. More notably, we can find in the literature by Dziembowski et al. [99] and Coleman et al. [63]. The work in [99] extends their virtual state channel construction in [62] by enabling virtual payment channels exceeding one intermediary party by recursively applying their construction on top of state channel as well as virtual state channel. The formal treatment of "virtual channel" [62], which allows the creation of a multi-hop state channel across a sequence of single-hop state channels. This technique allows payer and payee to operate on a shared state channel instead of having to setup a new HTLC instance for each individual payment. More concretely, consider parties Alice, Bob and Charles where Alice and Charles, as well as Charles and Bob share a state channel respectively.

For comparison, in [30] and [29], Alice and Bob can execute payments between each other making use of the existing infrastructure and without having to create a new payment channel between Alice and Bob. However, this requires that Charles participates at the payment protocols for each payment between Alice and Bob. In contrast, virtual payment channels allow the creation of a payment channel on the top of the existing state channel, adding a layer of indirection and enabling payments between Alice and Bob without Charles' participation outside of virtual channel setup, closure and any dispute. The protocol is proven in the UC framework [105],

and requires smart contract capabilities[8] and therefore it is not applicable to cryptocurrencies with limited smart contracts capabilities as Bitcoin.

In a sense, Coleman et al. [63] also aims to build a richer structure on the top of the off-chain channels. They introduced the notion of *counterfactual*, which is, in a nutshell, all the events of the channel, which can, or cannot, be committed to the blockchain. In this paradigm, a payment in the off-chain channel, changes a so-called *counterfactual state* of the system. In addition, it is also possible to create *counterfactual contracts* via commitments and signatures, which generalizes the channels even further. The work in [63] presents an framework, focused on practical implementation.

The Wormhole attack on PCNs. The attack introduced by Malavolta et al. [91] affects all the PCN constructions based on 2-step interaction between the nodes, therefore HTLC based PCNs are in general, vulnerable to it. The colluding nodes aim capture the transaction fee which the inner nodes on a payment route would expect to receive in order to carry the payments in the network. Although the gravity of the attack, the authors of [91] succeeded in proposing a fix for the main protocol and even warned the LN team about the protocol weakness, which triggered new developments. In particular the implementation of the two party ECDSA construction from [91], and its later incorporation into the system [106].

Briefly, the attack is carried by two colluding nodes within the same payment route. The two nodes share the pre-image value of the HTLC, while keeping the nodes in between them oblivious to the payment condition value. Consequently, in the point of view of the nodes in the extremities of the path, the payment is carried out, while the nods in the middle of the path see as it has failed. This condition allows the colluding nodes to jointly receive the transactions fee from the nodes which did not executed the payment.

V. Network Management Level

This is the upmost level in our classification, and, arguably, this level offers a greater variety of technical issues in comparison to the other levels. Thus, before reviewing the existing protocols, we described the known technical problems.

A. Technical Challenges

Here we assume that a network of channels is established, and all the nodes have access to it. Therefore we identify the following main functions:

1) Routing: Similarly to a regular computer network, in order the find a payment path, it is necessary to probe the network for nodes available to route the payments. On the other hand, in the case of PCN, other variables can influence the construction of such paths, in particular availability and *fees* to intermediate the payments can heavily influence the routing. A close look into routing in the LN is given by Di Stasi et al. [107] and McCorry et al. [77]. A more general desiderata for routing over PCN is given by Hoenisch and Weber [48] based on similarities between PCN and Mobile Ad Hoc Networks (MANET).

2) Re-balancing route: Each pairwise channel has associated with it a capacity, which is how much funds the channel can handle. A PCN can concatenate several channels, *i.e.*, different capacities, that need to harmoniously work together. The consequence is that inner channels in a path have its capacity exhausted forming bottlenecks in the whole route.

3) Stability of the route: The current techniques for channel constructions very often rely on the assumption that the users of the channel, and therefore of a PCN, will be online during all the lifespan of the channel. This is crucial in the case a Dispute Phase, as described in Section III-C1, when the parties need to act timely. Failing to claim the correct state of the channel in this phase, lead to the honest user to lose funds.

4) Privacy and anonymity: Intermediate nodes within a payment route, in principle, watch all the flow of transactions, since they are intermediating all the payments. Moreover, when enabling a payment path through, for example routing, information may be leaked about the payer and the payee.

B. Constructions

For each of the technical challenges detailed earlier, we now detail and discuss the existing approaches in the literature.

1) Routing: Here the challenges are similar to computer networks, however there are important differences due to the payment network nature. For one, routing should be scalable, *i.e.*, both the stored state per node as well as communication complexity when routing should be logarithmic on the number of nodes. Moreover, the amount of funds that can be routed through a path depends on the capacity of the channel with the lowest capacity on the path. Furthermore, nodes within a path can be either controlled by a malicious adversary or spontaneously go offline. Also the network should be able to handle multiple concurrent routing attempts and payments. However, on a different note, since individual transaction are not recorded on the blockchain payments through an offchain channel network can provide better privacy and anonymity guarantees compared to the underlying blockchain if done right. In addition to all these challenges intermediate routes that participate in forwarding a payment can ask for a fee and therefore nodes need to consider this and might want to optimize for a reduction of payment fees. Nevertheless, we note that as a fallback nodes can always create a new payment channel between payer and payee, however, this counteracts the scalability efforts of offchain payment networks so most approaches attempt to accomplish routing without relying on the fallback method.

We would like to note that the above stated challenges are similar to those in wireless sensor networks. There nodes

[8] A proof-of-concept implemented for Ethereum is available at https://github.com/PERUNnetwork/Perun.

often have limited resources in form of limited battery which results in communication being expensive. As a consequence communication over long routes is a burden on the whole networks similar to long payments allocating a lot of collateral which [49] defines as the total amount of funds allocated in a HTLC across a payment path. Communication overhead needs to stay scalable to avoid nodes running out of battery. Moreover, nodes in wireless sensor networks have limited storage such that the state stored per node needs to be scalable as well. Since the deployment of nodes in wireless sensor networks can be non-structured nodes need to setup networks among themselves and moreover might need to adjust their routing tables often due to changes in the environment as well as nodes that run out of battery and go offline. However, privacy requirements are different in wireless sensor networks compared to offchain channel networks as the receiver of messages is often a designated sink such that most related work focus on sender anonymity only.

Approaches for routing. A common limitation for long payment channels is that the intermediate nodes need to be online during the transaction period. Therefore, often routing algorithms rely on *landmark routing techniques* [108], where a *landmark*, *i.e.*, a node with extra guarantees regarding its connectivity. This node is used as the middle node in the path.

Briefly, there are two ways to approach routing in these networks. One is *landmark routing* where all nodes know the route to a set of *landmarks*. In such a protocol payments are, first, routed from the payer to a landmark and, second, routed from the landmark to the payee. This approach promotes centralization of payment channel networks. Nodes close to a landmark would have the advantage of being better connected and might need to pay less fees for routing as well as have a better routing success rate. They also might get more income through fees because payments would more often be routed through them compared to nodes far off from landmarks. Another approach is routing using *embeddings*. In this approach nodes among themselves decide on an address space of the network and, then they can find routes using these addresses. This approach enables more decentralization however it requires more communication for maintaining routing tables.

SilentWhispers Protocol [51] proposes using landmark routing, where all landmarks are publicly known and semi-honest. All routing is done through these landmarks by computing the shortest routes. This is done in epochs such that node's routing tables can update to changes within the network. The work [51] also defines privacy notions for credit networks and provides security proofs of their construction in the UC framework. The work also proposes multiple extensions. One of these allows for malicious landmarks whereas the other extension proposes routing through nodes that are offline. In the latter case nodes create secret shares of their long-term private keys and distribute those across the landmarks who can then impersonate offline parties, however, this requires that at least one landmark is honest and does not try to reconstruct long-term private keys of which it holds secret shares.

Similarly, the SpeedyMurmurs routing protocol [52] an embedding based routing protocol, for path-based transaction networks based on the work in [109]. In [52] some nodes are designated landmarks. Using these landmarks as roots of respective spanning trees are computed to create an address space within the network. A route can be computed using a distance function on nodes' addresses to find the next hop on route to the target. In comparison, LN/HTLC [30] relies on a *gossip protocol* for channel maintenance and recovery [9] and onion style routing for privacy [10]. However it is not clear how nodes can find routes in the network. An attempt to address this issue is the routing proposal algorithm for LN: Flare [54].

The protocol in [54] is a probabilistic, and proceeds in two ways: (1) each node stores the topology of its neighborhood, (2) each node chooses a set of beacon nodes globally and at random according to a uniform distribution. Routing between two nodes is done by first checking both peers neighborhoods for intersections, and if this does not work checking whether a beacon node is within the other peers neighborhood. The nodes continue by using the neighborhoods of a few beacon nodes when searching for mutually known nodes to route through.

The author's of [51] evaluate their work using Ripple's [110] payment network topology from 2013 and 2016 [111]. They compare success ratio, path length of payments and message complexity between their approach, a protocol based on the Ford-Fulkerson max-flow algorithm [112], SilentWhispers [51] among others. They show a better success ratio as well as smaller paths in average compared to SilentWhispers, however, a lower success ratio than the Ford-Fulkerson based protocol which had an unfeasible message complexity.

Hoenisch and Weber [48] pioneered the adaption of techniques from Mobile Ad Hoc Network (MANET). In [48] they compile a list of requirements and also argue that techniques from MANET, like the one they rely on for their protocol, which is named On-Demand Distance Vector (AODV), are more suitable to dealing with the issues of payment channel networks. An interesting feature of their adapted protocol is that it takes into account the balance and fees of intermediate nodes, a feature not present in MANET but which is highly relevant in payment networks, given that these values can change arbitrarily and without coordination.

Splitting payments. Moreover payments can be split-up into multiple smaller payments and routed through multiple routes [56]. It is important to highlight that [56] also considers privacy as well as sender/receiver anonymity. Routing success does not only rely on whether a route can be found or not, but also whether the route has enough capacity for the payment

[9] A description can be found in https://github.com/lightningnetwork/lightning-rfc/blob/master/07-routing-gossip.md

[10] A description can be found in https://github.com/lightningnetwork/lightning-rfc/blob/master/04-onion-routing.md

to be executed through it. Large payments have low routing success rates [11]. Approaches to tackle this issue attempt to split payments through multiple routes as AMP [56] that provides a protocol to split a payment across multiple paths, similarly to the Split Payment Protocol [55], however [55] allows for payments to complete only partially. Another approach is the Spider Network [53] which employs packet-based routing including congestion control to do payments through multiple paths. Each payment is split-up into smaller packets which can be routed through separate routes and claimed atomically by the sender as proposed in [56] or non-atomically. Nodes involved in routing payments can employ congestion control techniques when multiple payments are routed concurrently. Channel capacities and possible imbalances are considered when selecting paths for packages.

The work in [53] compares the success ratio of routing using an approach based on [112] as well as SpeedyMurmurs [52] and SilentWhispers [51]. They evaluated these approaches using, for one, network topologies existing in the Internet as found in [113] as well as Ripple's [110] payment network topology from 2013 [111] which has also been used for the work on SpeedyMurmurs [52]. The evaluation shows that the approach proposed in this paper works better on the former topologies than the latter ones where they are outperformed by the algorithm from [112]. Noteworthy is that they report significantly lower success ratios for SpeedyMurmur than have been reported in their paper [52]. Moreover, Speedy Murmurs does not consider privacy notions or malicious adversaries.

2) Channel Re-balancing: Given the limited capacity of a channel, it may be necessary to rebalance a set of payment channel to avoid that the channel turns into a simplex one because of missing capacity on one side. For instance, let Alice and Charlie, Charlie and Bob as well as Charlie and Alice share a payment channel respectively. Moreover assume that initially everyone has one coin allocated as their funds for each channel. Now, if Alice sends one coin to Bob, Bob sends one coin to Charlie and Charlie sends one coin to Alice, all within their respective channels then we would end up in a situation where in each channel one party has two coins and the other has no coin. In this case we would not be able to repeat this payment without routing the payments over an intermediate node even though the total funds of all nodes remain unchanged. To remove this skewness in funds the REVIVE protocol [50] relies on an untrusted third party that creates a block of transactions rebalancing funds. Each peer will lose money on one or more payment channels but gain an equal amount on others. After each peer verifies this on the rebalancing transactions they can apply the changes off-chain.

3) Channel Stability: The channel between two nodes assume that both stay online on the duration of the channel. Any misbehaviour of any node, can trigger the response and respective claim of the partner node, which have to be online in order to perform the dispute phase of the network protocol. An approach to address this is to assume that the receiver can assign a *custodian node*, sometimes also referred as *watch-tower* [58, 114], to watch over their payment channel while they are offline. Similarly, this is the idea of the PISA [57]. The custodian can dispute malicious commitments of the other peer while its customer is offline. The custodian gets a fee for the service and has a deposit that gets destroyed when it fails to make a dispute on behalf of its customer. However, the custodian's customer are not public and the custodian can use the same deposit as a safety for all customers. This leads to the problem that if there are enough customers the custodian can possibly get more funds by cheating on its customers than it would lose by having its deposit destroyed. Moreover, if the deposit is not destroyed and instead is payed out to all customers a malicious custodian can create customers for itself to mitigate the funds lost.

4) Privacy and Anonymity: The network management layer may leak crucial pieces of information from nodes belonging to a route to the origin node of a payment. Malavolta et al. [59] investigate privacy notions on PCN and how to enforce them, and propose routing protocols Fulgor and Rayo [59] that considers concurrent routing processes and attempt to avoid deadlocks in a sense that at least one payment completes. Unfortunately their privacy notions rely on assumptions that the underlying routing protocol does not store the state (capacity) of payment channels within the network, which if done could be used to circumvent their attempts for privacy.

Security on routing. Flare [54] requires inquiring the capacity of channels routed through and therefore cannot be used for finding routes with their work to uphold privacy, however the authors specifically state Flare routing as an option to use with their work. Moreover, they require a way of ordering of transactions for their protocols to prioritize which payments to execute when resolving deadlocks. However, finding such an ordering in a way that does not enable abuse is not trivial. The authors suggest the possibility of using the transactions hash, however, this could be manipulated by a malicious adversary by slightly altering the payment to get a hash that results in a higher priority for their payment when going through the network. Their constructions is proven secure in the UC framework [105], however, their proof for two users who sharing a payment channel of agreeing on the state of the channel only works in the semi-honest adversary setting.

Fees and node reputation. Since each node in the network can charge an arbitrary fee in order to conduct the payment along the path, an interesting question to investigate is the game-theory behavior of rational nodes within the network and other economic questions. In comparison to other works, as seen in previous transactions, these questions seem to be, to this day, largely understudied. A framework to study network topologies and economic aspects is given by Brânzei et al. [115]. Similarly, a reputation system, which would take account of fees and success payment rate, seems to be missing in the literature and are highly relevant for practical systems.

[11] Reported in https://diar.co/volume-2-issue-25/ and https://thenextweb.com/hardfork/2018/06/26/lighting-network-transactions/

VI. Final Remarks

We provided a major review, as shown in Table I, of the body of work on PCNs and related protocols for Layer-2 solutions, an emerging area targeting scalability of cryptocurrencies. We believe it provides a significant value for the community because it contains a wide overview of the landscape on the ideas and approaches present in the scientific literature and Internet repository and forums.

We highlight the rich work on routing protocols, for Network Management Level, in comparison to attention given to, for example, assure that the channels are properly balanced, *i.e.*, Re-Balancing. That can be explained by the similarities of PCN and computer networks, which can be seen as a source of already testes ideas to find routes. Similarly to state channel research, within the Off-Chain Level in Table I.

More recently, we can also note a trend on network construction protocols for the Network Level, in particular the works on Virtual Channels [62] and Counterfactual [63]. In both cases, they distinguish themselves by increasing the complexity of the constructions on the top of the off-chain channels. Surprisingly, the more extablished technique, *i.e.*, HTLC, does not seem to have a formal treatment yet in the literature, which can interest the research community.

Along the same lines, works on the economic aspects of the channels, *i.e.*, fees charged by nodes, and *reputation* of nodes seem also to be under reported in the literature. In both cases, they are important, because they are critical to pave the way of more practical systems. Finally, it is important to emphasise that this work left out the topic of *interoperability* of systems, *e.g.*, Cosmo [116], XClaim [117] and Comit [118].

References

[1] D. Chaum, "Blind signature system," in *Advances in Cryptology – CRYPTO'83*, D. Chaum, Ed. Santa Barbara, CA, USA: Plenum Press, New York, USA, 1983, p. 153.

[2] D. Chaum, A. Fiat, and M. Naor, "Untraceable electronic cash," in *Advances in Cryptology – CRYPTO'88*, ser. Lecture Notes in Computer Science, S. Goldwasser, Ed., vol. 403. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 21–25, 1990, pp. 319–327.

[3] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[4] EOS, "EOS," https://eos.io/, 2018, [Online; accessed 6-May-2018].

[5] Steemit, "Steemit," https://steemit.com/, 2018, [Online; accessed 6-May-2018].

[6] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," Cryptology ePrint Archive, Report 2017/454, 2017, http://eprint.iacr.org/2017/454.

[7] C. Foundation, "Cardano Hub," https://www.cardano.org/, 2018, [Online; accessed 28-March-2018].

[8] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014.

[9] V. Network, "Visa Network," https://www.visa.ca/, 2018, [Online; accessed 17-October-2018].

[10] P. Network, "Paypal Network," https://www.paypal.com/jp/home, 2018, [Online; accessed 17-October-2018].

[11] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Advances in Cryptology – CRYPTO 2017, Part I*, ser. Lecture Notes in Computer Science, J. Katz and H. Shacham, Eds., vol. 10401. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 20–24, 2017, pp. 357–388.

[12] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-ng: A scalable blockchain protocol," in *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation*, ser. NSDI'16. Berkeley, CA, USA: USENIX Association, 2016, pp. 45–59. [Online]. Available: http://dl.acm.org/citation.cfm?id=2930611.2930615

[13] Y. Sompolinsky and A. Zohar, "Accelerating Bitcoin's transaction processing. Fast money grows on trees, not chains," Cryptology ePrint Archive, Report 2013/881, 2013, http://eprint.iacr.org/2013/881.

[14] Y. Lewenberg, Y. Sompolinsky, and A. Zohar, "Inclusive block chain protocols," in *FC 2015: 19th International Conference on Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, R. Böhme and T. Okamoto, Eds., vol. 8975. San Juan, Puerto Rico: Springer, Heidelberg, Germany, Jan. 26–30, 2015, pp. 528–547.

[15] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "SPECTRE: A fast and scalable cryptocurrency protocol," Cryptology ePrint Archive, Report 2016/1159, 2016, http://eprint.iacr.org/2016/1159.

[16] Y. Sompolinsky and A. Zohar, "PHANTOM: A scalable BlockDAG protocol," Cryptology ePrint Archive, Report 2018/104, 2018, https://eprint.iacr.org/2018/104.

[17] B. Wiki, "Block size limit controversy," https://en.bitcoin.it/wiki/Block_size_limit_controversy, 2018, [Online; accessed 17-October-2018].

[18] D. Boneh, M. Drijvers, and G. Neven, "Compact multi-signatures for smaller blockchains," Cryptology ePrint Archive, Report 2018/483, 2018, https://eprint.iacr.org/2018/483.

[19] J. Chen, S. Gorbunov, S. Micali, and G. Vlachos, "ALGORAND AGREEMENT: Super fast and partition resilient byzantine agreement," Cryptology ePrint Archive, Report 2018/377, 2018, https://eprint.iacr.org/2018/377.

[20] I. Bentov, R. Pass, and E. Shi, "Snow white: Provably secure proofs of stake," Cryptology ePrint Archive, Report 2016/919, 2016, http://eprint.iacr.org/2016/919.

[21] R. Pass and E. Shi, "Thunderella: Blockchains with optimistic instant confirmation," in *Advances in Cryptology – EUROCRYPT 2018, Part II*, ser. Lecture Notes in Computer Science, J. B. Nielsen and V. Rijmen, Eds., vol. 10821. Tel Aviv, Israel: Springer, Heidelberg, Germany, Apr. 29 – May 3, 2018, pp. 3–33.

[22] ——, "The sleepy model of consensus," in *Advances in Cryptology – ASIACRYPT 2017, Part II*, ser. Lecture Notes in Computer Science, T. Takagi and T. Peyrin, Eds., vol. 10625. Hong Kong, China: Springer, Heidelberg, Germany, Dec. 3–7, 2017, pp. 380–409.

[23] U. Klarman, S. Basu, A. Kuzmanovic, and E. G. Sirer, "bloxroute: A scalable trustless blockchain distribution network whitepaper."

[24] J. Garay and A. Kiayias, "Sok: A consensus taxonomy in the blockchain era," Cryptology ePrint Archive, Report 2018/754, 2018, https://eprint.iacr.org/2018/754.

[25] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: Scaling blockchain via full sharding," in *ACM CCS 2018: 25th Conference on Computer and Communications Security*, D. Lie, M. Mannan, M. Backes, and X. Wang, Eds. Toronto, ON, Canada: ACM Press, Oct. 15–19, 2018, pp. 931–948.

[26] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds. Vienna, Austria: ACM Press, Oct. 24–28, 2016, pp. 17–30.

[27] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger: A secure, scale-out, decentralized ledger via sharding," in *2018 IEEE Symposium on Security and Privacy*. San Francisco, CA, USA: IEEE Computer Society Press, May 21–23, 2018, pp. 583–598.

[28] D. Mazieres, "The stellar consensus protocol: A federated model for internet-level consensus," *Stellar Development Foundation*, 2015.

[29] C. Decker and R. Wattenhofer, "A fast and scalable payment network with bitcoin duplex micropayment channels," in *Symposium on Self-Stabilizing Systems*. Springer, 2015, pp. 3–18.

[30] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," *See https://lightning. network/lightning-network-paper. pdf*, 2016.

[31] D. Wheeler, "Transactions using bets," in *Security Protocols*, M. Lomas, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 89–92.

[32] R. J. Lipton and R. Ostrovsky, "Micropayments via efficient coin-flipping," in *FC'98: 2nd International Conference on Financial Cryptography*, ser. Lecture Notes in Computer Science, R. Hirschfeld, Ed., vol. 1465. Anguilla, British West Indies: Springer, Heidelberg, Germany, Feb. 23–25, 1998, pp. 1–15.

[33] S. Micali and R. L. Rivest, "Micropayments revisited," in *Topics in Cryptology – CT-RSA 2002*, ser. Lecture Notes in Computer Science, B. Preneel, Ed., vol. 2271. San Jose, CA, USA: Springer, Heidelberg, Germany, Feb. 18–22, 2002, pp. 149–163.

[34] R. L. Rivest, "Electronic lottery tickets as micropayments," in *FC'97: 1st International Conference on Financial Cryptography*, ser. Lecture Notes in Computer Science, R. Hirschfeld, Ed., vol. 1318. Anguilla, British West Indies: Springer, Heidelberg, Germany, Feb. 24–28, 1997, pp. 307–314.

[35] S. N. Foley, "Using trust management to support transferable hash-based micropayments," in *FC 2003: 7th International Conference on Financial Cryptography*, ser. Lecture Notes in Computer Science, R. Wright, Ed., vol. 2742. Guadeloupe, French West Indies: Springer, Heidelberg, Germany, Jan. 27–30, 2003, pp. 1–14.

[36] N. van Someren, A. M. Odlyzko, R. L. Rivest, T. Jones, and D. Goldie-Scot, "Does anyone really need micropayments?" in *FC 2003: 7th International Conference on Financial Cryptography*, ser. Lecture Notes in Computer Science, R. Wright, Ed., vol. 2742. Guadeloupe, French West Indies: Springer, Heidelberg, Germany, Jan. 27–30, 2003, pp. 69–76.

[37] A. M. Odlyzko, "The case against micropayments," in *FC 2003: 7th International Conference on Financial Cryptography*, ser. Lecture Notes in Computer Science, R. Wright, Ed., vol. 2742. Guadeloupe, French West Indies: Springer, Heidelberg, Germany, Jan. 27–30, 2003, pp. 77–83.

[38] B. Yang and H. García-Molina, "PPay: Micropayments for peer-to-peer systems," in *ACM CCS 2003: 10th Conference on Computer and Communications Security*, S. Jajodia, V. Atluri, and T. Jaeger, Eds. Washington, DC, USA: ACM Press, Oct. 27–30, 2003, pp. 300–310.

[39] S. T. Ali, D. Clarke, and P. McCorry, "The nuts and bolts of micropayments: A survey," *CoRR*,

vol. abs/1710.02964, 2017. [Online]. Available: http://arxiv.org/abs/1710.02964

[40] R. Pass and a. shelat, "Micropayments for decentralized currencies," in *ACM CCS 2015: 22nd Conference on Computer and Communications Security*, I. Ray, N. Li, and C. Kruegel, Eds. Denver, CO, USA: ACM Press, Oct. 12–16, 2015, pp. 207–218.

[41] K. Hu and Z. Zhang, "Fast lottery-based micropayments for decentralized currencies," in *ACISP 18: 23rd Australasian Conference on Information Security and Privacy*, ser. Lecture Notes in Computer Science, W. Susilo and G. Yang, Eds., vol. 10946. Wollongong, NSW, Australia: Springer, Heidelberg, Germany, Jul. 11–13, 2018, pp. 669–686.

[42] A. Chiesa, M. Green, J. Liu, P. Miao, I. Miers, and P. Mishra, "Decentralized anonymous micropayments," in *Advances in Cryptology – EUROCRYPT 2017, Part II*, ser. Lecture Notes in Computer Science, J. Coron and J. B. Nielsen, Eds., vol. 10211. Paris, France: Springer, Heidelberg, Germany, Apr. 30 – May 4, 2017, pp. 609–642.

[43] P. McCorry, C. Buckland, S. Bakshi, K. Wüst, and A. Miller, "You sank my battleship! a case study to evaluate state channels as a scaling solution for cryptocurrencies."

[44] "Ligthing network daemon," https://github.com/lightningnetwork/lnd, accessed: 2019-03-26.

[45] "A scala implementation of the lightning network," https://github.com/ACINQ/eclair, accessed: 2019-03-26.

[46] "c-lightning a lightning network implementation in c," https://github.com/ElementsProject/lightning, accessed: 2019-03-26.

[47] D. Peterson, "Sparky: A lightning network in two pages of solidity," http://wpage.unina.it/giovanni.distasi/pub/blockchain2018-main.pdf, 2018, accessed: 2019-03-30.

[48] P. Hoenisch and I. Weber, "Aodv–based routing for payment channel networks," in *Blockchain – ICBC 2018*, S. Chen, H. Wang, and L.-J. Zhang, Eds. Cham: Springer International Publishing, 2018, pp. 107–124.

[49] A. Miller, I. Bentov, R. Kumaresan, and P. McCorry, "Sprites: Payment channels that go faster than lightning," *CoRR abs/1702.05812*, 2017.

[50] R. Khalil and A. Gervais, "Revive: Rebalancing off-blockchain payment networks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 439–453.

[51] G. Malavolta, P. Moreno-Sanchez, A. Kate, and M. Maffei, "Silentwhispers: Enforcing security and privacy in credit networks." NDSS, 2017.

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

[53] V. Sivaraman, S. B. Venkatakrishnan, M. Alizadeh, G. Fanti, and P. Viswanath, "Routing cryptocurrency with the spider network," *arXiv preprint arXiv:1809.05088*, 2018.

[54] P. Prihodko, S. Zhigulin, M. Sahno, A. Ostrovskiy, and O. Osuntokun, "Flare: An approach to routing in lightning network," 2016.

[55] D. Piatkivskyi and M. Nowostawski, "Split payments in payment networks," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, 2018, pp. 67–75.

[56] O. Osuntokun, "[lightning-dev] amp: Atomic multi-path payments over lightning," https://lists.linuxfoundation.org/pipermail/lightning-dev/2018-February/000993.html, accessed: 2018-10-11.

[57] P. McCorry, S. Bakshi, I. Bentov, A. Miller, and S. Meiklejohn, "Pisa: Arbitration outsourcing for state channels," Cryptology ePrint Archive, Report 2018/582, 2018, https://eprint.iacr.org/2018/582.

[58] G. Avarikioti, F. Laufenberg, J. Sliwinski, Y. Wang, and R. Wattenhofer, "Towards secure and efficient payment channels," *arXiv preprint arXiv:1811.12740*, 2018.

[59] G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, and S. Ravi, "Concurrency and privacy with payment-channel networks," in *ACM CCS 2017: 24th Conference on Computer and Communications Security*, B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds. Dallas, TX, USA: ACM Press, Oct. 31 – Nov. 2, 2017, pp. 455–471.

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

[61] C. Buckland and P. McCorry, "Two-party state channels with assertions."

[62] S. Dziembowski, L. Eckey, S. Faust, and D. Malinowski, "PERUN: Virtual payment channels over cryptographic currencies," Cryptology ePrint Archive, Report 2017/635, 2017, http://eprint.iacr.org/2017/635.

[63] J. Coleman, L. Horne, and L. Xuanji, "Counterfactual: Generalized state channels," 2018.

[64] Y. Zhang, Y. Long, Z. Liu, Z. Liu, and D. Gu, "Z-channel: Scalable and efficient scheme in zerocash," in *ACISP 18: 23rd Australasian Conference on Information Security and Privacy*, ser. Lecture Notes in Computer Science, W. Susilo and G. Yang, Eds., vol. 10946. Wollongong, NSW, Australia: Springer, Heidelberg, Germany, Jul. 11–13, 2018, pp. 687–705.

[65] R. Khalil and A. Gervais, "NOCUST - A non-custodial 2nd-layer financial intermediary," Cryptology ePrint Archive, Report 2018/642, 2018, https://eprint.iacr.org/2018/642.

[66] "Raiden network," raiden.network, accessed: 2018-09-03.

[67] M. Green and I. Miers, "Bolt: Anonymous payment channels for decentralized currencies," in *ACM CCS 2017: 24th Conference on Computer and Communications Security*, B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds. Dallas, TX, USA: ACM Press, Oct. 31 – Nov. 2, 2017, pp. 473–489.

[68] J. Lind, I. Eyal, P. Pietzuch, and E. G. Sirer, "Teechan: Payment channels using trusted execution environments," *arXiv preprint arXiv:1612.07766*, 2016.

[69] C. Burchert, C. Decker, and R. Wattenhofer, "Scalable funding of bitcoin micropayment channel networks-regular submission." in *SSS*, 2017, pp. 361–377.

[70] "Rapidly-adjusted (micro)payments to a pre-determined party," https://en.bitcoin.it/wiki/Contract, 2011, accessed: 2018-09-03.

[71] A. Dmitrienko, D. Noack, and M. Yung, "Secure wallet-assisted offline bitcoin payments with double-spender revocation," in *ASIACCS 17: 12th ACM Symposium on Information, Computer and Communications Security*, R. Karri, O. Sinanoglu, A.-R. Sadeghi, and X. Yi, Eds. Abu Dhabi, United Arab Emirates: ACM Press, Apr. 2–6, 2017, pp. 520–531.

[72] T. Takahashi and A. Otsuka, "Short paper: Secure offline payments in bitcoin."

[73] I. Bentov, R. Kumaresan, and A. Miller, "Instantaneous decentralized poker," in *Advances in Cryptology – ASIACRYPT 2017, Part II*, ser. Lecture Notes in Computer Science, T. Takagi and T. Peyrin, Eds., vol. 10625. Hong Kong, China: Springer, Heidelberg, Germany, Dec. 3–7, 2017, pp. 410–440.

[74] B. David, R. Dowsley, and M. Larangeira, "Kaleidoscope: An efficient poker protocol with payment distribution and penalty enforcement," Cryptology ePrint Archive, Report 2017/899, 2017, http://eprint.iacr.org/2017/899.

[75] ——, "ROYALE: A framework for universally composable card games with financial rewards and penalties enforcement," Cryptology ePrint Archive, Report 2018/157, 2018, https://eprint.iacr.org/2018/157.

[76] ——, "21 - bringing down the complexity: Fast composable protocols for card games without secret state," in *ACISP 18: 23rd Australasian Conference on Information Security and Privacy*, ser. Lecture Notes in Computer Science, W. Susilo and G. Yang, Eds., vol. 10946. Wollongong, NSW, Australia: Springer, Heidelberg, Germany, Jul. 11–13, 2018, pp. 45–63.

[77] P. McCorry, M. Möser, S. F. Shahandashti, and F. Hao, "Towards bitcoin payment networks," in *ACISP 16: 21st Australasian Conference on Information Security and Privacy, Part I*, ser. Lecture Notes in Computer Science, J. K. Liu and R. Steinfeld, Eds., vol. 9722. Melbourne, VIC, Australia: Springer, Heidelberg, Germany, Jul. 4–6, 2016, pp. 57–76.

[78] "Bip 68," https://github.com/bitcoin/bips/blob/master/bip-0068.mediawiki, accessed: 2018-11-28.

[79] C. Decker, R. Russel, and O. Osuntokun, "eltoo: A simple layer2 protocol for bitcoin," Whitepaper, https://blockstream.com/eltoo.pdf, accessed: 2019-03-29.

[80] B. Bip, "Bip-0065," https://github.com/bitcoin/bips/blob/master/bip-0065.mediawiki, 2018, [Online; accessed 30-October-2018].

[81] R. Kumaresan and I. Bentov, "Amortizing secure computation with penalties," in *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds. Vienna, Austria: ACM Press, Oct. 24–28, 2016, pp. 418–429.

[82] R. Kumaresan, V. Vaikuntanathan, and P. N. Vasudevan, "Improvements to secure computation with penalties," in *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds. Vienna, Austria: ACM Press, Oct. 24–28, 2016, pp. 406–417.

[83] J. Spilman, "[bitcoin-development] anti dos for tx replacement," https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2013-April/002433.html, 2013, accessed: 2018-09-03.

[84] D. Wheeler, "Transactions using bets," in *International Workshop on Security Protocols*. Springer, 1996, pp. 89–92.

[85] R. L. Rivest, "Electronic lottery tickets as micropayments," in *International Conference on Financial Cryptography*. Springer, 1997, pp. 307–314.

[86] R. J. Lipton and R. Ostrovsky, "Micro-payments via efficient coin-flipping," in *International Conference on Financial Cryptography*. Springer, 1998, pp. 1–15.

[87] S. Micali and R. L. Rivest, "Micropayments revisited," in *Cryptographers Track at the RSA Conference*. Springer, 2002, pp. 149–163.

[88] P. McCorry, M. Möser, S. F. Shahandasti, and F. Hao, "Towards bitcoin payment networks," in *Australasian Conference on Information Security and Privacy*. Springer, 2016, pp. 57–76.

[89] "Lightning in scriptless script," Mailing list port: https://github.com/lightningnetwork/lnd, accessed: 2019-03-26.

[90] "Scriptless script," Presentation slides: https://download.wpsoftware.net/bitcoin/wizardry/mw-slides/2017-05-milan-meetup/slides.pdf, accessed: 2019-03-26.

[91] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei, "Multi-hop locks for secure, privacy-preserving and interoperable payment-channel networks," Cryptology ePrint Archive, Report 2018/472, 2018, https://eprint.iacr.org/2018/472.

[92] "Zcash," https://z.cash, 2016, accessed: 2018-09-26.

[93] J. Camenisch, S. Hohenberger, and A. Lysyanskaya, "Compact e-cash," in *Advances in Cryptology – EUROCRYPT 2005*, ser. Lecture Notes in Computer Science, R. Cramer, Ed., vol. 3494. Aarhus, Denmark: Springer, Heidelberg, Germany, May 22–26, 2005, pp. 302–321.

[94] D. Chaum, "Blind signatures for untraceable payments," in *Advances in Cryptology – CRYPTO'82*, D. Chaum, R. L. Rivest, and A. T. Sherman, Eds. Santa Barbara, CA, USA: Plenum Press, New York, USA, 1982, pp. 199–203.

[95] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE Symposium on Security and Privacy*. Berkeley, CA, USA: IEEE Computer Society Press, May 18–21, 2014, pp. 459–474.

[96] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek, "Secure multiparty computations on bitcoin," in *2014 IEEE Symposium on Security and Privacy*. Berkeley, CA, USA: IEEE Computer Society Press, May 18–21, 2014, pp. 443–458.

[97] ——, "Fair two-party computations via bitcoin deposits," in *FC 2014 Workshops*, ser. Lecture Notes in Computer Science, R. Böhme, M. Brenner, T. Moore, and M. Smith, Eds., vol. 8438. Christ Church, Barbados: Springer, Heidelberg, Germany, Mar. 7, 2014, pp. 105–121.

[98] I. Bentov and R. Kumaresan, "How to use bitcoin to design fair protocols," in *Advances in Cryptology – CRYPTO 2014, Part II*, ser. Lecture Notes in Computer Science, J. A. Garay and R. Gennaro, Eds., vol. 8617. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 17–21, 2014, pp. 421–439.

[99] S. Dziembowski, S. Faust, and K. Hostakova, "Foundations of state channel networks," Cryptology ePrint Archive, Report 2018/320, 2018, https://eprint.iacr.org/2018/320.

[100] A. Ghosh, M. Mahdian, D. M. Reeves, D. M. Pennock, and R. Fugger, "Mechanism design on trust networks," in *Internet and Network Economics*, X. Deng and F. C. Graham, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 257–268.

[101] D. Karlan, M. Mobius, T. Rosenblat, and A. Szeidl, "Trust and social collateral," *The Quarterly Journal of Economics*, vol. 124, no. 3, pp. 1307–1361, 2009.

[102] P. Resnick and R. Sami, "Sybilproof transitive trust protocols," in *Proceedings of the 10th ACM Conference on Electronic Commerce*, ser. EC '09. New York, NY, USA: ACM, 2009, pp. 345–354. [Online]. Available: http://doi.acm.org/10.1145/1566374.1566423

[103] P. Dandekar, A. Goel, R. Govindan, and I. Post, "Liquidity in credit networks: A little trust goes a long way," in *Proceedings of the 12th ACM Conference on Electronic Commerce*, ser. EC '11. New York, NY, USA: ACM, 2011, pp. 147–156. [Online]. Available: http://doi.acm.org/10.1145/1993574.1993597

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

[105] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *42nd Annual Symposium on Foundations of Computer Science*. Las Vegas, NV, USA: IEEE Computer Society Press, Oct. 14–17, 2001, pp. 136–145.

[106] "tpec: 2p-ecdsa signatures," Github repository, https://github.com/cfromknecht/tpec, accessed: 2019-03-27.

[107] R. C. Giovanni Di Stasi, Stefano Avallone and G. Ventre, "Routing payments on the lightning network," http://wpage.unina.it/giovanni.distasi/pub/blockchain2018-main.pdf, 2018, accessed: 2019-03-28.

[108] P. F. Tsuchiya, "The landmark hierarchy: A new hierarchy for routing in very large networks," *SIGCOMM Comput. Commun. Rev.*, vol. 18, no. 4, pp. 35–42, Aug. 1988. [Online]. Available: http://doi.acm.org/10.1145/52325.52329

[109] S. Roos, M. Beck, and T. Strufe, "Anonymous addresses for efficient and resilient routing in f2f overlays," in *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*. IEEE, 2016, pp. 1–9.

[110] "Ripple," ripple.com, accessed: 2018-10-17.

[111] "Speedymurmurs," https://crysp.uwaterloo.ca/software/speedymurmurs, accessed: 2018-10-17.

[112] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, p. 399404, 1956.

[113] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *Selected Areas in Communications, IEEE Journal on*, vol. 29, no. 9, pp. 1765 –1775, October 2011.

[114] G. Avarikioti, F. Laufenberg, J. Sliwinski, Y. Wang, and R. Wattenhofer, "Incentivizing payment channel watchtowers," in *Scaling Bitcoin*, 2018.

[115] S. Brânzei, E. Segal-Halevi, and A. Zohar, "How to charge lightning," *arXiv preprint arXiv:1712.10222*, 2017.

[116] "Cosmo," cosmos.network, accessed: 2019-03-27.

[117] A. Zamyatin, D. Harz, J. Lind, P. Panayiotou, A. Gervais, and W. J. Knottenbelt, "XCLAIM: Interoperability with cryptocurrency-backed tokens," Cryptology ePrint Archive, Report 2018/643, 2018, https://eprint.iacr.org/2018/643.

[118] "Comit whitepaper," https://www.comit.network/doc/COMIT/white/paper/v1.0.2.pdf, note = Accessed: 2019-03-27.