A Survey of Layer-Two Blockchain Protocols

Ankit Gangwal, Haripriya Ravali Gangavalli*, and Apoorva Thirupathi*

***Abstract*—After the success of the Bitcoin blockchain, came several cryptocurrencies and blockchain solutions in the last decade. Nonetheless, Blockchain-based systems still suffer from low transaction rates and high transaction processing latencies, which hinder blockchains' scalability. An entire class of solutions, called *Layer-1* scalability solutions, have attempted to incrementally improve such limitations by adding/modifying fundamental blockchain attributes. Recently, a completely different class of works, called *Layer-2* protocols, have emerged to tackle the blockchain scalability issues using unconventional approaches. *Layer-2* protocols improve transaction processing rates, periods, and fees by minimizing the use of underlying slow and costly blockchains. In fact, the main chain acts just as an instrument for trust establishment and dispute resolution among *Layer-2* participants, where only a few transactions are dispatched to the main chain. Thus, *Layer-2* blockchain protocols have the potential to transform the domain. However, rapid and discrete developments have resulted in diverse branches of *Layer-2* protocols. In this work, we systematically create a broad taxonomy of such protocols and implementations. We discuss each *Layer-2* protocol class in detail and also elucidate their respective approaches, salient features, requirements, etc. Moreover, we outline the issues related to these protocols along with a comparative discussion. Our thorough study will help further systematize the knowledge dispersed in the domain and help the readers to better understand the field of *Layer-2* protocols.**

***Index Terms*—Blockchain, *Layer-2*, Off-chain, Scalability.**

I. Introduction

Blockchain is a digital ledger of assets (e.g., financial transactions) that is typically managed by a network of peer-to-peer nodes. It provide a transparent and decentralized approach for publicly-verifiable and tamper-evident record keeping. Its unique properties help in eliminating the control of a centralized authority, provide ubiquity, and facilitate fairness via its underlying consensus protocol. Blockchain is essentially the fundamental building block of Bitcoin [1], which is a decentralized cryptocurrency - or, simply a digital cash system - for which the researchers have been working towards over multiple decades [2, 3]. Blockchain helps in establishing an agreement between mutually distrusting entities even in the absence of a trusted third party. After the success of Bitcoin, a multitude of financial and non-financial fields have been dramatically transformed by the idea of utilizing a blockchain-based distributed public ledger.

According to a widely accepted belief, called blockchain trilemma [4], blockchains can prioritize only two features among decentralization, security, and scalability. Decentralization reflects the fundamental nature of a blockchain while security is an absolute requirement. Therefore, achieving scalability has always remained a challenge for the blockchain researchers and developers. Even after a decade of its birth, Bitcoin still suffers from high transaction latency and fails to handle transaction load when compared to conventional payment systems. One of the key factors behind limited scalability of blockchain is directly related to its core working principle, i.e., their underlying consensus protocol. As a representative example, a block in the Bitcoin blockchain can fit only a limited number of transactions while the Bitcoin network adapts itself to generate only one block every ten minutes on average. Such calibrations have severely restricted its transaction throughput to roughly ten Transactions Per Second (TPS) [5] while regular payment systems, such as VISA and PayPal, handle thousands of TPS.

To tackle the issue of scalability, researchers from both academia and industry have proposed different solutions for scaling blockchains. The primary class of such solutions, commonly known as *Layer-1* solutions, mainly targets and improves the working principles of blockchains by (i) modifying block data [6–8]; (ii) proposing alternative consensus mechanisms [9–15];(iii) sharding the network [16–20]; or (iv) using solutions based on Directed Acyclic Graphs (DAG) [21–23] (cf. Fig. 1). Since *Layer-1* solutions involve changing the core design elements of blockchains, these solutions typically lack backward compatibility. As a representative example, modifying the consensus mechanism of a blockchain that is already in-use leads to blockchain forking. Similarly, sharding protocols make significant changes to the overall network layout. Thus, *Layer-1* solutions come with critical issues that hinder their implementation in practice [24].

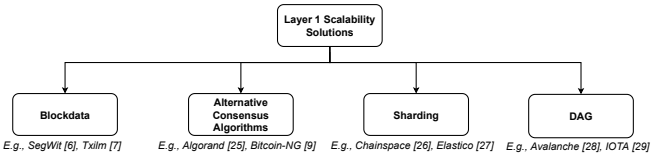


Fig. 1. *Layer-1* scalability solutions.

The limitations of *Layer-1* solutions induced an orthogonal research direction for blockchain scalability, generally called *Layer-2* blockchain protocols. These protocols aim to scale blockchains without altering the underlying consensus mechanism of the concerned blockchain or modifying the *Layer-1* trust assumptions. These solutions are called *Layer-2* as they are primarily built over the stack of blockchain layers (cf. Fig. 2), where the lowest level (i.e., *layer -1*) represents the hardware, *layer 0* comprises the network of nodes used for information exchange, the blockchain executes in the *layer 1* of the stack, and *Layer-2* scalability solutions sit

* Both authors contributed equally.
All authors are with International Institute of Information Technology Hyderabad, India. E-mail: gangwal@iiit.ac.in, haripriya.ravali@students.iiit.ac.in, apoorva.thirupathi@research.iiit.ac.in.

in *layer 2*. *Layer-2* protocols do not broadcast every transaction on the underlying main chain, they instead enable participants to execute off-the-chain transactions over an authenticated communication medium. As a result, transaction load on the main chain is immensely reduced without compromising on backward compatibility. The transactions in *Layer-2* protocols are secured with collateral (e.g., in payment channels [30–33]) or with delayed finality (e.g., in commit chains [34]).

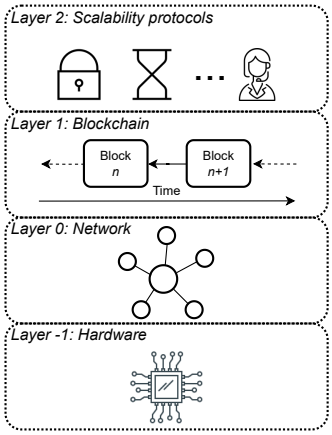


Fig. 2. Blockchain layered stack, common in blockchain community.

Motivation: The unique and promising features of *Layer-2* protocols have attracted the attention of researchers in the community. As a result, the research efforts on *Layer-2* blockchain scalability protocols are continuing to expand pervasively. Different *Layer-2* solutions come with their own set of goals, assumptions, requirements, advantages, etc. There is little clarity - especially for new entrants to the community - to map, understand, and evaluate different *Layer-2* solutions. Despite the existence of a rich body of literature on various *Layer-2* solutions, a comprehensive study to cover the state of the art detailing their characteristics, limitations, issues, etc. is still missing. Thus, navigating through this research space is not straight forward; especially when the field is growing at a fast pace in different directions. We aim to fill such gap in the literature by consolidating and systematizing the information about the state of the art of *Layer-2* blockchain protocols.

Contribution: In this paper, we survey various *Layer-2* blockchain scalability protocols proposed over the years since the birth of Bitcoin in 2009. Our aim is to present a holistic view of this research field. To the best of our knowledge, our work is the first such study. We create a broad taxonomy of *Layer-2* protocols and implementations and discuss each protocol class in detail. In particular, we explain their respective approaches, key characteristics, advantages, limitations, etc. We also discuss the key networking aspects, security concerns, and privacy issues present in the literature. Finally, we present a comparative discussion to help readers assess the feasibility of different *Layer-2* solutions.

Organization: The remainder of this paper is organized as follows. Section II covers the key fundamental concepts and related works. We discuss in detail different *Layer-2* protocols in Section III. Section IV describes networking aspects while Section V focuses on security and privacy issues. We present a comparative discussion on different *Layer-2* scalability solutions in Section VI. Finally, Section VII concludes the paper.

II. Background

We introduce the key building blocks of *Layer-2* protocols in Section II-A and the related works in Section II-B.

A. Blockchain and HTLC

Blockchain is an immutable, linked-list style, append-only chain of blocks, where each block stores transactions sent among network entities. Typically, each transaction reflects an exchange of digital assets between network peers. Participants in the network execute a consensus algorithm to achieve a common agreement about the state of the blockchain, which also helps in maintaining its integrity. Today, there are a number of consensus algorithms available. Different consensus algorithms follow fundamentally different approach to achieve distinct goals. Another key aspect is whether the access to blockchain is open or restricted. In the former case, the blockchain is permissionless while the latter represents a permissioned blockchain. Finally, the scripting language supported by the blockchain defines its expressiveness. As a representative example, Bitcoin blockchain uses a simple and Turing-incomplete script [1] while Ethereum uses a Turing-complete language to support more powerful smart contract [35]. While *Layer-2* protocols can be built upon both permissioned and permissionless blockchains, the expressiveness of underlying blockchain plays an important role in designing *Layer-2* protocols built upon it. Importantly, *Layer-2* protocols assume that the underlying blockchain will only include valid transaction to the ledger.

The key to a successful P2P transaction system without a central entity relies on a simple and efficient trust-based mechanism. Hash Time Locked Contract (HTLC) [32, 36, 37] provides such a solution and acts as the fundamental construction for several *Layer-2* protocols. As a representative example, HTLC implementation can be observed in Lightning network. The core idea of HTLC includes a hash verification and time expiration. Fig. 3 shows an example of using HTLC in payment channels.

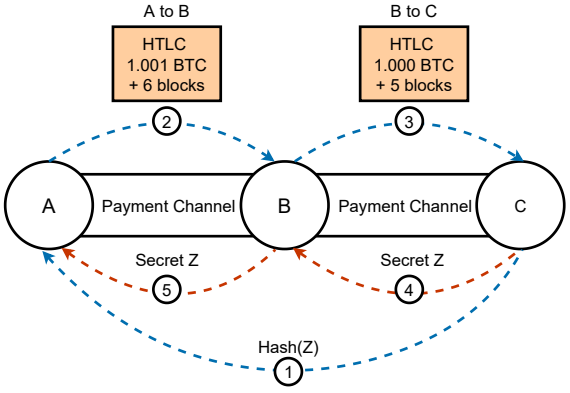


Fig. 3. A representative example of using HTLC

Consider a payment channel network from A to C through B, where A is the sender and C is the receiver of funds. C

generates a secret Z and computes its hash Hash(Z). Then, C communicates Hash(Z) to A. A locks the required funds (including a fee for transfer to intermediaries) in his channel to B and shares Hash(Z) with B in a locking script. B can retrieve the funds only if it can produce Z corresponding to Hash(Z). B further locks funds in his channel with C and passes Hash(Z) to C; B reduces the time to reveal Z and makes some margin on the fee. C can claim these funds only it can produce Z. Since C knows Z, it can redeem funds in the channel between B and C by revealing Z to B, which can further collect the funds from channel between B and A. Thus, intermediaries receive a fee for relaying the transaction. It is worth mentioning that each party can safely participate without worry about losing their funds as funds frozen in the channel are returned to the original sender when the secret is not produced. Once the secret is revealed each involved intermediary would work to redeem its payment from the previous channel before the time expiration.

B. Related works

Several efforts from both academia and industry have been made to address the scalability issues in blockchain. These efforts have targeted different of aspects of blockchains starting from finding alternative consensus algorithms, modifying block size, sharding, DAG, *Layer-2* protocols, etc. Continuous and rapid development in this research area have led to many parallel as well as distinct branches of works. Many works have attempted to systematize the knowledge in the rich body of the literature.

Several works, such as [38, 39], survey scaling solutions that directly engage with the fundamental building blocks of the blockchains. *Layer-2* scaling solutions remain out of the scope of such surveys. Only a brief literature on the survey of *Layer-2* solutions exists; many of which focus on creating a taxonomy. Authors in [40] classify *Layer-2* solutions along few dimensions. However, their classification omits major protocols such as bisection protocols, commit chains, and TEE (Trusted Execution Environment)-based solutions. Similarly the works [41, 42] focus on only popular *Layer-2* protocols. Authors in [43] discuss various categories of *Layer-2* protocols while the work [44] focuses primarily on the network and routing aspects of *Layer-2* solutions.

Our paper aims at furnishing a comprehensive guide for *Layer-2* protocols, starting with a much broader taxonomy, detailed explanation of each protocol, their salient features, and their key concerns. To the best of our knowledge, our work covers all the solutions present as of December 2021.

III. Layer-two blockchain protocols

In this section, we elucidate different *Layer-2* protocol along with their requirements, working procedures, salient features, etc. In Fig. 4, we depict the taxonomy of different blockchain scalability solutions at *Layer-1* as well as *Layer-2* Here, a box represents a class/subclass while implementations in a class/subclass are mentioned below respective boxes. These protocols can be broadly categorized into four classes, i.e., channels, side/child chains, cross chains, and hybrid solutions. We now describe each of the categories.

A. Channels

One of the key *Layer-2* protocols to achieve scalability along with privacy is channels. Channels enable any pair of users to create private mediums for their transactions. The main idea is to enable transactions to happen off the main blockchain and yet maintain the same level of security as an on-chain transaction. For transactions' security a set of rules are predefined and agreed between the participants. Channels can be mainly classified into two main categories, i.e., state channels and payment channels. State channels (discussed in Section III-A1) are generalized version while payment channels (discussed in Section III-A2) are specific to payment-oriented applications. For this reason, we show payment channels in a branch nested from the state channels in Fig. 4. Payment channels have been further improvised to form a network and a hub.

1) State channels: A state channel [45] is a channel that allows exchange/transfer of states between two or more participants. These states can represent any arbitrary application (e.g., voting, auctions). Typically, a channel can be built upon threshold signatures - often referred as *multisig* - and instructions for timelocks [130], where the participants sign a multisig contract and lock in funds to participate in such a transfer. In practice, state channels are established using smart contract as shown in Fig. 5. On these channels, the states are exchanged among all the participants who enter the branched out channel of states. Once all the transactions complete, the participants commit the final state of the channel to the main chain via the contract.

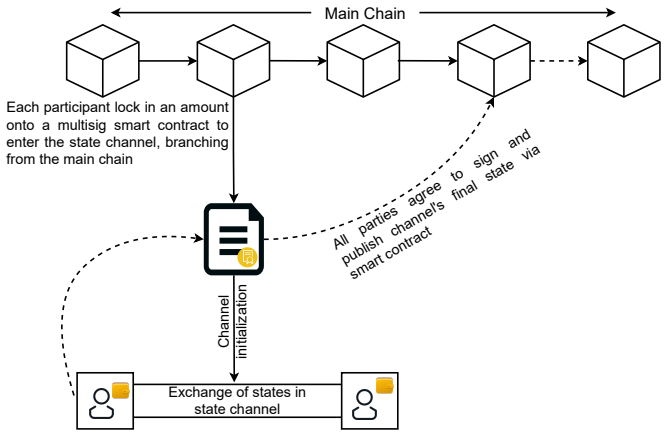


Fig. 5. Typical lifecycle of a state channel.

Such a channel is especially useful when states are exchanged between the participants frequently because off-chain state exchange is far more faster than on-chain exchanges. Thus, state channels greatly improve the slow transaction rate of parent blockchain.

Lifecycle: The typical lifecycle of a state channel consists of establishment, execution, and termination phases. To establish the channel, the participants first lock some funds (or assets) using a smart contract on the main chain. The sum of the initially locked funds is the capacity of the established channel. Before proceeding with execution the participants wait for the

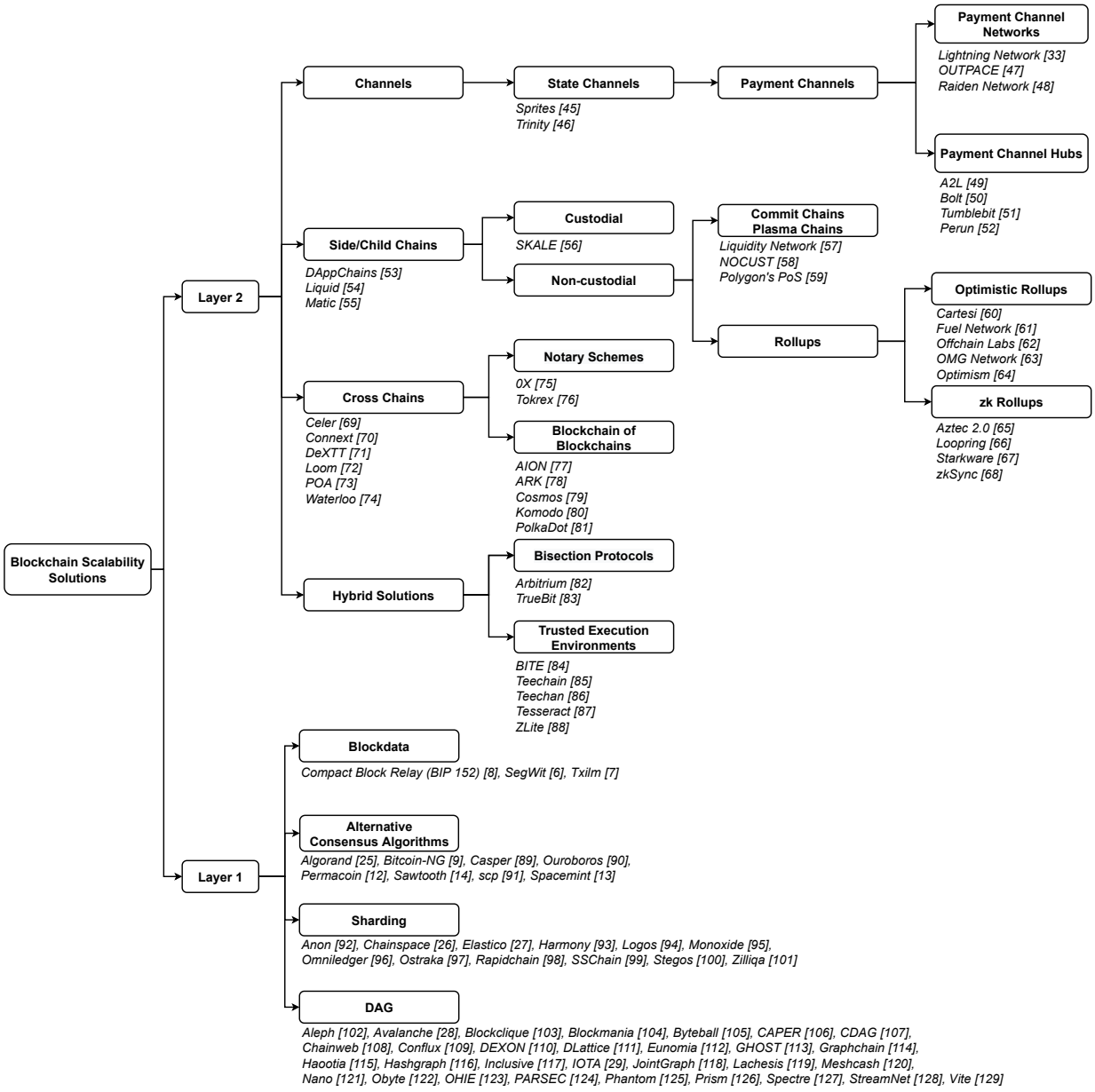


Fig. 4. Taxonomy of blockchain scalability solutions.

confirmation of fund locking on the main chain. Moreover, the locked funds can not be used outside the channel. In the execution phase, a transaction happens by an exchange of states between participants. A transaction redistributes the funds from the last agreed states. After exchanging states, the involved participants sign and authorize the new states as valid and true. This new state is then shared with other participants and the order of states is logged. Next, a participant publishes the final state of the channel to the smart contract, which verifies the signatures. Now for the duration of a challenge period, participants other than the publisher are allowed to check if the state is correct. In case of dispute, one can publish the appropriate final state disapproving the previously published state. All the other participants are informed about it, and the challenge period restarts. Typically, the state with highest version number is considered as the latest state. The latest state is executed after the challenge period ends to reflect each participants' state.

State replacement: A transaction in state channels is essentially equivalent to replacing the old state with the new state. State replacement should ideally happen once for transaction finality. But to accommodate disagreements among participants about the new proposed state, some state replacement techniques offer a dispute mechanism. Overall, there are four main state replacement techniques:

- *Replace-by-Incentive (RbI):* The sender of a transaction signs and announces a new state. The receiver needs to countersign it to accept the announced state. The motivation of the receiver to accept the state is an incentive; a higher incentive converts to higher chances of state acceptance [30, 48, 131].
- *Replace-by-Timelock (RbT):* A state has an associated

timelock in terms of either absolute or relative blockchain block height. As the blockchain's block height increases with time, the remaining timelock on a state decreases. Before the expiry of timelock, a state can be replaced with a newer state. Intuitively, a state with the lowest timelock gets included in the blockchain before older states. Post timelock expiry, the transactions represented in a state are written to the blockchain and can not be replaced [32].

- *Replace-by-Revocation (RbR):* There might be a situation where participants want to revoke a state submitted to the blockchain. To do so, all the participants must together propose a new state within a time window defined by the parent blockchain [33].
- *Replace-by-Version (RbV):* Here, the version of a state is represented by an incrementing counter. A higher version number means a newer state. So, a state with a higher number can replace the older state [45, 52, 132–134].

RbI and RbT allow the latest state to be inserted into the blockchain only once. The participants in RbR and RbV can invalidate the submitted state via a dispute process of presenting counter-evidence. The dispute process leads to either a closure dispute or a command dispute. A closure dispute proceeds towards closing the channel and resolving the dispute exclusively on underlying blockchain. After the dispute is raised, the relevant parties provide evidences within a fix time duration. At the end of the evidence submission step, any one of the participants processes the evidences to resolve the dispute [52, 133]. Instead of closing the channel, command disputes execute a set of commands on the parent-chain to resolve disputes. After command execution, the channel resumes its operation off-chain. The blockchain provides a fixed time duration to collect commands from the participants and executes all the collected commands to find a resolution [45]. Some solutions [134, 135] extend dispute process expiry time to support execution of a large number of commands. However, both the dispute resolution mechanisms assume the relevant participants to be always online. Watching services [132, 136–138] help participants subvert the requirement of staying online by taking the responsibility of observing disagreements.

Advantages: The main advantage of using state channels is that all the exchanges happen inside the channel. Unlike main chain transactions where each transaction is broadcast, state channel only publish the final states onto the parent-chain offering more privacy. Instant transaction finality is another advantage, i.e., as soon as all the participants authorize a state update, transaction in that state can be safely considered final. Furthermore, state channels are very economical; especially when state updates are expected to happen frequently between participants. It is so as the cost of updating the states inside the channel is cheaper compared to main chain transaction fees.

Limitations: State channels are not suggested for scenarios where the participants are not fixed, i.e., the participants come and leave or whose addresses are not known. Essentially, all the participants must open dedicated channels and be present for state exchange to occur. Furthermore, the dispute process induces an always online assumption. Watching services help here, but such services increase the cost for the participants.

2) Payment channels: Enabling blockchains to support (micro-) payments with near-instant confirmation, fewer on-chain transactions, and reduced fees has been one of the major scalability goals [139–142]. Payment channels tailor state channels for payment-specific applications. Initially designed to support one-way payments [30], payment channels evolved into bi-directional channels [32, 33] to empower each participant to send and receive payments.

Lifecycle: Similar to state channels, the lifecycle of a payment channel comprises of establishment, execution, and termination/dispute of the payment channel. The payer creates a channel by setting an expiration time, a settlement delay, and a public key to verify claims against the channel. The payee checks if the parameters of the payment channel are suitable for its specific requirements, e.g., destination, settlement delay, channel ID, etc. Importantly, there can be multiple channels between the same pair of participants. Thus, it is important to check the attributes of a channel before initiating a payment. The payer creates a signed claim for the required amount of payment in the channel, which it sends to the payee as the payment for goods or services. It is worth noting that this communication happens "off-ledger" over a communication medium suitable for the payer and payee. The payee verifies the claim to ensure that the claim amount is greater than or equal to the total value of the services provided. At this point, the payee can release the goods to the payer because the payment has been assured. The payee is now free to redeem a claim for the authorized amount at any point of time. As the claim values are cumulative, redeeming the largest, i.e., the most recent, claim is sufficient for the payee to get the full amount. A channel closure request can lead to two scenarios depending upon whether some funds are still remaining the channel. If the channel has no fund remaining in it, then the channel can close immediately. Otherwise, the request to close the channel serves as an intimation to the payee to redeem any outstanding claims by the end of settlement delay. The channel expires after the settlement delay has elapsed or the planned expiration time for the channel has arrived. Further transactions can only close the channel, returning any unclaimed funds to the payer. However, an expired channel can last indefinitely on the ledger in its expired state because the ledger can not close it with a closure transaction.

Channel extensions: Payment channels help channel participants to avoid publishing every transaction on the main chain and wait for subsequent confirmations. Thus, the payments are processed faster and finalized instantly. Fig. 6 shows a typical bi-directional channel, where two participants transact with each other in either direction after locking funds during channel setup. Closing the channel reflects their respective final state on the main chain.

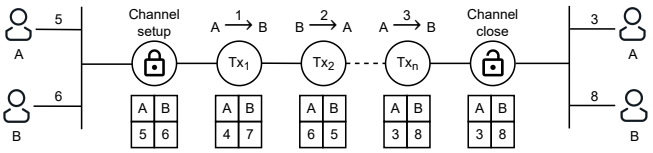


Fig. 6. A simplified lifecycle of a typical payment channel.

Such an approach is especially useful when the participants frequently transact. However, there are some limitations associated with the payment channels starting with the creation of the channel. Setting up a channel requires locking funds exclusively to the channel. The initial fund locking is also not instant and requires confirmation from the main chain. Moreover, there must be a dedicated channel between the participants. Therefore, such constraints limit the usage of payment channels for micropayments. Nevertheless, several solutions have been proposed to improvise on payment channels, including channel factories, Payment Channel Network (PCN), payment channel hubs, and virtual channels.

- In *channel factories* [143, 144], many participants jointly fund a factory. In particular, n participants jointly lock funds in an n-party deposit, which is then used to create payment channels for each pair of depositors. Whenever two participants want to establish a direct channel between them, all depositors update the n-party deposit to re-allocate funds for the new channel. The advantage here is that there is no need to fund and set up separate payment channels for each pair of participants. Nonetheless, opening a factory via n-party fund locking still requires confirmation from the parent chain.
- Payment channels in their original form require participants to have a direct link or channel between them. Such requirement limits the potential, and to some extent scalability, of the payment channels. The reason is the practicality (i.e., channel setup delay, locked funds required, etc.) of having a direct channel with many, if not all, participants. *PCNs* [33] help with the requirement of having a direct channel by creating a network of channels. PCNs have attracted a lot of attention from both academia and industry. The idea behind PCNs is that if A has a channel with B, who has a channel with C. Then, PCN enables A to transact with C via B, i.e., A→B→C. B gets an incentive in terms of a small fee for participating in such a transaction. PCNs primarily utilize conditional payment constructions, such as HTLC (cf. Section II-A) [32, 33]. The payer conditionally locks the fund of a transaction such that the payee can redeem the funds only if the locking condition is met. Another parameter in this conditional lock is an expiry time, which stimulates faster resolution of the lock by the payee as well as intermediaries. Such conditional transactions must be atomic in nature, meaning that either the transaction should execute completely from the payer to payee or not execute at all. This property helps in providing security for the funds locked by the participating intermediaries [49, 145, 146]. In HTLC-based solutions, the overall amount of funds locked as collateral along the payment path increases with the length of the payment path. Furthermore, increasing payment length also increases the time for which the funds are reserved. Authors in [45] use a global PreimageManager smart contract to convert a local channel dispute to a global problem, which helps in reducing collateral locking time. Alternatively, authors in [147] introduce a novel cryptographic primitive for channel synchronization that is independent of the parent blockchain's scripting language, and therefore it removes bottlenecks induced by scripts.
- *Payment channel hubs* [50–52] aim to further optimize PCNs by introducing a special node called a hub. A hub acts as the center of a star topology and relays payment to connected nodes. The core idea here is to reduce routing overheads and funds locked by individual nodes in PCNs [148]. Multiple inter-connected hubs in a network can lead to reduced routing length, and consequently, reduced routing cost and collateral cost at each channel. However, the total funds required by a hub to lock can grow significantly with the increasing number of channels and transaction volume. The situation can worsen when the transactions flow majorly in one direction, requiring expensive and slow rebalancing operations [149].
- Channel extensions with intermediaries require intermediaries to actively participate in related transactions. Two-party [150] and multi-parti [151] *virtual channels* relax such requirements. Virtual channels give an illusion to the payer and payee of having a direct channel between them. Virtual channels are established when all intermediaries between the payer and payee lock funds for a fixed time duration. Setting up a virtual channel between a pair of participants comes at the cost of installing a new virtual channel for each intermediary, where each intermediary must oversee its channels' closure. The main advantage of virtual channels is that channels can be created and closed without blockchain interaction [151].

B. Side/Child chains

A side chain [152, 153] is an independent distributed ledger running in parallel to the main chain. Its primary goals include reducing load on the main chain by transferring computationally heavy work off the chain. It also allow assets to be transferred across different blockchains. A side chain generally utilize its own consensus mechanism (e.g., proof-of-authority and proof-of-stake) to process transactions. Side chains use a two-way bridge, called a two-way peg, to communicate and exchange funds with the main chain (cf. Fig. 7). The usability of any side chain depends on its ability to swiftly exchange information with the main chain and quickly process the transactions. Typically, side chains use custom block parameters to process transaction efficiently. In what follows, we explain the pegging mechanism used by side chains.

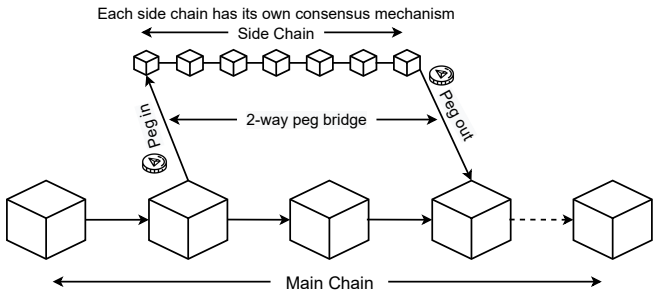


Fig. 7. A generalized view of side chains.

Lifecycle: The two-way peg mechanism allows funds to be transferred between the main chain and side chain at a deterministic exchange rate. Simplified Payment Verification (SPV) peg is used as the two-way peg. It starts with transferring required funds in the main chain to a special output. Such an output is unlocked by an SPV proof-of-possession inside the side chain. The SPV proof contains a list of block headers showcasing the proof-of-work and a cryptographic proof as evidence (i.e., proof-of-inclusion) that the output was indeed created in one of those blocks. SPV-based pegs enable verifiers to confirm the existence of the special output without downloading the entire main chain. Synchronizing the two chains involves a confirmation period and a contest period. The confirmation period corresponds to the time required for the finality of transaction on the main chain that binds funds to an special output (as mentioned above). After a finalized SPF proof is created in the main chain, the funds reflect on the side chain in a frozen state. The duration of such freezing is referred to as contest period, during which a new proofs can be published to contest the validity of the locked special output. Contest period helps in preserving integrity of fund conversion between the main chain and a given side chain.

The funds inside a side chain can move within it without any interaction with the main chain. However, funds remain bonded to the parent chain and can not be transferred further to other chains. Redeeming the funds from a side to the main chain follows the same procedure, i.e., the funds from side chain are locked to a special output, which is then spent using the corresponding SPV proof on the main chain.

Advantages: Side chains act as secondary blockchains that provide diverse features and flexibility to their main chains. A side chain has its own independent consensus protocol, and it can control the block parameters. Hence, the transactions on side chains are typically executed faster as compared to main chains. Such processing capabilities also help in reducing the load on the main chain via transaction offloading. Side chains are permanent that can keep running. A new participant can join the same side chain. In contrast, adding participants to the state channel network requires creating a new state channel for each participant. Finally, any compromise or damage remains confined to the side chain only, leaving the main chain unaffected. Such an attribute can be utilized for testing applications before their deployment to the main chain.

Limitations: A two-way pegged side chain is slower in execution as a participant needs to wait for a confirmation period as well as a contest period to access the funds on either chain. Another concern is the centralization of mining power on side chains, especially on newer ones. The initial investment required to stabilize the mining process of a side chain and its interoperability with different blockchains constitute the bottleneck of its success. Moreover, the security of funds in a side chain is handled by the side chain. Thus, disputes in side chains are local that can not be resolved in the main chain.

Side chains can be classified into two categories, namely, custodial and non-custodial. Custodial side chains move assets in a chain parallel to the main chain (as explained above) with its own consensus mechanism and security assumptions. On the contrary, the assets and their states are secured via smart contracts on the main chain in non-custodial side chains. Two major classes in non-custodial side chains are Commit (and, Plasma) chains (discussed in Section III-B1) and Rollups (discussed in Section III-B2).

1) Commit chains: Channel-based solutions, such as PCN, require participants to open dedicated channels, where funds are locked within the channels and can not be reused elsewhere unless withdrawn from channels, participants must remain online for fund reception, etc. Commit chain [58, 154] were introduced to address such issues present in channel-based scalability solutions. As shown in Fig. 8, commit chains employ a non-custodial operator that initializes and maintains a commit chain while a smart contract prevents the operator from misbehaving.

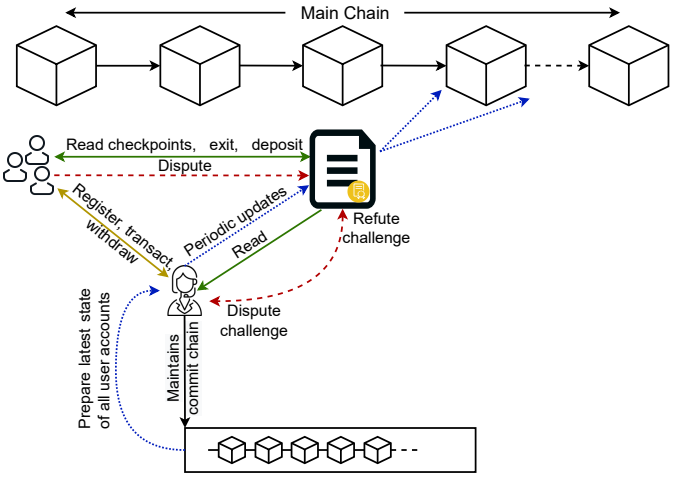


Fig. 8. An overview of commit chain operations.

Lifecycle: Participants willing to join the commit chain first register with the operator to get an account ID on the commit chain. Participants lock funds via the smart contract; the operator reads and updates corresponding account IDs on the commit chain. Recipients do not need to deposit any funds. To transfer funds, the sender authorizes the operator to deduct its account. The operator processes the transactions among the participants off the chain. The operator also periodically commits the latest state of participants' account balances to the main chain through the smart contract using constant-

sized checkpoints. The participants observe the checkpoints and challenge the operator via the smart contract in case of a dispute. The smart contract penalizes the operator if found misbehaving, and it also halts the commit chain to recover the balances from the last known stable checkpoint. Finally, a participant can withdraw funds by submitting a withdrawal request to the operator, or it can force exit with the help of the smart contract to close and refund all its account IDs.

Advantages: Registering with a commit chain requires no on-chain transaction. Though participants are advised to come online periodically to observe checkpoints, they still receive funds while being offline like any on-chain transactions. Without collateral from an operator, commit chains offer eventual finality. Such an attribute is useful from the operator's perspective. However, an operator may also choose to insure transactions by staking collateral to provide instant finality. Unlike typical side chains, a commit chain is dependent on its parent chain's consensus mechanism, which provides it the same level of security as its parent chain.

Limitations: Although the non-custodial operator is kept in check by the smart contract, it is still the single point of failure. Another issue is that the participants should maintain the commit chain data, which is not published on the parent chain while creating checkpoints, to challenge the operator and exit the commit chain [58].

Plasma chains: Another related concept is Plasma [155] chain. Plasma chains have critical limitations and issues compared to commit chains. Hence, we briefly explain its key concept and concerns. We refer the interested readers to the work [155] for a detailed description. Commit chain implementations, such as NOCUST [58], are account-based systems. Plasma chain proposes a UTXO-based ledger system running over an account-based blockchain, e.g., Ethereum. Plasma enables multiple blockchains to exist as branches of a tree with the help of a series of smart contracts. Each branch (i.e., blockchain) can have sub-branches (i.e., child chains). Each Plasma chain maintains its own block validation mechanism, which can be independent of its parent chain. However, all computations in the hierarchy of chains are globally enforced/dependent on one single root chain. Plasma chains suffer from several issues, such as steadily growing data storage costs, high computation requirements, and no native support for instant finality.

2) Rollups: Rollups are non-custodial side chain solutions that aim at reducing the load on the main chain. Rollups employ data compression techniques along with a smart contract for scaling *Layer-1* chains. The concept is analogous to Plasma chain, except Rollups retain minimal data (in the form of a Merkle root) on-chain about state updates. Such data facilitate on-chain verification and faster withdrawals. The transactions execute off the chain in batches and are bundled together for on-chain verification. In particular, the smart contract maintains the Merkle root (referred to as state root, cf. Fig. 9) on-chain from the current state (e.g., individual balances) of the Rollup. The same root can also be computed/verified from the data available on-chain. However, the Merkle tree is not stored on-chain to save space. A new state root is computed after a batch of transactions induces balance updates. Anyone can publish the batch by including the transactions in a compressed form, the previous state root, and the newly computed state root. If the current state root in the contract matches the previous state root mentioned in the new batch, the contract updates its state root to the new state root. If a batch requires external inputs (or outputs), the required funds are transferred to the contract before processing (or sent to outputs after processing) the transactions.

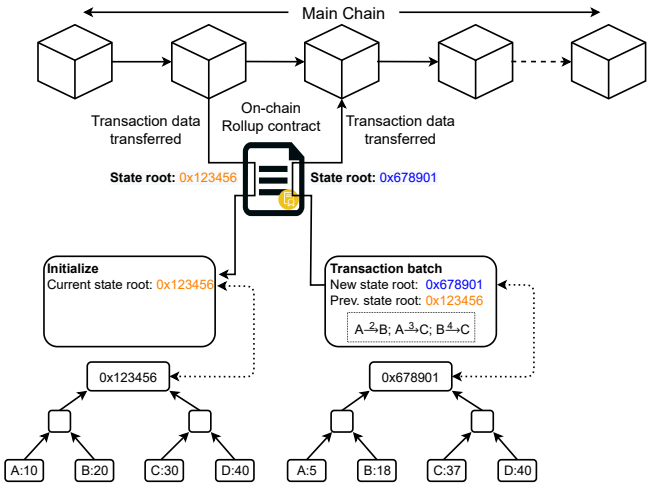


Fig. 9. An abstract presentation of Rollups.

Since anyone can publish the batch of transactions, the process of preventing frauds and verifying the new state root leads to two types of rollups, namely Optimistic and Zero-Knowledge (zk) Rollups.

- *Optimistic Rollups:* Such Rollups take an optimistic approach and assume that transactions are valid unless challenged. Thus, no computation for verification is done by default to significantly improve scalability. However, the contract maintains a history of state root updates along with corresponding batch hashes. Challenging a batch requires publishing a proof of incorrect computations (called fraud proofs) on-chain, which is verified by the contract. Upon verification, the contract reverts the incorrect batch along with all subsequent batches.
- *zk Rollups:* Contrary to Optimistic Rollups, zk Rollups suspect every transaction. Every batch contains a cryptographic proof (called validity proof), which proves that the new state root indeed matches the output of executing the batch of transactions. Such proofs are constructed using zk-SNARK and PLONK protocol [156]. Computing validity proofs is complex, but their on-chain verification happens quickly.

Advantages: Rollups use compression to reduce transaction footprint on-chain that saves space and scales *Layer-1* chain. Another key advantage is their ability to bypass the data availability problem [157] with fraud proofs. Optimistic Rollups are suitable for general-purpose computations while zk Rollups are fit for simple payment scenarios.

Limitations: The net throughput of Optimistic Rollups is limited. On another side, the cost and complexity of computing validity proofs for zk Rollups is high.

C. Cross Chains

A huge number of fundamentally different blockchains - in terms of consensus, goals, features, etc. - emerged after the success of the Bitcoin blockchain. Apart from scalability, another major issue with a multitude of blockchains is interoperability. Interoperability not only propels application portability and flexibility further, but it can also assist in improving blockchain scalability by offloading transactions from one blockchain to another [158].

Cross chains are used to transfer assets between different blockchains. Basically, they facilitate a communication medium between different independent blockchains. Different blockchains have different consensus mechanisms. Transferring assets between a blockchain with weak consensus to a blockchain with strong consensus can lead to potential safety risks. Cross chains help in establishing a mutual trust procedure between users on different blockchains who are interested in swapping assets. In essence, cross chains act as the intermediary platform for inter-blockchain transactions [159, 160].

Celer Network [69] employs a layered architecture with clear abstraction for quick development of generalized state channel as well as side chain suites to support rapid off-chain state transition. Celer network sits between the blockchain and the decentralized applications ("dApps"). Its main component, cStack, is an off-chain technology that supports different blockchains. cStack consists of three main components: cChannel, cRoute, and cOS. cChannel comprises generalized state channels for transactions. cRoute handles routing. cOS is the core component that handles several tasks including resolving dependencies between on-chain and off-chain states, node failures, unified implementations of on-/off-chain modules. Celer network aims to bring Internet scale to every blockchain while supporting high availability and stable liquidity. Other earlier cross chains are based on HTLC [71], relays (or, side chains) [73, 74], smart contracts [70]. However, there are two state-of-the-art approaches to realize cross chains, i.e., notary scheme [75, 76] and blockchain of blockchains [77–81].

1) Notary schemes: A notary is an entity that actively observes multiple blockchains and listens for transaction events, such as smart contract execution on a chain. It creates a transaction in a chain when a corresponding event happens on another chain. Crypto exchanges, e.g., Binance and Coinbase, are examples of such notary schemes. In practice, exchanges maintain order books to match sellers and buyers. Here, two distrusting parties form an agreement indirectly through the notary. Exchanges that handle and execute trades on behalf of a customer - by holding the customer's private keys - are centralized exchanges while typical decentralized exchanges only offer match-making services. Decentralized exchanges, such as 0x [75], take a smart contract-based approach (called automated market makers) to constitute a real-time price-adjustment system that replaces the on-chain order books.

2) Blockchain of blockchains: Blockchain of blockchains, or simply the Internet of blockchains, prioritizes customizability along with interoperability. The general idea here is to build an ecosystem, where independent blockchains can share data and/or tokens with each other via a backbone chain. The backbone chain only facilitates a platform for inter-chain communication programmatically and does not act as the central entity. The customizability perspective emphasizes shortening the blockchain development cycle from years to months [79]. To summarize, a blockchain of blockchains provides a platform for reusing network, data, incentive, consensus, and layer of contracts to tailor customized, application-specific, interoperable blockchains. Two prominent solutions in this domain are Cosmos [79] and Polkadot [81].

Cosmos creates a decentralized network of independent blockchains. These blockchains are called "zones", where each zone can have its own constraints on its assets. To enable transactions between zones, Cosmos follows a bridge-hub model. There are multiple hubs present in the network, and each hub can connect multiple zones. Hubs help in reducing the number of connections required to connect different zones. Registering with a hub enables a zone to communicate with all the other zones connected via this hub. Zones communicate with each other only via hubs using Cosmos's Inter-Blockchain Communication (IBC) protocol. A hub acts as a mutually trusted intermediary that enables zones to share updates regarding their states with other zones. Polkadot introduces globally coherent dynamic data structures called "parachains" hosted in parallel while the main chain is called the "relay" chain. State transition validation is performed by relay-chain validators. Polkadot defines Cross-Consensus Message Format (XCM) and Cross-Chain Message Passing (XCMP) for inter-parachain communication. Polkadot supports connecting hundreds of parachains directly to the relay chain, but only for a short to medium-term. Long-term connections and nested parachains are still under development. A detailed comparison of different cross chain solutions can be found in the work [158].

D. Hybrid solutions

Hybrid solutions help in further improving the scalability of off-chain protocols. These solutions are called hybrid because they change a few fundamental properties of off-chain solutions. We identify two categories of such solutions, where one aims to reduce on-chain dependence of dispute resolution mechanisms while the other uses secure execution mechanisms to eliminate trust requirements among peers. The former category is called bisection protocols (discussed in Section III-D1) while the latter (explained in Section III-D2) is implemented using Trusted Execution Environments (TEE).

1) Bisection protocols: Existing dispute resolution mechanisms in off-chain solutions typically execute on-chain. Thus, these solutions are not purely off-chain, at least from the dispute handling perspective. Bisection protocols form a branch of *Layer-2* solutions that primarily aim at improving the dispute resolution mechanism. These protocols take part of the computations off-chain, thus helping to reduce the load on the main chain. Generally, bisection protocols involve two steps. First, a user presents minimal evidence to a verifier to testify the validity of its transaction. Next, when users contradict each other, a verifier inspects evidence from contradicting users to determine the correct state. Truebit [83] and Arbitrum [82] employ such dispute resolution mechanism.

Truebit was introduced as a blockchain enhancement to improve the computational efficiency of smart contracts at a reduced cost. It ports computations from the main chain to off-chain. Truebit depends on *judges*, who have limited computational power. These judges are mutually trusted by all the participants. Given a computational problem, the user who solves it, called the *solver*, publishes the solution as well as the sub-problems used to reach the solution. A challenge period is set aside during which other users can challenge the published solution - such a user is called a *challenger*. If a challenge is raised, the judges solve the sub-problems recursively using binary search; this is to reduce the problem size by half in each iteration. The solution provided by the judges is mutually agreed to be the correct solution by all participants. The judges compare their correct solution with the solutions provided by the solver and the challenger to identify and penalize the cheating participant.

On another side, Arbitrum uses a Virtual Machine (VM) to implement a smart contract. A user can create a VM and designate other users as VM managers. An honest manager enforces the VM to follow VM's coded functionality. Any change in the state of the VM has to be approved by all the managers. There might be scenarios where managers do not agree with each other about the state of the VM. Such disagreements should be raised within a challenge period after a new state has been committed. In case of a conflict among the managers, the verifiers/miners invoke a bisection protocol to reduce the conflict down to single-instruction execution. Now, managers present their outcome for that single-instruction execution. The verifiers can verify the presented outcomes efficiently to identify and punish the cheating participant.

2) TEE-based solutions: A trusted execution environment, e.g., Intel SGX [161, 162], is typically an isolated and safeguarded area inside a CPU, where the integrity and confidentiality of loaded data are protected. TEE-based solutions for blockchain scalability utilize integrity protection offered by TEEs to eliminate the on-chain collateral used for establishing trust among participants. In fact, TEE is used as a mutually trusted entity in such solutions because they offer a higher level of security for application execution.

Teechan[86] uses TEEs to enable two mutually distrusting nodes to transact with each other. Here, a channel is set up between the two nodes by exchanging secrets via their TEEs. As long as the channel is open, the nodes can exchange funds with each other in a peer-to-peer manner using TEE-supported operations; even without involving the parent Bitcoin blockchain. The TEEs bear the responsibility to maintain and update the channel's state securely throughout the channel's lifetime. Upon channel termination, the TEEs create a Bitcoin transaction to be added to the parent chain. During the entire lifetime of such a channel, only two transactions reflected on-chain; one for channel establishment using a 2-of-2 multisig Bitcoin address and the other for channel closure. To summarize, Teechan reduces the load on the parent chain and increases transaction throughput among distrusting nodes.

Teechain [85] is another such solution that executes off-chain transactions asynchronously with the main chain. Teechain employs TEE-protected *treasuries* to preserve the correct channel state. Teechain forms a chain of committee that holds replicated states of treasuries to handle treasury failures.

Some other prominent solutions leveraging features of TEEs are Tesseract [87], BITE [84], and ZLiTE [88]. Tesseract is a TEE-based cryptocurrency exchange, BITE focuses on the privacy of Bitcoin lightweight clients, and ZLiTE improves the privacy of Zcash lightweight clients by involving TEE-equipped servers. Nevertheless, all TEE-based solutions rely upon the integrity of the TEEs while TEEs have their own vulnerabilities and concerns [163, 164].

IV. Network issues

Layer-2 protocols are built on top of *Layer-1* blockchains. Enabling participants to transact with each other off the chain may require an overhead communication network to help, for instance, finding a payment path between participants that do not have a direct connection. Routing, re-balancing, stability, and privacy are the key concerns related to such overhead networks. We take the example of channel-based solutions to briefly discuss these issues. We refer the interested readers to works [43, 44] for an in-depth analysis of network management issues and a comparison of routing algorithms.

1) Routing: For a transaction to occur between distant participants in a network, a payment path must be established from the payer to the payee involving intermediate nodes. Deciding such a payment route involves various factors, i.e., inter-node channel capacity, length of the route, cost-effectiveness, and availability of the nodes participating in the transaction. Several works on routing algorithms have attempted to find the most efficient path for the transactions. These routing algorithms can be broadly classified into two categories, namely, global routing (e.g., [165] and Di Stasi et al. [166]) and local routing (e.g., SilentWhispers [167] and SpeedyMurmurs [168]). The source of the payment in global routing makes use of the global view of the network to find the optimal path. The performance of such source routing depends on the accuracy of the available global view. By pre-computing paths, these routing algorithms attempt to minimize the overall communication overheads and latencies. However, their scalability is limited by the cost of maintaining an accurate global view and path computations on the source. On another side, local routing algorithms utilize the local information and take a greedy approach. Local routing algorithms are inherently scalable but are less efficient due to local optimizations.

2) Re-balancing: Exhaustion of capacity in payment channels is a recurring problem. A naive and inefficient solution is to close the channel and refund it, which results in at least two transactions. Protocols like REVIVE [149] allow nodes to safely rebalance their skewed channel with the help of funds available in their other existing channels. An untrusted third party, called a leader, is elected to execute the channel rebalancing process. After receiving nodes' requests and preferences, the leader coordinates with the nodes to freeze relevant channels. The frozen channels are now rebalanced using linear programming and commitments signed by all the nodes. After the process, the nodes lose funds from their one or more payment channels to gain equal funds on the others.

3) Stability and privacy: Initial *Layer-2* solutions, particularly channel-based, require participants to remain online to monitor transactions and handle disputes. To address such a limitation, participants can engage watchtowers to detect discrepancies on their behalf while they are offline. Watchtowers get a fee for their services. On the contrary, watchtowers are penalized from their collateral for failing to report disputes [176]. A key concern with watchtowers is that a watchtower may be bribed to cheat its customers for mutual gains. If the bribe is higher than the collateral, the watchtower may act dishonestly. On another side, routing payments over a network of nodes may lead to privacy issues, such as disclosing information about the payer and payee. Malovolta et al. [145] attempt to formally define the privacy requirements over PCNs and implement Fulger and Rayo using Multi-Hop HTLC smart contract to handle transaction concurrency. However, a routing protocol that learns the capacity of payment channels over a period of time can bypass their privacy notions.

V. Security and privacy issues

In this section, we discuss the security and privacy attacks on different *Layer-2* protocols. TABLE I presents a summary of all such attacks present in the literature. We clarify their setup requirements, impact, affected parties, and key prevention/mitigation techniques.

A. Wormhole attack

Wormhole attack [147] steals rewards of PCN intermediaries. A payment is typically relayed through multiple intermediaries in PCN since a direct channel may not exist between payer and payee. In such a scenario, the intermediaries are aware of their immediate neighbors and may not be aware of other nodes on the path, sometimes even about the payer and payee. An attacker with just two malicious nodes on the path from the payer to payee can launch wormhole attack. The name wormhole denotes how the funds are rerouted through a wormhole channel between the malicious nodes of the attacker.

In PCN, the payment is locked in a cryptographic lock, i.e., HTLC (cf. Section II), which is routed to the payee. Each intermediate node that relays the payment retains a small fee, which is the difference between the amount it will receive from the channel with previous node and the amount it will pay to the channel with next node. Once the payee reveals the key to the lock, it is routed back to the payer. While the key travels back to the payer, it sequentially unlocks funds in the channels on the path. However, a malicious node can bypass the key to another malicious node, skipping all the nodes/channels between these two malicious node. As a result, the benign nodes assume that the transaction has failed and return to their original state while the malicious node steal the reward from the benign nodes.

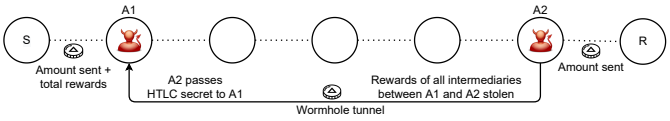


Fig. 10. Wormhole attack stealing rewards of intermediaries.

Fig. 10 depicts a scenario for wormhole attack. For higher impact, the attack requires the malicious nodes to be closer to the sender (S) and receiver (R). To make a payment from S to R, S locks the original payment amount along with total reward, i.e., sum of fees for all intermediaries. Out of all the nodes which agreed to participate in the payment, two nodes A1 and A2 are malicious nodes. When R reveals HTLC secret to A2, it pays the original payment amount to R. Now A2 bypasses the HTLC secret to A1. A1 uses the HTLC secret to receive the sum of original payment and the total reward. Thus, A1 and A2 stole the rewards by colluding and bypassing the intermediate nodes, which will return to their original state assuming that the transaction has failed.

Countermeasures: The main reasons behind wormhole attack are: (1) the same HTLC secret is used to unlock funds from each channel on the payment path, and (2) each channel can be unlocked independently. Anonymous Multi-Hop Lock (AMHL) [147] communicates path-specific secret information to nodes and makes locks interdependent to ensure that the funds are unlocked only in a hierarchical manner.

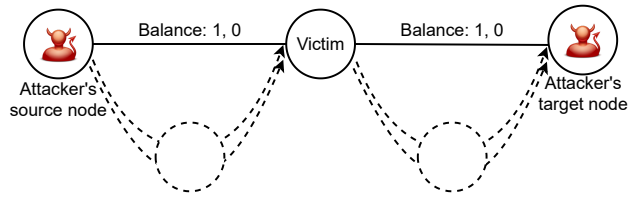
TABLE I
A summary of major attacks on *Layer-2* solutions.

Attack	Impact	Affected parties	No. of adversary nodes needed	Setup and launch	Key prevention/mitigation approaches
Wormhole [147]	Intermediaries' funds are temporarily frozen; incurs loss of useful time; transaction processing reward is stolen.	All nodes along the path between adversary nodes.	Two	Adversary sets up an additional round of communication to share and bypass the HTLC secret.	AMHL offer interoperable, secure, and privacy-preserving cryptographic construction that works in both script-based and scriptless.
Flood and loot [169]	Attacker claims disputed transaction; exploits replace by fee policy.	All nodes that agree to open channel with attacker's source node.	Two	Adversary establishes several channels through victims and sends a multitude of payments. While settling payments attacker's source node forces victims to create several blockchain transactions all at once.	Reduce the maximum number of unresolved HTLCs; allow more time (blocks) to claim HTLCs on blockchain; use anchor commitment output; use non-replaceable HTLC transactions.
Griefing [37]	Network stalling; capacity exhaustion; may incur channel force-closing fee; eliminating competitors from network.	All nodes participating in the payment path towards attacker.	One	Adversary refuses to resolve payments off-chain, locking victims' funds for entire duration of contract.	Limit the number of incoming channels; faster HTLC resolution; constant payment locktime; incentivizing/punishing nodes; griefing penalty.
Time dilation [170] (eclipse)	Victim isolated from the network; feeds blocks to the victim at a slower rate; funds can be stolen.	Primarily, trust-minimized Bitcoin light clients with limited connections.	Multiple	Adversary deploys hundreds of Sybil nodes, opens a payment channel with the victim, then eclipses/time-dilates the victim.	Increase adoption of BIP 157 [171]; anonymize peer-to-peer protocols; engaging watchtower.
Balance lockdown [172]	Adversary gets a dominant position; blocks the victim's ability to act as an intermediary.	Middle nodes in multi-hop payments. Typically, it targets a single node that relays many payments.	One	Adversary aims to disrupt availability of a victim, opens a channel with the victim, sends self-destined payments that go and come back via the victim.	Increase AER to reduce attack profitability; reduce the maximum length of a route; minimize/forbid loops in a payment route.
Balance discovery [173, 174]	Balance between a pair of victim nodes disclosed; nodes' privacy compromised.	The pair of nodes targeted by attacker.	One	Adversary opens a channel with one of the two victim nodes. It tries to disclose the balance between victims by routing invalid payments. The value of payments typically follows a binary search pattern.	Adhere to protocol specification and prevent payments with values higher than maximum allowed limits; clients should resist closing a channel upon receiving malformed payments.
Congestion [175]	No monetary gains; stalls or paralyses network; DoS affects competitors' gains.	All nodes participating in the transaction between attacker's source and destination node.	One	Adversary overloads channels with unresolved requests to block high liquidity channels and disconnect/isolate individual/pair of nodes.	Avoid paths with loops; enforce fast HTLC resolution; reduce route length; limit the number of maximum concurrent payments.

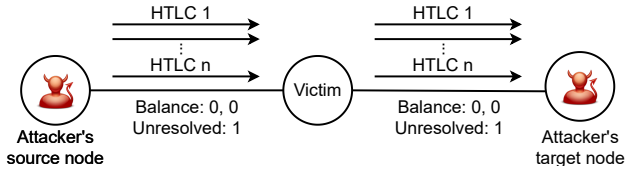
B. Flood and loot

Flood and loot [169] is a systemic attack on the Lightning network. The attacker controls two nodes in the network, where one node acts as the source of the payment while the other node is the destination of the payment. Nodes that participate in forming channels between source and target nodes are affected by flood and loot attack. The key idea of the attack is to trigger closing of multiple channels simultaneously. In such a scenario, the victim nodes can only resort to the parent blockchain to claim their funds locked into corresponding open HTLCs. Thus, victim nodes benignly overload the blockchain, which opens a window of opportunity for the attacker to steal the funds. In particular, the attack leverages the replace-by-fee policy [177] employed in Bitcoin blockchain.

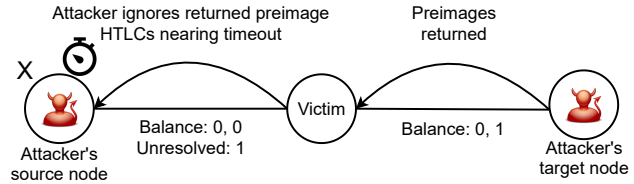
The attacker's source node opens multiple channels through the victim nodes to the attacker's target node. The attacker initiates multiple HTLC payments from the source to target node, accept these payments at the target node, but refuses to resolve them at the source node. As the timeout for HTLCs approach, victims are compelled to close the channels with the source node and claim open HTLCs on the blockchain.



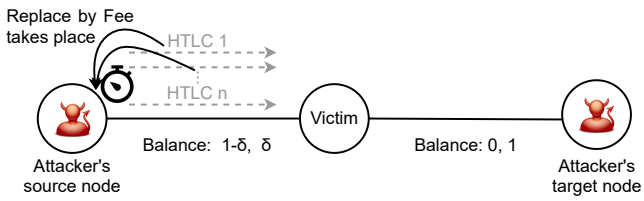
(a) Establishing channels



(b) Excessively loading channels with HTLCs



(c) Resolving payments on the last hop



(d) Waiting for all HTLCs to expire (approx. the same time) and collecting unspent outputs

Fig. 11. Different stages of a flood and loot attack.

Importantly, a node can become a victim of flood and loot attack only if it opens a channel with the attacker's source node. Fig. 11 depicts a different steps of a simplified flood and loot attack. In the first step, the attacker establishes a number of channels from source node to target node via victim nodes. It is worth mentioning that a given victim can be participating in multiple distinct paths from source to target, which is a much worse for such a victim node (see Fig. 11(a)). Now, source node commences (preferably when blockchain fees are low) multiple HTLC payments to the target node using the channels previously created (see Fig. 11(b)). The aim is to use as many channels as possible with maximum funds utilization. Next, the target node acts honestly, reveals the secret to the victim node, settles HTLC, and gets the payment (see Fig. 11(c)). The victim nodes forwards the secret to the next node in the chain, which is source node in this case. Source node refuses to respond, leaving open HTLCs in the channels (see Fig. 11(c)). As the timeouts for HTLCs approach, the victim node tries to close the channels with the source node to claim all open HTLCs on the blockchain. Consequently, numerous transactions towards the parent blockchain are induced simultaneously (see Fig. 11(d)). Not all transactions will enter the blockchain due to congestion. At the same time, the replace-by-fee protocol will allow transactions offering higher fees to replace other conflicting transactions. Thus, the attacker raises the fee for his transactions to exceed those of the victims to claim the funds using replace-by-fee policy.

Countermeasures: There are two prominent countermeasures for this attack. The first one is to limit the maximum number of HTLCs that can remain unresolved at any point of time. This limitation will decrease the number of transactions competing to include in the blockchain in a given window of time. So, the attacker will require more victims to trigger the race conditions necessary for the attack. The second one is to use a reputation scores, where channel opening requests from unknown counterparts is given a lower priority, which reduces the scope of locking larger funds in such channels. Another solution is to use anchor commitment outputs [178] that manipulates the way transaction fees are paid. However, anchor outputs remain ineffective for an attacker with sufficiently large capital.

C. Griefing attack

Unlike most of the other attacks on *Layer-2* protocols, griefing attack [37] is not directly motivated by stealing funds or information. It instead focuses on stalling payment networks by exhausting the channel capacity. It steals useful time of benign participants by preventing them from processing further transactions, which eventually leads to temporal loss of funds, decreased network throughput, and routing disruption.

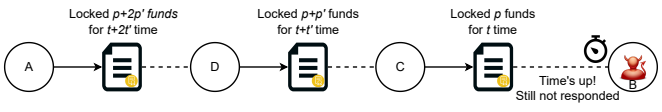


Fig. 12. Griefing attack to stall network and participants.

Fig. 12 illustrates a representative griefing attack setup, where A is transferring p funds to B over a payment path A→D→C→B, and each intermediate node charges a processing fee p'. A forwards a conditional payment to D using an

off-chain contract, where it locks $p + 2p'$ funds for a time period $t + 2t'$. Here, t' indicates deadline for confirmation time for on-chain settlement. D forwards the payment to C using another off-chain contract locking $p+p'$ funds for a time period $t+t'$. C forwards the payment to B using another off-chain contract locking p funds for a time period t. In order to claim p funds from C, B must resolve the payment within time period t. Otherwise, C can claim a refund on-chain by closing its channel with B; similar procedure to be followed by D and then A. Nevertheless, C can go on-chain only after its contract with B expires. As a result, B can block $\mathcal{O}(p)$ funds in each of the preceding contract for a time period t without losing any funds. It is worth mentioning that the order of t can be days for genuine users' convenience. Thus, each of the involved nodes can not utilize their funds for the entire duration of t. The attacker can stall the network by simultaneously launching multiple griefing attacks from its single or multiple nodes.

Countermeasures: Apart from limiting the number of incoming channels, faster contract resolution, and constant payment locktime, incentivizing/punishing nodes is a crucial way to tackle griefing attack. Griefing penalty [179] maneuvers along the same idea, which punishes the attacker to pay a penalty for compensating the victims' lost time. This way it prevents nodes from ever attempting to participate in such an attack.

D. Time dilation/eclipse attack

Time dilation attack [170] dilates a victim's clock. As a result, the victim always remains late in becoming aware of the new blocks in the chain. In other words, the victim misses updates from the network and keeps on working with outdated information. To this end, the attacker eclipses a victim node such that it cannot participate in the network owing to the delayed block delivery. Although time dilation attack is an expensive attack, it yields high returns in the form of stealing the total channel capacity in a single eclipse period. Authors in [180, 181] propose channel exhaustion and node isolation attacks along similar avenues. For the sake of brevity, we discuss only time dilation attack. Fig. 13 depicts

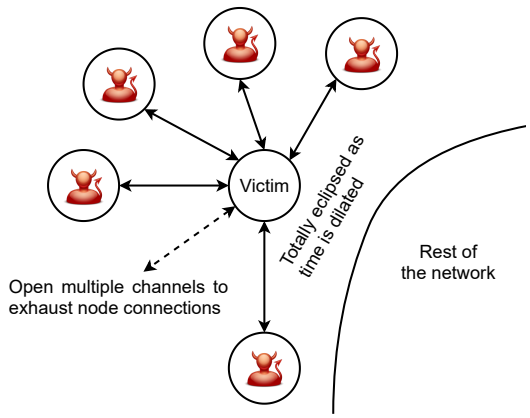


Fig. 13. Eclipse attack where attacker dilates a user.

multiple malicious nodes eclipsing a victim node. Typically, such an attack focuses on single victim node. Essentially, eclipsing a user means preventing it from observing current network activities or state and cutting their communication with other honest users.To do so, the attacker must occupy every connection a victim can have using pseudonymous nodes with pretend identities. After eclipsing a victim, the attacker induces time dilation by introducing a delay between the time it receives a block to the time it will feeds the block to the victim node. Time dilation attack is commonly seen in Lightning networks and can be classified - based on the target of the attack - into the following three main categories:

- *Target channel's state finalization:* It pushes the victim's observed block height several blocks behind the tip of the main chain, typically by negotiating a new state committed to an outdated chain state.
- *Target per-hop delay:* Once the attacker gets ahead of the victim's block height by dilation, it routes a payment through the victim, and at the same time finalizes the state of their channel onto the blockchain. It stops the victim from renegotiating through preimage disclosure, leaving no option and funds with the victim.
- *Target packet finalization:* When a victim node knows the preimage for an incoming contract, but the remote peers do not respond on time, the victim node goes on-chain a few blocks before the expiration time to claim the contract. But, a dilated victim is bound lose such a race despite responding ahead of time.

Countermeasures: A solution to time dilation attack must detect as well as mitigate the attack. Detecting whether a block was delayed in being sent or delivered is complex due to high rate false positives. Nevertheless, abnormal routing failure rate could be an indicator for detection. Similarly, mitigating the attack is also not straightforward mainly because identifying attacker committed state or choosing the right channels from multiple open channels is difficult. Watchtowers [132, 136–138] act as a substitute to built-in defense mechanism, but they come with extra assumption on their honesty and efficiency.

E. Balance lockdown attack

Balance lockdown attack [172], also known as balance availability attack, affects the ability of a victim node to successfully participate in payment routing. The attack impedes the victim node from taking part in any further transactions by blocking their balance funds. Such an attack confers a dominant position to the attacker, as well as reduces system's efficiency. In particular, it enables the attacker to block certain payments paths, giving a competitive advantage to attacker favored paths. The attacker has precise knowledge about network topology using which the attacker routes a payment via the victim node.

A multihop payment in a channel network is atomic in nature, which means that the fund transfer can happen only after establishment of the complete payment path. Thus, intermediate nodes on the payment path need to keep their funds locked. An attacker can lock an amount p on each victim node on the payment path just by sending a payment of p through them. It is worth noting that the cost of sending the payment and the time taken for the completion of transaction

are the two key factors that decide the feasibility of this attack. Hence, the attacker aims to increase the completion time for the transaction as well as to minimize the cost of the transaction. Authors in [172] show that the overall effect of the attack can be captured by a parameter called Attack Effort Ratio (AER), which is defined as the ratio of capacity required to launch the attack and the capacity that the attack effectively blocks (cf Eq. 1).

$$AER = \frac{Capacity_{Required}}{Capacity_{Blocked}} \quad (1)$$

Another parameter, called Total Blocked Time (TBT), to measure attack's effectiveness is defined as the amount of time for which the funds of victims is blocked in the transaction.

Countermeasures: An optimal solution to handle balance lockdown attack should aim at increasing AER and reducing TBT. AER can be increased by disallowing loops in payment routes to minimize attacker's profitability. Another way to increase AER is to limit the maximum length of payment routes, though limiting route length may have impact on the performance. Regulating TBT is tricky and has to traded-off with AER. Nevertheless, such trade-offs need detailed studies.

F. Balance discovery attack

Balance discovery attack [173, 174] affects PCN, in particular Lightning network. A PCN comprises of payment channels with a fixed deposit known as the channel capacity. The total capacity of a channel is publicly known. However, the individual balances on either side of a channel are not disclosed to preserve the privacy of users participating in the channel. Balance discovery attack aims to disclose individual balances of users, thus compromising users' privacy. Any node creating a channel with a malicious node is vulnerable to this attack. Moreover, the attack automatically extends to all the immediate neighboring nodes of that victim node.

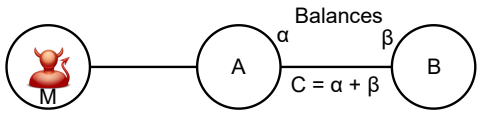


Fig. 14. Balance discovery attack where attacker tries to disclose the balance between A and B.

Fig. 14 illustrates the setup for balance discovery attack. Here, two nodes A and B form a channel with total capacity C. We can safely assume that A and B perform some transactions via their channel over the period of time, and their current balances in the channel are α and β, respectively. It is important to mention that the total channel capacity remains unchanged, i.e., $C = \alpha + \beta$. A malicious node M wants to disclose α and β. To do so, M first creates a channel with A. This channel also facilitates M with a path to B through A. Now, M sends a payment p to B through A. As long as p is less than α, it can be sent successfully through the route M→A→B. M increases the value of p and sends a new payment to B. M repeats this process until an error occurs. The last successful payment value p reflects the approximate balance of A while β can be computed by subtracting α from publicly known C. As an improvement, the value of p can be efficiently calculated by using a binary search on lower and higher payment bounds. Furthermore, M can also send fake payments for probing such that none are finalized. Such strategy helps in reducing cost of the attack.

Countermeasures: One straightforward countermeasure is to limit the maximum allowed payment value. It would increase the efforts required by the attacker, but it would also affect the overall functionality of the system. The key approaches to tackle balance discovery attack are dropping rate parameter and dynamic absorption of negative balances [173]. The former suggests each node to randomly deny a specified percentage of transactions without disclosing the cause of failure to the payment sender. Such a strategy will deceive the attacker to assume that the payment has failed due to a lack of funds. Consequently, the attacker will interpret the wrong values of the balances. The latter advocates absorbing negative channel balances to prevent accurate probing of channel's remaining capacities.

G. Congestion attack

Similar to balance lockdown attack, the goal of congestion attack [175] is to block victims' funds at a lower cost. The attack targets PCNs and obstructs several payment channels at once for a long period of time. It exploits the trustless payment mechanism and long HTLC expiration duration. In particular, the attacker acts as both the source and destination of a payment. It can establish multiple such payments for wider coverage of the network. The attacker then sends funds along each of such routes, blocking nodes on each route.

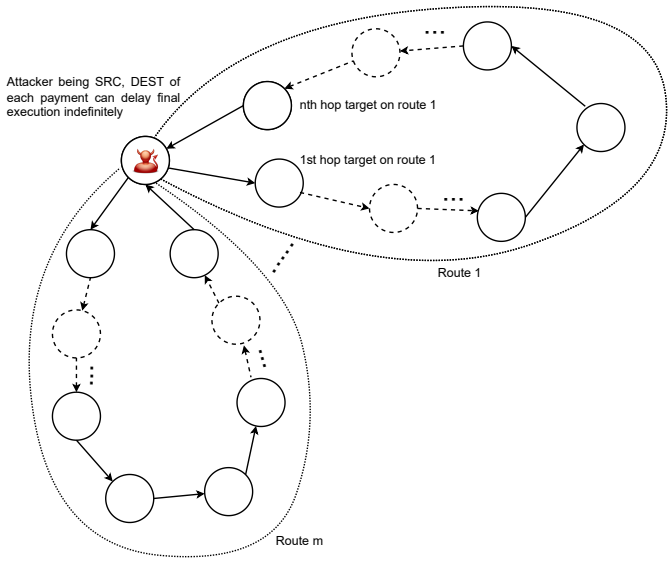


Fig. 15. Congestion attack over m payment routes.

Fig. 15 depicts a congestion attack on m payment routes, where all nodes along the route 1 to m are blocked until the attacker's transaction completes. After joining the network, the attacker evaluates different routes considering maximum route length, available funds, contract expiration limits, etc. Next, it establishes channels with two nodes along a target route such

that one of the nodes sees the attacker as the source while the other nodes sees the attacker as the destination of the payment. The attacker can delay a transaction on such a route because it is both the source and destination of the payment, which leaves the nodes being blocked. Right before the HTLC expiration time, the attacker cancels the transaction and restores the channels to their original state. At this point, it can relaunch the attack to block the same path till the next HTLC expiry, which be in the order of weeks. Importantly, this attack can be used in three ways: (1) to block high liquidity channels, (2) to isolate multiple node pairs, and (3) to disconnect individual nodes from the network.

Countermeasures: A simple approach to increase the number of HTLCs supported by benign nodes would remain ineffective as the attacker can take over all HTLCs with a larger volume of payments. Nevertheless, the effect of the attack can be limited by reducing the maximum allowed route length. Furthermore, the nodes can regulate the number of maximum concurrent payments with a peer based on its behavior.

VI. Discussion

Different *Layer-2* solutions aim to scale blockchains while offering distinct functionalities based on different principles and underlying technologies. In this section, we compare the key categories of *Layer-2* solutions. We have omitted cross chains and hybrid solutions as properties of cross chains implementations are highly dependent on chains targeted for interoperability while hybrid solutions have diverse hardware requirements. Thus, generalizing these categories may not truly reflect their characteristics. We compare these solutions over multiple attributes that can be organized into four categories, which are performance, temporal requirements, security considerations, and miscellaneous options. TABLE II presents a summary of the comparisons.

In terms of performance, channels offer faster transaction processing (i.e., almost instant *Layer-2* transfers) at a meager cost of transactions. However, channels require locking collaterals on-chain, while other solutions may not enforce this requirement. In particular, each channel must have collateral locked to start the operations. Thus, other solutions are perceived as more capital efficient than channels.

Withdrawing balances from channels requires just one on-chain confirmation. Commit chains and Optimistic Rollups typically elongate withdrawal time for balance security and dispute resolutions. Such longer durations can be shortened by introducing risk insurers or liquidity providers. However, the reliability of such insurers has its own concerns. Withdrawing the balance on *Layer-1* from zk Rollups takes fewer minutes in the best case. The finality of a transaction reflects a state, where the transaction can not be reverted on the main chain. The underlying principles of channels offer instant finality while other solutions typically offer delayed finality. Nevertheless, instant finality can be tweaked on other *Layer-2* solutions as well to reflect instant confirmation to user, but full security guarantees remain exclusive to channels.

Most of *Layer-2* protocols use standard cryptographic primitives, SNARKs and STARKs are heavily used in zero knowledge based protocols. User liveness assumption is critical, which can be defined as the requirement of a user to remain online to receive/verify transactions, monitor disputes, and handle misbehaving counterparties. Unlike channels, commit chains enable users to receive transaction while remaining offline. However, users are advised to come periodically online to inspect checkpoint commitment. This assumption about a user's online status can be delegated to a trusted third-party that monitors transactions on behalf of the user. Nonetheless, such a third-party may compromise if the incentive to misbehave is greater than its guarantee deposit. Only commit chains allow all users to successfully withdraw - mainly for security reasons - in a short period of time, and all the protocols prevent validators from confiscating funds of participants.

Both Optimistic and zk Rollups offer support importing existing EVM-bytecode with minor modifications, and thus, have a flexible support for smart contracts. Transaction deanonymization and user profiling are major privacy concerns

TABLE II
A comparison of different *Layer-2* solutions.

Criteria	Aspect	Channels	Commit Chains	Optimistic Rollups	zk Rollups
Performance and operations	Transaction Speed	Fast	Moderate	Slow	Slowest
	Transaction cost	Very low	Very low	Low	Low
	On-chain transaction for account opening	Yes	No	No	No
	Capital-efficient	No	Yes	Yes	Yes
Temporal requirements	Withdrawal time	One confirmation	Multiple days	Multiple days	<10 minutes
	Finality	Instant	One confirmation	One confirmation	<10 minutes
	Instant transaction confirmations with full security guarantees	Yes, self supported	No, parent-linked	No, parent-linked	No, parent-linked
Security considerations	Cryptographic primitives used	Standard	Standard	Standard	New
	User liveness	Yes	Yes	Delegable	No
	Mass exiting	No	Yes	No	No
	Fund freezing by validators	No	No	No	No
Support, options, and miscellaneous	Smart contract support	Yes, limited	Yes, limited	Yes, flexible	Yes, flexible
	Imports existing EVM-bytecode	No	No	Yes	Yes
	Privacy options	Limited	No	No	Full

that only zk Rollups address by default [182–184].

VII. Conclusion

Scalability is a major issue for blockchain-based solutions. Vast efforts, from both academia and industry, have been put in different directions to tackle the blockchain scalability issue. Such efforts have resulted in a rich literature of *Layer-2* protocols that primarily aim to scale underlying main chains. In this work, we first create a broader taxonomy of *Layer-2* protocols, which is followed by a detailed explanation of each *Layer-2* protocol class. These protocols bring scalability to the main chains at the cost of different security assumptions and guarantees. Thus, we discuss various issues associated with these protocols and also compare them against each other. We believe that our study offers better explanations and analyses to help the readers understand the domain better.

References

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," https://bitcoin.org/bitcoin.pdf, 2008.

[2] D. Chaum, "Blind Signatures for Untraceable Payments," in *Advances in Cryptology*, 1983, pp. 199–203.

[3] D. Chaum, A. Fiat, and M. Naor, "Untraceable Electronic Cash," in *Advances in Cryptology*, 1988, pp. 319–327.

[4] M. Conti, A. Gangwal, and M. Todero, "Blockchain Trilemma Solver Algorand has Dilemma over Undecidable Messages," in *International Conference on Availability, Reliability and Security*, 2019, pp. 1–8.

[5] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer *et al.*, "On Scaling Decentralized Blockchains," in *International Conference on Financial Cryptography and Data Security*, 2016, pp. 106–125.

[6] J. Levine, "Scalability Controversy: Understanding Past Cryptocurrency Returns through Segregated Witness," https://escholarship.org/content/qt0x670791/qt0x670791.pdf, 2019.

[7] D. Ding, X. Jiang, J. Wang, H. Wang, X. Zhang, and Y. Sun, "Lossy Block Compression with Salted Short Hashing," *arXiv preprint:1906.06500*, 2019.

[8] "Compact Block Relay," https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki, 2020.

[9] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-NG: A Scalable Blockchain Protocol," in *USENIX symposium on Networked Systems Design and Implementation)*, 2016, pp. 45–59.

[10] R. Pass and E. Shi, "Hybrid Consensus: Efficient Consensus in the Permissionless Model," in *31st International Symposium on Distributed Computing*, 2017.

[11] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol," in *Annual International Cryptology Conference*, 2017, pp. 357–388.

[12] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz, "Permacoin: Repurposing Bitcoin Work for Data Preservation," in *IEEE Symposium on Security and Privacy*, 2014, pp. 475–490.

[13] S. Park, A. Kwon, G. Fuchsbauer, P. Gaži, J. Alwen, and K. Pietrzak, "Spacemint: A Cryptocurrency Based On Proofs Of Space," in *International Conference on Financial Cryptography and Data Security*, 2018, pp. 480–499.

[14] "Sawtooth," https://github.com/hyperledger/sawtooth-core, 2019.

[15] L. Luu, V. Narayanan, K. Baweja, C. Zheng, S. Gilbert, and P. Saxena, "SCP: A Computationally Scalable Byzantine Consensus Protocol for Blockchains," *IACR Cryptology ePrint Archive*, no. 1168, 2015.

[16] "Sharding Roadmap," https://github.com/ethereum/wiki/wiki/Sharding-roadmap, 2019.

[17] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A Secure Sharding Protocol for Open Blockchains," in *ACM Conference on Computer and Communications Security*, 2016, pp. 17–30.

[18] A. E. Gencer, R. van Renesse, and E. G. Sirer, "Service-oriented Sharding with Aspen," *arXiv preprint:1611.06816*, 2016.

[19] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: Scaling Blockchain via Full Sharding," in *ACM Conference on Computer and Communications Security*, 2018, pp. 931–948.

[20] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A Secure, Scale-out, Decentralized Ledger via Sharding," in *IEEE Symposium on Security and Privacy*, 2018, pp. 583–598.

[21] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "SPECTRE: A Fast and Scalable Cryptocurrency Protocol," *IACR Cryptology ePrint Archive*, vol. 2016, no. 1159, 2016.

[22] T. Zhou, X. Li, and H. Zhao, "DLattice: A Permission-less Blockchain based on DPoS-BA-DAG Consensus for Data Tokenization," *IEEE Access*, vol. 7, pp. 39 273–39 287, 2019.

[23] L. Cui, S. Yang, Z. Chen, Y. Pan, M. Xu, and K. Xu, "An Efficient and Compacted DAG-based Blockchain Protocol for Industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4134–4145, 2019.

[24] "Bitcoin Cash," https://www.bitcoincash.org, 2008.

[25] J. Chen and S. Micali, "Algorand: A Secure And Efficient Distributed Ledger," *Theoretical Computer Science*, vol. 777, pp. 155–183, 2019.

[26] M. Al-Bassam, A. Sonnino, S. Bano, D. Hrycyszyn, and G. Danezis, "Chainspace: A Sharded Smart Contracts Platform," 2017.

[27] L. luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "Elastico : A Secure Sharding Protocol For Open Blockchains," in *ACM Conference on Computer and Communications Security*, 2016, pp. 17–30.

[28] Ava Labs, "Avalanche," https://www.avax.network/, 2018.

[29] W. F. Silvano and R. Marcelino, "Iota Tangle: A Cryptocurrency To Communicate Internet-of-Things Data," *Future Generation Computer Systems*, vol. 112, pp. 307–319, 2020.

[30] M. Hearn, "Micro-payment Channels Implementation now in bitcoinj," https://bitcointalk.org/index.php?topic=244656, 2013.

[31] "bitcoinj," https://bitcoinj.github.io/, 2019.

[32] C. Decker and R. Wattenhofer, "A Fast and Scalable Payment Network with Bitcoin Duplex Micropayment Channels," in *Symposium on Self-Stabilizing Systems*, 2015, pp. 3–18.

[33] J. Poon and T. Dryja, "The Bitcoin Lightning Network: Scalable Off-chain Instant Payments," https://lightning.network/lightning-network-paper.pdf, 2016.

[34] R. Khalil, A. Gervais, and G. Felley, "Nocust-a securely scalable commit-chain," *IACR Cryptology ePrint Archive*, no. 642, 2018.

[35] G. Wood *et al.*, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[36] Bitcoin Wiki, "Hash Time Locked Contracts," https://en.bitcoin.it/wiki/Hash_Time_Locked_Contracts, 2019.

[37] D. Robinson, "HTLCs Considered Harmful," in *Proceedings of Stanford Blockchain Conf.*, 2019.

[38] G. Yu, X. Wang, K. Yu, W. Ni, J. A. Zhang, and R. P. Liu, "Survey: Sharding in Blockchains," *IEEE Access*, vol. 8, pp. 14 155–14 181, 2020.

[39] G. Wang, Z. J. Shi, M. Nixon, and S. Han, "SoK: Sharding on Blockchain," in *1st ACM Conference on Advances in Financial Technologies*, 2019, pp. 41–61.

[40] S. Kim, Y. Kwon, and S. Cho, "A Survey of Scalability Solutions on Blockchain," in *International Conference on Information and Communication Technology Convergence*, 2018, pp. 1204–1207.

[41] A. Hafid, A. S. Hafid, and M. Samih, "Scaling Blockchains: A Comprehensive Survey," *IEEE Access*, vol. 8, pp. 125 244–125 262, 2020.

[42] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, "Solutions to Scalability of Blockchain: A Survey," *IEEE Access*, vol. 8, pp. 16 440–16 455, 2020.

[43] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais, "SoK: Layer-two Blockchain Protocols," in *International Conference on Financial Cryptography and Data Security*, 2020, pp. 201–226.

[44] M. Jourenko, K. Kurazumi, M. Larangeira, and K. Tanaka, "SoK: A Taxonomy for Layer-2 Scalability related Protocols for Cryptocurrencies," *IACR Cryptology ePrint Archive*, no. 352, 2019.

[45] A. Miller, I. Bentov, S. Bakshi, R. Kumaresan, and P. McCorry, "Sprites and State Channels: Payment Networks that go Faster than Lightning," in *International Conference on Financial Cryptography and Data Security*, 2019, pp. 508–526.

[46] T. Network, "Trinity: Universal Off-chain Scaling Solution," https://trinity.tech/#/writepaper, 2020.

[47] AdEx Network, "Introducing OUTPACE: Off-Chain Unidirectional Trustless Payment Channels," https://www.adex.network/blog/introducing-outpace-off-chain-unidirectional-trustless-payment-channels/, 2018.

[48] R. Network, "Fast, Cheap, Scalable Token Transfers for Ethereum," https://raiden.network/, 2018.

[49] E. Tairi, P. Moreno-Sanchez, and M. Maffei, "A^2L: Anonymous Atomic Locks for Scalability in Payment Channel Hubs," in *IEEE Symposium on Security and Privacy*, 2021, pp. 1834–1851.

[50] M. Green and I. Miers, "Bolt: Anonymous Payment Channels for Decentralized Currencies," in *ACM Conference on Computer and Communications Security*, 2017, pp. 473–489.

[51] 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, pp. 1–36.

[52] S. Dziembowski, L. Eckey, S. Faust, and D. Malinowski, "PERUN: Virtual Payment Channels over Cryptographic Currencies," *IACR Cryptology ePrint Archive*, no. 635, 2017.

[53] G. Konstantopoulos, "DAppChains: Scaling Ethereum DApps through Sidechains," https://medium.com/loom-network, 2018.

[54] "Liquid," https://liquid.net/, 2020.

[55] J. Kanani, S. Nailwal, and A. Arjun, "Matic Whitepaper," https://github.com/maticnetwork/whitepaper, 2021.

[56] I. SKALE Labs, "The SKALE Network : An Ethereum Interoperable Elastic Blockchain Network," https://skale.network/whitepaper, 2020.

[57] "Liquidity Network," https://liquidity.network/, 2020.

[58] R. Khalil, "NOCUST-A Non-Custodial 2nd-Layer Blockchain Payment Hub," Ph.D. dissertation, Master's thesis, Swiss Federal Institute of Technology, Zurich, 2018.

[59] "Polygon's PoS," https://polygon.technology/solutions/polygon-pos, 2020.

[60] "Cartesi," https://cartesi.io/cartesi_lightpaper_english.pdf, 2020.

[61] "Fuel Network," https://fuel.sh/, 2020.

[62] "Offchain Labs' Arbitrum," https://arbitrum.io/, 2020.

[63] "OMG Network," https://omg.network/, 2020.

[64] "Optimism," https://www.optimism.io/, 2020.

[65] "Aztec 2.0," https://aztec.network/, 2020.

[66] D. Wang, J. Zhou, A. Wang, and M. Finestone, "Loopring: A Decentralized Token Exchange Protocol," 2018.

[67] "Starkware," https://starkware.co/, 2020.

[68] M. Labs, "zkSync: Bringing Trustless, Scalable Payments to Ethereum," https://zksync.io/, 2019.

[69] M. Dong, Q. Liang, X. Li, and J. Liu, "Celer Network: Bring Internet Scale to Every Blockchain," *arXiv preprint:1810.00037*, 2018.

[70] C. Network, "Connext : The Interoperability Protocol of L2 Ethereum," https://docs.connext.network/, 2016.

[71] M. Borkowski, M. Sigwart, P. Frauenthaler, T. Hukkinen, and S. Schulte, "DeXTT: Deterministic Cross-blockchain Token Transfers," *IEEE Access*, vol. 7, pp. 111 030–111 042, 2019.

[72] L. Network, "Intro to Loom Network," https://loomx.io/developers/en/intro-to-loom.html, 2016.

[73] P. K. Igor Barinov, Viktor Baranov, "POA Network," https://www.poa.network/v/master-1/for-users/whitepaper/poadao-v1, 2018.

[74] T. Baneth, "Waterloo - a Decentralized Practical Bridge between EOS and Ethereum," https://blog.kyber.network/waterloo-a-decentralized-practical-bridge-between-eos-and-ethereum-1c230ac65524, 2019.

[75] W. Warren and A. Bandeali, "0x: An Open Protocol for Decentralized Exchange on the Ethereum Blockchain," https://github.com/0xProject/whitepaper, 2017.

[76] T. M. Mayer, C. Mai, and N. Jesse, "Tokrex: Meta-system for Real-time Intra- and Cross-chain Swaps," https://tokrex.org/whitepapers/WhitePaper-Tokrex-RealTimeIntraCrossChainSwaps.pdf, 2017.

[77] M. Spoke and N. Team, "AION: Enabling the Decentralized Internet," https://whitepaper.io/document/31/aion-whitepaper, 2017.

[78] ARK, "ARK Ecosystem Whitepaper," https://ark.io/Whitepaper.pdf, 2019.

[79] B. Ethan and K. Jae, "Cosmos Whitepaper: A Network Of Distributed Ledgers," https://v1.cosmos.network/resources/whitepaper, 2016.

[80] Komodo, "Komodo Whitepaper," https://komodoplatform.com/wp-content/uploads/2018/06/Komodo-Whitepaper-June-3.pdf, 2018.

[81] G. Wood, "Polkadot: Vision For A Heterogeneous Multi-chain Framework," https://github.com/polkadot-io/polkadotpaper/raw/master/PolkaDotPaper.pdf, 2016.

[82] H. Kalodner, S. Goldfeder, X. Chen, S. M. Weinberg, and E. W. Felten, "Arbitrum: Scalable, Private Smart Contracts," in *USENIX Security Symposium)*, 2018, pp. 1353–1370.

[83] J. Teutsch and C. Reitwießner, "A Scalable Verification Solution for Blockchains," *arXiv preprint:1908.04756*, 2019.

[84] S. Matetic, K. Wüst, M. Schneider, K. Kostiainen, G. Karame, and S. Capkun, "BITE: Bitcoin Lightweight Client Privacy using Trusted Execution," in *USENIX Security Symposium)*, 2019, pp. 783–800.

[85] J. Lind, O. Naor, I. Eyal, F. Kelbert, E. G. Sirer, and P. Pietzuch, "Teechain: A Secure Payment Network with Asynchronous Blockchain Access," in *ACM Symposium on Operating Systems Principles*, 2019, pp. 63–79.

[86] J. Lind, I. Eyal, P. Pietzuch, and E. G. Sirer, "Teechan: Payment Channels using Trusted Execution Environments," *arXiv preprint:1612.07766*, 2016.

[87] I. Bentov, Y. Ji, F. Zhang, L. Breidenbach, P. Daian, and A. Juels, "Tesseract: Real-Time Cryptocurrency Exchange using Trusted Hardware," in *ACM Conference on Computer and Communications Security*, 2019, pp. 1521–1538.

[88] K. Wüst, S. Matetic, M. Schneider, I. Miers, K. Kostiainen, and S. Čapkun, "ZLiTE: Lightweight Clients for Shielded Zcash Transactions using Trusted Execution," in *International Conference on Financial Cryptography and Data Security*, 2019, pp. 179–198.

[89] V. Buterin and V. Griffith, "Casper the Friendly Finality Gadget," *arXiv preprint:1710.09437*, 2017.

[90] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A Provably Secure Proof-of-stake Blockchain Protocol," in *Annual International Cryptology Conference*, 2017, pp. 357–388.

[91] M. Lokhava, G. Losa, D. Mazières, G. Hoare, N. Barry, E. Gafni, J. Jove, R. Malinowsky, and J. McCaleb, "Fast and Secure Global Payments with Stellar," in *ACM Symposium on Operating Systems Principles*, 2019, pp. 80–96.

[92] Z. Fathi, A. J. Rafsanjani, and F. Habibi, "Anon-ISAC: Anonymity-Preserving Cyber Threat Information Sharing Platform Based on Permissioned Blockchain," in *Iranian Conference on Electrical Engineering*, 2020, pp. 1–5.

[93] H. Team, "Harmony: Technical Whitepaper," https://harmony.one/whitepaper.pdf, 2018.

[94] L. Josep, A. El-Fakdi, V. Torres, and X. Amengual, "Logo Recognition By Consensus For Enabling Blockchain Implementations," in *International Conference on Recent advances in Artificial Intelligence Research and Development*, vol. 300, 2017, p. 257.

[95] J. Wang and H. Wang, "Monoxide: Scale Out Blockchains With Asynchronous Consensus Zones," in *USENIX Symposium on Networked Systems Design and Implementation)*, 2019, pp. 95–112.

[96] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A Secure, Scale-out, Decentralized Ledger Via Sharding," in *IEEE Symposium on Security and Privacy*, 2018, pp. 583–598.

[97] A. Manuskin, M. Mirkin, and I. Eyal, "Ostraka: SecurE Blockchain Scaling By Node Sharding," in *IEEE European Symposium on Security and Privacy Workshops*, 2020, pp. 397–406.

[98] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: Scaling Blockchain Via Full Sharding," in *ACM Conference on Computer and Communications Security*, 2018, pp. 931–948.

[99] H. Chen and Y. Wang, "SSchain: A Full Sharding Protocol For Public Blockchain Without Data Migration Overhead," *Pervasive and Mobile Computing*, vol. 59, p. 101055, 2019.

[100] S. AG, "Stegos : A Platform for Privacy Applications," https://stegos.com//docs/stegos-whitepaper.pdf, 2019.

[101] Z. team, "ZILLIQA Technical Whitepaper," https://docs.zilliqa.com/whitepaper.pdf, 2017.

[102] A. Gągol and M. Świętek, "Aleph: A Leaderless, Asynchronous, Byzantine Fault Tolerant Consensus Protocol," 2018.

[103] S. Forestier, D. Vodenicarevic, and A. Laversanne-Finot, "Blockclique: Scaling Blockchains Through Transaction Sharding In A Multithreaded Block Graph," 2018.

[104] G. Danezis and D. Hrycyszyn, "Blockmania: From Block Dags To Consensus," 2018.

[105] A. Churyumov, "Byteball: A Decentralized System for Storage and Transfer of Value," https://byteball.org/Byteball.pdf, 2016.

[106] M. J. Amiri, D. Agrawal, and A. E. Abbadi, "CAPER: A

Cross-application Permissioned Blockchain," *Proceedings of the VLDB Endowment*, vol. 12, no. 11, pp. 1385–1398, 2019.

[107] H. Gupta and D. Janakiram, "CDAG: A Serialized Blockdag for Permissioned Blockchain," *arXiv preprint:1910.08547*, 2019.

[108] W. Martino, M. Quaintance, and S. Popejoy, "Chainweb: A Proof-of-work Parallel-chain Architecture For Massive Throughput," https://neironix.io/documents/whitepaper/6793/chainweb-v15.pdf, 2018.

[109] C. li, P. Li, D. Zhou, W. Xu, F. Long, and A. Yao, "Scaling Nakamoto Consensus To Thousands Of Transactions Per Second," 2018.

[110] T.-Y. Chen, W.-N. Huang, P.-C. Kuo, H. Chung, and T.-W. Chao, "DEXON: A Highly Scalable, Decentralized Dag-based Consensus Algorithm," 2018.

[111] T. Zhou, X. Li, and H. Zhao, "DLattice: A Permissionless Blockchain Based On DPoS-BA-DAG Consensus For Data Tokenization," *IEEE Access*, vol. 7, pp. 39 273–39 287, 2019.

[112] "Eunomia," https://eunomia.social/, 2020.

[113] Y. Sompolinsky and A. Zohar, "Secure High-rate Transaction Processing In Bitcoin," in *International Conference on Financial Cryptography and Data Security*, 2015, pp. 507–527.

[114] X. Boyen, C. Carr, and T. Haines, "Graphchain: A Blockchain-free Scalable Decentralised Ledger," in *2nd ACM Workshop on Blockchains, Cryptocurrencies, and Contracts*, 2018, pp. 21–33.

[115] S. Tang, Q. Zhang, Z. Gao, J. Zheng, and D. Gu, "Bracing A Transaction DAG with A Backbone Chain," *IACR Cryptology ePrint Archive*, p. 472, 2020.

[116] L. Baird, "The Swirlds Hashgraph Consensus Algorithm: Fair, Fast, Byzantine Fault Tolerance," *Swirlds Tech Reports SWIRLDS-TR-2016-01*, 2016.

[117] Y. Lewenberg, Y. Sompolinsky, and A. Zohar, "Inclusive Block Chain Protocols," in *International Conference on Financial Cryptography and Data Security*, 2015, pp. 528–547.

[118] F. Xiang, W. Huaimin, S. Peichang, O. Xue, and Z. Xunhui, "Jointgraph: A DAG-based Efficient Consensus Algorithm for Consortium Blockchains," *Software: Practice and Experience*, vol. 51, no. 10, pp. 1987–1999, 2021.

[119] Q. Nguyen, A. Cronje, M. Kong, E. Lysenko, and A. Guzev, "Lachesis: Scalable Asynchronous BFT on DAG Streams," 2021.

[120] I. Bentov, P. Hubáček, T. Moran, and A. Nadler, "Tortoise and Hares Consensus: The Meshcash Framework for Incentive-compatible, Scalable Cryptocurrencies," in *International Symposium on Cyber Security Cryptography and Machine Learning*, 2021, pp. 114–127.

[121] C. LeMahieu, "Nano: A Feeless Distributed Cryptocurrency Network," https://nano.org/en/whitepaper, p. 17, 2018.

[122] "Obyte," https://obyte.org/, 2020.

[123] H. Yu, I. Nikolić, R. Hou, and P. Saxena, "Ohie: Blockchain Scaling Made Simple," in *IEEE Symposium on Security and Privacy*, 2020, pp. 90–105.

[124] A. K. Jaiswal, "Parsec: A State Channel for the Internet of Value," *arXiv preprint:1807.11378*, 2018.

[125] Y. Sompolinsky and A. Zohar, "Phantom," *IACR Cryptology ePrint Archive*, no. 104, 2018.

[126] V. Bagaria, S. Kannan, D. Tse, G. Fanti, and P. Viswanath, "Prism: Deconstructing The Blockchain To Approach Physical Limits," in *ACM Conference on Computer and Communications Security*, 2019, pp. 585–602.

[127] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "SPECTRE: A Fast and Scalable Cryptocurrency Protocol," *IACR Cryptology ePrint Archive*, no. 1159, 2016.

[128] Z. Yin, A. Ruan, M. Wei, H. Li, K. Yuan, J. Wang, Y. Wang, M. Ni, and A. Martin, "StreamNet: A DAG System with Streaming Graph Computing," in *Future Technologies Conference*, 2020, pp. 499–522.

[129] C. Liu, D. Wang, and M. Wu, "Vite: A High Performance Asynchronous Decentralized Application Platform," https://cryptorating.eu/whitepapers/VITE/vite_en.pdf, 2018.

[130] "OP_CHECKLOCKTIMEVERIFY," https://github.com/bitcoin/bips/blob/master/bip-0065.mediawiki, 2014.

[131] J. Spilman, "Anti DoS for tx replacement," https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2013-April/002433.html, 2013.

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

[133] 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," in *International Conference on Financial Cryptography and Data Security*, 2019, pp. 35–49.

[134] J. Coleman, L. Horne, and L. Xuanji, "Counterfactual: Generalized State Channels," *https://l4.ventures/papers/statechannels.pdf*, 2018.

[135] T. Close and A. Stewart, "Forcemove: An n-party State Channel Protocol," *Magmo, White Paper*, 2018.

[136] T. Dryja, "Unlinkable Outsourced Channel Monitoring," https://diyhpl.us/wiki/transcripts/scalingbitcoin/milan/unlinkable-outsourced-channel-monitoring/, 2016.

[137] M. Khabbazian, T. Nadahalli, and R. Wattenhofer, "Outpost: A Responsive Lightweight Watchtower," in *ACM Conference on Advances in Financial Technologies*, 2019, pp. 31–40.

[138] G. Avarikioti, F. Laufenberg, J. Sliwinski, Y. Wang, and R. Wattenhofer, "Towards Secure and Efficient Payment Channels," *arXiv preprint:1811.12740*, 2018.

[139] A. Dmitrienko, D. Noack, and M. Yung, "Secure Wallet-assisted Offline Bitcoin Payments with Double-spender Revocation," in *ACM on Asia Conference on Computer and Communications Security*, 2017, pp. 520–531.

[140] T. Takahashi and A. Otsuka, "Short Paper: Secure Offline Payments in Bitcoin," in *International Conference*

on Financial Cryptography and Data Security, 2019, pp. 12–20.

[141] K. Hu and Z. Zhang, "Fast Lottery-based Micropayments for Decentralized Currencies," in *Australasian Conference on Information Security and Privacy*, 2018, pp. 669–686.

[142] R. Pass and A. Shelat, "Micropayments for Decentralized Currencies," in *ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 207–218.

[143] C. Burchert, C. Decker, and R. Wattenhofer, "Scalable Funding of Bitcoin Micropayment Channel Networks," *Royal Society Open Science*, vol. 5, no. 8, p. 180089, 2018.

[144] A. R. Pedrosa, M. Potop-Butucaru, and S. Tucci-Piergiovanni, "Scalable Lightning Factories for Bitcoin," in *ACM/SIGAPP Symposium on Applied Computing*, 2019, pp. 302–309.

[145] G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, and S. Ravi, "Concurrency and Privacy with Payment Channel Networks," in *ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 455–471.

[146] C. Egger, P. Moreno-Sanchez, and M. Maffei, "Atomic Multi-channel Updates with Constant Collateral in Bitcoin-compatible Payment Channel Networks," in *ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 801–815.

[147] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei, "Anonymous Multi-Hop Locks for Blockchain Scalability and Interoperability," in *Network and Distributed System Security Symposium*, 2019, pp. 1–15.

[148] G. Avarikioti, G. Janssen, Y. Wang, and R. Wattenhofer, "Payment Network Design with Fees," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, 2018, pp. 76–84.

[149] R. Khalil and A. Gervais, "REVIVE: Rebalancing Off-blockchain Payment Networks," in *ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 439–453.

[150] S. Dziembowski, S. Faust, and K. Hostáková, "General State Channel Networks," in *ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 949–966.

[151] S. Dziembowski, L. Eckey, S. Faust, J. Hesse, and K. Hostáková, "Multi-party Virtual State Channels," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2019, pp. 625–656.

[152] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille, "Enabling Blockchain Innovations with Pegged Sidechains," http://kevinriggen.com/files/sidechains.pdf, 2014.

[153] A. Singh, K. Click, R. M. Parizi, Q. Zhang, A. Dehghantanha, and K.-K. R. Choo, "Sidechain Technologies in Blockchain Networks: An Examination and State-of-the-art Review," *Elsevier Network and Computer Applications*, vol. 149, 2020.

[154] R. Khalil, A. Zamyatin, G. Felley, P. Moreno-Sanchez, and A. Gervais, "Commit-Chains: Secure, Scalable Off-Chain Payments," *IACR Cryptology ePrint Archive*, no. 642, 2018.

[155] J. Poon and V. Buterin, "Plasma: Scalable Autonomous Smart Contracts," *White paper*, pp. 1–47, 2017.

[156] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge," *IACR Cryptology ePrint Archive*, no. 953, 2019.

[157] J. Song, "Data Availability Problem in Implementing Plasma Design," https://medium.com/onther-tech/data-availability-problem-in-implementing-plasma-design-6e23df1a147f, 2018.

[158] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia, "A Survey on Blockchain Interoperability: Past, Present, and Future Trends," *ACM Computing Surveys*, vol. 54, no. 8, pp. 1–41, 2021.

[159] A. Zamyatin, D. Harz, J. Lind, P. Panayiotou, A. Gervais, and W. J. Knottenbelt, "XCLAIM: Trustless, Interoperable, Cryptocurrency-backed Assets," in *IEEE Symposium on Security and Privacy*, 2019, pp. 193–210.

[160] H. Tian, K. Xue, X. Luo, S. Li, J. Xu, J. Liu, J. Zhao, and D. S. Wei, "Enabling Cross-chain Transactions: A Decentralized Cryptocurrency Exchange Protocol," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 3928–3941, 2021.

[161] Intel, "Intel Software Guard eXtensions (Intel SGX)," https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/overview.html, 2021.

[162] V. Costan and S. Devadas, "Intel SGX Explained," *IACR Cryptology ePrint Archive*, no. 086, 2016.

[163] A. Nilsson, P. N. Bideh, and J. Brorsson, "A Survey of Published Attacks on Intel SGX," *arXiv preprint:2006.13598*, 2020.

[164] S. Fei, Z. Yan, W. Ding, and H. Xie, "Security Vulnerabilities of SGX and Countermeasures: A Survey," *ACM Computing Surveys*, vol. 54, no. 6, pp. 1–36, 2021.

[165] V. Sivaraman, S. B. Venkatakrishnan, M. Alizadeh, G. Fanti, and P. Viswanath, "Routing Cryptocurrency with the Spider Network," in *ACM Workshop on Hot Topics in Networks*, 2018, pp. 29–35.

[166] G. Di Stasi, S. Avallone, R. Canonico, and G. Ventre, "Routing Payments on the Lightning Network," in *International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2018, pp. 1161–1170.

[167] G. Malavolta, P. Moreno-Sanchez, A. Kate, and M. Maffei, "SilentWhispers: Enforcing Security and Privacy in Decentralized Credit Networks," in *Network and Distributed Security Symposium*, 2017, pp. 1–18.

[168] S. Roos, P. Moreno-Sanchez, A. Kate, and I. Goldberg, "Settling Payments Fast and Private: Efficient Decen-

tralized Routing for Path-based Transactions," *arXiv preprint arXiv:1709.05748*, 2017.

[169] J. Harris and A. Zohar, "Flood & Loot: A Systemic Attack on the Lightning Network," in *ACM Conference on Advances in Financial Technologies*, 2020, pp. 202–213.

[170] A. Riard and G. Naumenko, "Time-Dilation Attacks on the Lightning Network," *arXiv preprint:2006.01418*, 2020.

[171] "Peer Services," https://github.com/bitcoin/bips/blob/master/bip-0157.mediawiki, 2017.

[172] C. Pérez-Sola, A. Ranchal-Pedrosa, J. Herrera-Joancomartí, G. Navarro-Arribas, and J. Garcia-Alfaro, "Lockdown: Balance Availability Attack Against Lightning Network Channels," in *International Conference on Financial Cryptography and Data Security*, 2020, pp. 245–263.

[173] J. Herrera-Joancomartí, G. Navarro-Arribas, A. Ranchal-Pedrosa, C. Pérez-Solà, and J. Garcia-Alfaro, "On the Difficulty of Hiding the Balance of Lightning Network Channels," in *ACM Asia Conference on Computer and Communications Security*, 2019, pp. 602–612.

[174] G. v. Dam, R. A. Kadir, P. N. Nohuddin, and H. B. Zaman, "Improvements of the Balance Discovery Attack on Lightning Network Payment Channels," in *IFIP International Conference on ICT Systems Security and Privacy Protection*, 2020, pp. 313–323.

[175] A. Mizrahi and A. Zohar, "Congestion Attacks in Payment Channel Networks," in *International Conference on Financial Cryptography and Data Security*, 2021, pp. 170–188.

[176] Z. Avarikioti, O. S. Thyfronitis Litos, and R. Wattenhofer, "Cerberus Channels: Incentivizing Watchtowers for Bitcoin," in *International Conference on Financial Cryptography and Data Security*, 2020, pp. 346–366.

[177] D. A. Harding and P. Todd, "Opt-in Full Replace-by-Fee Signaling in Bitcoin Improvement Proposal 125," https://github.com/bitcoin/bips/blob/master/bip-0125.mediawiki, 2015.

[178] "Anchor Outputs, Lightning Network Specifications," https://github.com/lightningnetwork/lightning-rfc/pull/688, 2019.

[179] S. Mazumdar, P. Banerjee, and S. Ruj, "Griefing-penalty: Countermeasure for Griefing Attack in Lightning Network," *arXiv preprint:2005.09327*, 2020.

[180] S. Tochner, S. Schmid, and A. Zohar, "Hijacking Routes in Payment Channel Networks: A Predictability Tradeoff," *arXiv preprint:1909.06890*, 2019.

[181] E. Rohrer, J. Malliaris, and F. Tschorsch, "Discharged Payment Channels: Quantifying the Lightning Network's Resilience to Topology-based Attacks," in *IEEE European Symposium on Security and Privacy Workshops*, 2019, pp. 347–356.

[182] A. Biryukov and D. Feher, "Deanonymization of Hidden Transactions in Zcash," *University of Luxembourg*, pp. 1–15, 2018.

[183] F. Béres, I. A. Seres, A. A. Benczúr, and M. Quintyne-Collins, "Blockchain is Watching You: Profiling and Deanonymizing Ethereum Users," in *International Conference on Decentralized Applications and Infrastructures*, 2021, pp. 69–78.

[184] "Evaluating Ethereum L2 Scaling Solutions: A Comparison Framework," https://blog.matterlabs.io/evaluating-ethereum-l2-scaling-solutions-a-comparison-framework-b6b2f410f955, 2020.