Designing Layer-2 based solutions to tackle the blockchain trilemma

PhD Thesis Proposal

PhD Candidate: Andrea Lisi
Supervisors: Prof. Laura Ricci, Dr. Paolo Mori

October 20, 2020

Contents

1 Introduction

The last ten years have been characterized by the successful adoption of the blockchain technology, beginning with Bitcoin, the first digital currency secured by cryptography. Bitcoin [38] creates a trusted network composed by trustless actors that cooperate together to exchange the cryptocurrency and to maintain the network. After Bitcoin other projects became popular, especially Ethereum [55] that has the goal of building a decentralized worldwide computer running Decentralized Applications (DApps), i.e. applications with no single authority owning them, implemented by software entities called smart contracts. However, such networks sacrifice scalability, performance and user privacy to provide security and decentralization [14]. Real world applications need to scale, and for this reason permissioned networks have been introduced [56]. However, to provide scalability, permissioned networks sacrifice decentralization. Since Bitcoin and Ethereum are currently well participated, the open source community is working on improving their scalability with the goal to encourage their adoption in real world scenarios. One direction is to change the underlying architecture and the protocol itself: they are so-called **Layer-1** solutions. A list of Layer-1 proposals can be found in [57]. The modifications in this direction can either be small or big, but they typically preserve the overall functionality and behaviour of the blockchain. Otherwise, it is required to build an entirely new architecture and protocol (as it happens for Ethereum 2.0[1]). An alternative is to build an overlay that executes the operations off-chain, but a proof of execution must to be stored, or verified, on chain. These are so-called **Layer-2** solutions [18]. A well designed layer-2 overlay can improve the scalability, or privacy, of a blockchain-based application by moving out the blockchain the operations and the data that have specific requirements, while preserving the underlying architecture and, as a consequence, its properties. Auditability and immutability are preserved linking the off-chain operations to the blockchain with transactions. The successful implementation of a layer-2 overlay may require a few changes to the layer-1: for example, the adoption of the Lightning Network [42] on top of Bitcoin required only a modification to the block structure that was integrated with a soft-fork called Segregated Witness [50] in August 2017.

The proposal of the thesis consists in analyzing the existing layer-2 solutions, evaluate their weaknesses and propose a alternatives to solve them, and finally propose a methodological framework to integrate a layer-2 model

[1]`https://ethereum.org/en/eth2/`

to blockchain-based applications. The document is structured as follows: Section 2 introduces the blockchain technology, the state of the art of layer-2 high level models and the technologies implementing them. Finally, Section 3 lists the current open problems characterizing blockchain-based applications, and presents the possible research directions that can be investigated during the following years of the PhD.

2 State of the art

This section provides the current state of the art in blockchain technology, describing how the technology works using as examples the most known implementations, Bitcoin and Ethereum. Afterwards, we focus the attention to the Layer-2 technology i.e. an abstract layer that operates on top of the blockchain technology and connected to it.

This section is structured as follows: in Section 2.1 we describe the generic functionality of the blockchain technology; in Section 2.1.1 we describe the two main implementations, Bitcoin and Ethereum; Section 2.2 lists the state of the art of existing Layer-2 high level models, Section 2.2.1, and their implementations, Section 2.2.2, with a particular care on the Lightning Network protocol in Section 2.2.3.

2.1 Blockchain technology

The blockchain is a data structure modeled as a linked list of **blocks**, each block storing a sequence of ordered **transactions**, and is maintained by the nodes of a P2P network. A transaction sent by a peer represents an update of the state modeled by the blockchain. In Bitcoin, that implements a digital currency, the state models the balance of all the users, so the blockchain is also known as "ledger". The blockchain can be updated only appending a new block, and it is an action performed in rounds roughly having the same size in seconds or minutes. At each round, one selected peer picks some transactions it received, cached in a transaction pool, it executes them and it stores each transaction in a new block. The peer appends this new block to its local copy of the blockchain and, finally, it broadcasts the block to the rest of the network so that every peer updates their blockchains with the same block. Each peer receiving a new block verifies that the transactions have been executed correctly, and if they do, each peer updates its blockchain otherwise they refuse the block. We can see the blockchain as a P2P replicated state machine.

	Permissionless	**Permissioned**
Public	No restrictions on readers and writers. Examples: Bitcoin, Ethereum, ZCash.	Restrictions on the writers. Examples: EOS.
Private	Not realistic	Fully restricted network. Examples: Hyperledger Fabric, Ripple, Quorum.

Table 1: Configurations of DLT

The most complicated part of the protocol is how to choose the peer proposing the new block. Since there is not a manager selecting a peer, the peers need to have a rule establishing the conditions a peer must satisfy to be the next block proposer. This is known as the **consensus algorithm**. To incentivize the participation, the protocol is designed to reward the peers proposing a new block: the reward is an hardcoded transaction that is automatically put in the block with the proposer as receiver. As a consequence, the peers compete to be block proposers and to receive the reward, e.g. an amount of cryptocurrency, and the consensus algorithm needs to be designed so it works in a trustless network, i.e. a network where each peer acts for its own good, beneficially or maliciously. The most popular trustless consensus algorithm is called Proof-Of-Work (PoW). The verification process performed by each peer when receiving a block is also very important, because it prevents a proposer to broadcast garbage blocks, or transactions that cheat the system (e.g. double spending transactions): if the block is not accepted by the majority of the peers, the proposer does not get compensated because the block with the reward does not belong to the ”agreed” blockchain, i.e. a blockchain that is identical for at least 51% of the peers.

The last ten years, after the presentation of the Bitcoin paper, were characterized by many projects involving the blockchain technology investigating alternative approaches to the linked list of blocks. As a more general term, Distributed Ledger Technology (DLT) is used. For simplicity, in the following of the paper we use the terms blockchain, chain, ledger and DLT interchangeably. Also, DLTs are not only interesting in academic research, but industries are approaching to the technology as well. However, industries might have different requirements, and typically they do not wish to participate to a P2P network that has no restrictions on the participants like Bitcoin. We present different architectural configuration following the definition proposed in [56]: a *writer* is a peer in charge to propose new blocks; a *reader* is a peer allowed to participate to the network, can read the blocks, send transactions but cannot propose blocks. A network with no restriction on the writers is known as **permissionless**, otherwise as **permissioned**. A network with no restrictions on the readers is known as **public**, otherwise as **private**. We can have 4 different configurations that are shown in Table 1, however a private permissionless DLT is not realistic (readers are restricted but not writers).

Permissionless architectures create a trustless network, because the peers require no prior identification, and typically can verify a smaller amount of transactions per block due to a complex consensus algorithm. On the contrary, permissioned architectures are closer to a client-server or a federated one because the number of writers is typically small. Therefore, the security on the data stored in the blocks is higher in permissionless networks, that are typically classified as **immutable** under some security assumptions, for example at least 51% of the nodes behaving honestly: immutability means some data cannot be tampered once it has been stored in the blockchain, so the blockchain can only grow. Public architectures are typically known as **transparent** (or auditable), because a peer joining the network is allowed to read all the blocks and verify their correctness. As a consequence, privacy-sensitive data cannot be stored on such networks: private networks are more suitable for this requirement.

2.1.1 Bitcoin and Ethereum

Bitcoin Bitcoin [38] adopts a blockchain to implement a P2P digital currency secured by cryptography. Its paper came out in 2009. The currency has the name of bitcoin[2] (BTC). The motivations behind Bitcoin are "anarchic" and its concept is shared by the cypherpunk community, because the protocol allows the exchange of currency between distant users without involving trusted parties like banks. As in the majority of blockchains, in Bitcoin a block consists of an header and a body. The body contains the list of transactions, the header contains the root of the Merkle tree built from the transactions, used to verify they have not been tampered, and the hash of the previous block, named hash pointer, chaining the blocks together (Figure 1). If a transaction is tampered then it modifies the Merkle root, and the hash of the block becomes inconsistent with the hash pointer of the next block.

[2]Capital B identifies the protocol, the currency otherwise

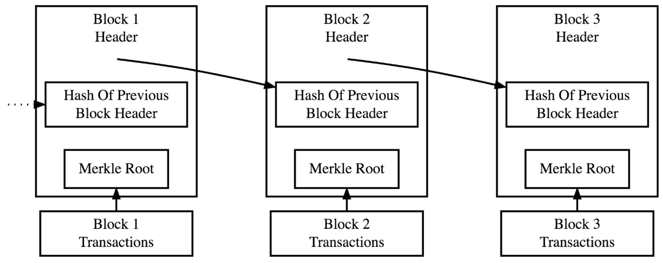


Figure 1: A visualization of the headers of the blocks

A user sends to another user some bitcoins with a transaction, and that transaction will be processed by a block proposer and recorded in the blockchain. The Bitcoin blockchain is the ledger of all the transactions sent by all the users. A transaction can be sent by a wallet, a program that generates pairs of private and public keys that are used to sign transactions and to verify the receiver respectively, and stores the balance of the user. A wallet is able to verify its transactions without the need to store the whole blockchain via the Simple Payment Verification (SPV) protocol[3], a technique that involves Bloom Filters and is correct with high probability. The keys are created by the wallet with the Elliptic Curve Digital Signature Algorithm (ECDSA): the wallet generates a random 256-bits private key *sk*, computes the public key *pk* and finally generates an address *addr* from *pk*. The address is the destination of a payment and typically identifies a user who, if does not reveal their identity, is unknown. But since to an address corresponds a single user, users in Bitcoin are not anonym but pseudo-anonym.

In Bitcoin, a transaction is composed by: 1) the list of its inputs, the BTC received by an address; 2) the list of its outputs, the BTC to send to other addresses; 3) for each input and output, a script that defines a locking condition for each output, and an unlocking condition for each input. The locking conditions means ”I lock X BTC to this public address” that is used to send bitcoins, the unlocking conditions means ”Whoever provides a signature valid for this address receives X BTC” and is used to receive the same bitcoins. These conditions are specified by smart contracts that are intentionally not Turing-complete to avoid misuse. When creating a transaction, the user must spend all the bitcoins (of that address), i.e. the sum of all the inputs. Therefore, the user must send themselves the change, known

[3]SPV: `https://en.bitcoinwiki.org/wiki/Simplified_Payment_Verification`

as Unspent Transaction Output (UTXO). After a transaction is signed it is broadcast to the network so that the nodes can process it and put that in a block. Bitcoin currently uses the PoW consensus algorithm. In PoW [17], the block proposers are known as miners and compete together to publish the new block and get the reward. In PoW the miner that can publish a new block is chosen after it finds a solution of a cryptographic puzzle that is hard to solve but easy to verify the solution (like a sudoku). This puzzle consists in finding the pre-image of an hash value, and that can be solved only with brute force: in PoW the miner with more computational power has more possibility to publish a new block, and therefore it has more decision power.

Ethereum Ethereum [55] was released in 2015 and like Bitcoin is powered by a cryptocurrency that is called ether (ETH) and utilizes the PoW consensus algorithm. While the goal of Bitcoin is to implement a P2P digital currency, the goal of Ethereum is to create a distributed computing platform, i.e. a computer that is composed by all the nodes of the Ethereum network. The programs are called smart contracts, and they are Turing complete. A smart contract keeps a state and provides a set of functions to read and update that state, it can be implemented in a programming language, Solidity is an example, and it is compiled in bytecode to execute on the Ethereum Virtual Machine (EVM) running on all the nodes of the Ethereum network. The bytecode and the contract's state are recorded in the Ethereum blockchain, therefore replicated by all the nodes, and each update happens through a transaction. Smart contracts can hold ETH as well as users, so it is possible to write programs that natively handle cryptocurrency. Since the programs are Turing complete, the Halting Problem is an issue that could freeze the whole network. As a countermeasure, the Ethereum protocol defines the concept of *gas*. Every transaction is composed by EVM instructions, and to every instruction is assigned a cost in units of gas [55], therefore the summation of the cost of each instruction determines the cost of the transaction in terms of gas. To every transaction the user can assign a *gas limit*, i.e. the maximum amount of units of gas that the computation can take to terminate: if the computation exceeds that amount, it halts and all the intermediate state updates are reverted. A block has its own gas limit, therefore the gas limit of a transaction cannot exceed the gas limit of a block. Moreover, the gas represents the fee the user sending the transaction has to pay. To determine its cost in ETH, the user assigns a *gas price* in ETH to each unit of gas, and the summation determines the fee to be paid: higher the fee, higher the chance to be mined sooner. The user applications based on smart contracts are known as Decentralized Applications (DAPP), and they have an economic cost to the users when using smart contracts, so DAPPs need to carefully think which data is worth to pay for and which not. The most popular DAPP is called Cryptokitties[4], a digital marketplace of unique cats that can be exchanged between users.

2.2 Layer-2 technologies

This section presents high level models and solutions labeled as **Layer-2** [18]. Layer-2 solutions have the goal of improving scalability and privacy of public permissionless blockchains creating a coupled higher layer. Such solutions are welcome within a blockchain community, because applying changes to the low level protocol, the Layer-1, is not an easy decision, and it is always a point of debate within the community. Applying a breaking change to the protocol has success only if the big majority of the nodes update their software, leading to a fork. This fork can be *soft*, typically a consequence of minor changes that are retro-compatible so that new and old software are able to cooperate; otherwise a fork can be *hard*, typically a consequence of major changes that are not retro-compatible with the old version of the software: the nodes with old software would not be able to join the network unless they update their software. Given the difficulty of such decisions, new techniques have been proposed known as Layer-2 because they implement a software layer that works on top of a blockchain but it is connected to it. Applying a layer-2 protocol on top of an existing blockchain may require a modification to the layer underneath: as an example, building the Lightning Network on top of Bitcoin required a modification on the transaction data structure that was introduced with the Segregated Witness soft-fork [50]. Below we provide an informal definition of layer-1 and layer-2 solutions:

Definition 2.1 (Layer-1 solutions)**.** Layer-1 solutions reflect modifications to the architecture supporting the blockchain protocol so that it can increase its security, performances and encourage its adoption. Modifications at this layer require a fork, soft or hard, that most of the peers need to shift so that it becomes official. Such solutions focus on the consensus algorithm, on the parameters like the block size or the average difficulty of finding a new block. Layer-1 solutions are also known as "On-chain".

[4]`http://www.cryptokitties.co/`

Definition 2.2 (Layer-2 solutions)**.** Layer-2 solutions improve the blockchain ecosystem by building an application layer that works on top of the blockchain-based protocol, allowing the users to securely exchange transactions and modifying a shared state in a fast and safe way. These solutions may require a fork of the blockchain. Layer-2 solutions are also known as "Off-chain".

Given the variety of existing blockchains protocols, each with different features, layer-2 technologies can connect the state of different blockchains. We differentiate the layer-2 solutions between single-chain and multi-chain, that have in common the goal of improving a blockchain-based application, but solve different types of use cases. We define as **Single-Chain Layer 2 solutions** those designed to work with a single blockchain protocol and relaying on specific transactions, and as **Multi-Chain Layer 2 solutions** those designed to solve a problem connecting independent blockchains and following a more generic approach. Multi-chain protocols are also known as interoperability or interledger techniques [6, 52, 23, 49, 24] and have the goal of building more complex blockchain-based applications that do not rely on a single chain, but rather on a set of them to meet different requirements.

Independently from how many chains are connected, layer-2 solutions are mainly divided into three categories: i) state channels, ii) sidechains and iii) off-chain computation. Section 2.2.1 describes the high level models characterizing layer-2 solutions, while Section 2.2.2 provides a detailed list of technologies existing in literature.

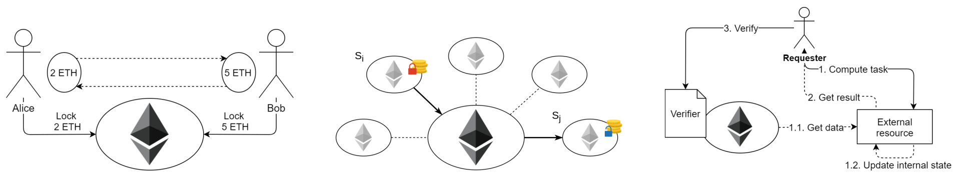
2.2.1 Layer-2 models

In this section we present the high level layer-2 models and architectures that are common to many existing, or proposed, technologies. We present the state channel model, a link between two peers secured with cryptography typically involving hash-locking; the sidechain model, which subdivides the blockchain system into multiple ones typically involving a single (or a few) main chain(s) and several sidechains with the goal to parallelize independent computations; the off-chain computation model, which outsources heavy computations, or some application logic, from the blockchain to external units.

The state channel and hash-locking models

A **state channel**[5] is a virtual connection between two peers that allows

[5]State channels: `https://www.jeffcoleman.ca/state-channels/`



(a) Payment channel (b) Sidechains (c) Off-chain computation

Figure 2: Visualization of layer-2 models

them to update a shared state. Typically, opening and closing a state channel requires a transaction to the blockchain from both the peers. The transaction locks an asset on the blockchain and allows the peers to safely apply updates on that asset. When the asset of a state channel is cryptocurrency, then it is typically known as **payment channel** and the updates are transfers from a user to another. A locked asset cannot be used in any transaction within the blockchain until the channel will be closed. An update on the state does not require any transaction: this has the positive consequences of improving both performance in terms of transactions per second, and privacy of blockchain related operations because they are not permanently recorded. Figure 2(a) shows an example of payment channel between Alice and Bob where they locked 2 and 5 ETH respectively. The solid arrows represent a transaction in the blockchain, the dashed arrows the channel and the circles the balance of each peer in the channel. Every state update in the channel is signed by who triggers the operation, and anti-cheating systems are typically supported by hash and time lock mechanisms (see Section 2.2.3).

Definition 2.3 (Timelock)**.** A timelock is a constraint that prevents the execution of an action until a specific time period (measured in seconds or blocks) passes.

Definition 2.4 (Hashlock)**.** A hashlock is a constraint that prevents the execution of an action unless the preimage of a hash value has been provided. This preimage is typically known as *secret*.

The timelock acts as the opposite of a deadline, i.e. denying an operation before a certain date, meanwhile the hashlock locks an operation with a value $h = H(s)$, where H is an hash function and s is a random number generated by a peer, and its preimage s unlocks it.

The concept of state channel can be generalized with multiple peers therefore creating a network of channels. This is realistic because a peer may not have enough resources to open a channel with any other peer of the blockchain, but a peer can exploit already opened channels to forward a state update or a payment: the latter is the case of the Lightning network. To achieve that, Hash-Time locked contracts (HTLC) are typically used to commit and perform payments in a fair way between multiple peers in a state channel network.

Definition 2.5 (Hash Time Locked Contracts)**.** An Hash time Locked Contract (HTLC) is a class of payments that uses hashlocks and timelocks allowing a receiver to claim a payment by generating a cryptographic proof, or to forfeit with the consequence of returning the payment back to the payer [19, 42].

Since hash locking mechanisms do not make any assumption on the underlying blockchain, it is possible to abstract the concept of locking assets living in different blockchains, like cryptocurrencies, tokens and other generic asset [22, 37, 20].

The sidechain model

A **sidechain** architecture [40, 41, 6] is composed by a blockchain, typically known as main chain[6], connected to multiple parallel smaller chains each consisting of an independent set of peers sharing a consistent state, executing and validating transactions in parallel with the peers of other sidechains: this has the goal to improve the transactions per second blockchain network can perform.

Depending on the design of the solution, each sidechain can be responsible of its own security [8], improving the sovereignty of each set of peers but risking a vulnerable bootstrap when the peers tend to be a few: the main chain acts as a hub connecting different sidechains. Otherwise, the system can rely its security to the main chain [40], for example storing into the main chain the header of the blocks produced by each sidechain, or giving to the main chain the role of sidechain validator with SPVs to allow fast verification with high probability and low data requirements. This models has the goal of exchanging assets between each sidechain: the typical approach consists of locking the asset in the main chain, i.e. label it as ”not usable” and unlocking it in the target sidechain so that it can be used. Figure 2(b) shows a visual representation of a main chain with 5 sidechains transferring some currency from S_i to S_j.

[6]For simplicity we provide an example with only one main chain

The sidechain approach can be used to connect also two existing blockchains and allow them to exchange or validate data. The connection can be either one-way, i.e. only one chain has the capability to read another chain, or two-ways [24, 6]. Depending on the technological capability, reading an external chain can be done with or without the help of an external entity like an oracle or a notary (see next paragraph). Authors in [18] do not consider this type of sidechains as layer-2 because they allow transactions between different consensus protocols, and according to their definition of layer-2, the security of a layer-2 protocol rely on the consensus of the underneath blockchain. The Ethereum 2.0 network has the goal of adopting an equivalent solution that is called sharding [14], but integrating it at the layer-1 level, i.e. Ethereum 2.0 itself is structured as a sidechain.

The off-chain computation model

The **off-chain computation** model [10, 12, 3, 1] aims to move out the blockchain the computation of an intensive task to reduce time, the fees, and the storage of bulky data. In [10] the authors distinguish between off-chain storage architectures, externalizing data storage but processing data on a blockchain node, off-chain computation architectures, externalizing computational tasks that read the (public) input from the blockchain, and hybrid architectures that are a combination of the twos: as an example, the authors cite the Lightning Network, Plasma and Raiden (see Section 2.2.2), i.e. state channels and side-chains models, as examples of hybrid off-chain computation architectures. For simplicity, we describe the off-chain computation model in its generality, thus externalizing either the data, the computation, or both. Figure 2(c) shows a visual representation of the model with a verifier smart contract.

In this model, the blockchain has the responsibility to record specific events to guarantee the correctness of the process. Protocols under this model rely most of their computation off-chain and store/process on the blockchain enough information to run a verification algorithm that will be used in case of a dispute: this approach is called verifiable computation [10]. A general architecture of this approach includes a Prover that performs an off-chain computation and stores on-chain a proof of that computation, and a Verifier that verifies the proof with an on-chain computation and confirms/denies the proof: the proof size and the on-chain verification algorithm needs to be cheap and must ensure the blockchain properties to justify this approach. When privacy and confidentiality are required, zero-knowledge proofs can be used for the verification [11]. As an example, a game of chess based on the blockchain [12] requires to store only the data relevant to the position of each piece, and moving a piece is not an heavy task to perform. However, checking endgame conditions, check mate, may be too expensive because it needs to verify that no move of the king can save him and no piece can sacrifice itself to save the king: this procedure can be moved off-chain. A player can claim a check mate on the blockchain chess, and the adversary can only confirm the check mate, or make a move that saves the king: both of them are light computations that can be done on-chain. Other approaches described in [10] consist of multi-party computation, enclave systems and incentive systems.

Since off-chain computation moves out relevant data from a blockchain, it can be used as well to implement a middleware that connects two or more blockchains, known as notary schemes [6, 24]. This approach is more intuitive with respect to state channels or sidechains. An entity known as notary acts as a gateway between two chains, A and B, and is able to send transactions on both chains: this means the notary has control of a private key in each chain. The notary can either be passive, i.e. whose actions on chain A are triggered by events risen on chain B, or active, i.e. the notary pushes data on both chains without any trigger. The notary introduces a trusted third party, and a single point of failure. However, the resilience of this approach can be improved by having a set of entities acting as a notary, and whose actions require multisignature operations by the majority of those entities.

2.2.2 Related work: Layer-2 existing technologies

In this section we provide a survey of the works that exist in literature that implement the layer-2 models described in the previous section, both single-chain and multi-chain.

State channel and hash-locking based solutions

The most known layer-2 solution based on payment channels is the Bitcoin Lightning Network [42] that has the goal of performing the transactions between the Bitcoin users with payment channels, and rely on the Bitcoin blockchain only to open and close a channel. Section 2.2.3 provides a detailed description.

The Raiden Network [43] is the equivalent approach for what concerns the Ethereum network but specialized in the transfer of tokens, in particular the standard ERC20 that is the most widespread among the Ethereum community. Channels can be created (and closed) with on-chain transaction between two users to exchange tokens via commitments called *balance proofs*, and can be used to forward a payment from a user A to a user B who do not have a direct open channel.

Pisa [36] is a solution that allows the creation of state channels robust to malicious or offline parties: the authors present a proof of concept on Ethereum. The initialization of a state channel involves the creation of the correspondent smart contract establishing the participants in the channel. All the parties can update the state of the channel with off-chain operations, but if a participant does not cooperate a dispute can be initialized in the smart contract. The goal of the dispute is to store on-chain the next authorized state depending on the actions submitted by the participating parties. For example, if the state S_{i-1} is the latest state of the channel and the parties issue a series of actions, the smart contract needs to authorize the state S_i by choosing one action among the submitted ones. A participant can "hire" a third-party called Custodian to watch the channel in their behalf. The custodian receives the list of signatures leading to the current state of the channel, and the job of the Custodian is to win the dispute on the smart contract as long as its customer is offline. The Custodian receives a payment for its success, or has to pay a penalty otherwise.

The Interledger protocol [22] is a protocol that creates channels to exchange cryptocurrency between different ledgers. The original whitepaper proposes two alternatives, one is called *Universal mode*. The Universal mode relies on building incentives between the participants to execute the payment transfers. The actors involved in the payment, from sender to recipient, commit all the transfers similarly to the Lightning Network and the payments will be performed backwards, from recipient to sender. This technique makes use of the general concept of Hash Time Lock Agreements (HTLAs)[7]. For instance an HTLC is an HTLA requiring strong technological support from a ledger and low trust between two parties, while a *trustline* has opposite requirements: in this way is possible to connect different ledgers. Nowadays, the new protocol adopted by Interledger is the 4th version[8], ILPV4, which avoids the setup of HTLAs but rather uses communication channels to guarantee fast transfer of low-value packets: this reduces the latency of the entire operation due to the timeouts of ledgers with slow processing time. The packets used by ILPV4 are *prepare*, *fullfill* and *reject* corresponding to

[7]HTLA: `https://interledger.org/rfcs/0022-hashed-timelock-agreements/`

[8]ILPV4: `https://interledger.org/rfcs/0027-interledger-protocol-4/`

request, respond and error respectively.

As a final example we present a technique called atomic cross-chain swaps [20, 37]. This technique allows an exchange of assets between two actors having their assets stored in different ledgers: an example would be the exchange of BTC with ETH between Alice and Bob (obviously, Alice and Bob have both Bitcoin and Ethereum wallets). The technique makes use of a pair of HTLCs, one in each ledger, and regulating the operations with the same secret that unlocks all the operations. Each time Alice and Bob want to perform an exchange, one of them need to generate a new secret.

Sidechain based solutions

Ethereum Plasma [41] was a solution that tried to improve the scalability of the Ethereum network creating smaller copies of the blockchain. Merkle trees and smart contracts run fraud proofs, generated by validator nodes of a sidechain, to verify the correctness of deposits and withdrawals with the root chain (i.e. Ethereum). However, in January 2020 the Plasma Group announced the end of the Plasma project on Medium[9].

Bitcoin Liquid [5] is a sidechain solution for Bitcoin. A Bitcoin user can move their bitcoins from the main network to the Liquid sidechain, which benefits of faster finalization and allows the creation of personal tokens to represents a certain asset. A bitcoin in Liquid is referred to L-BTC. Intermediate nodes between Bitcoin and Liquid are called *functionaries*, and have two roles: as *block signers*, who are in charge to verify and sign new Liquid blocks, and as *watchmen*, who are in charge of securing the BTC stored in Liquid. In Liquid blocks are proposed in a round robin fashion, and are deterministically appended every minute when two-third of the block signers sign the block; in case such quorum is not reached, the round is skipped. Sending BTC to Liquid is called *peg-in*; the revers process is called *peg-out*.

A live project connecting DLTs is called Cosmos [8, 25]. In Cosmos each chain is called *Zone*, and is composed by nodes that together form a so-called Application-Specific Blockchain[10]. Each Zone has full responsibility in validating internal transactions, and to be Cosmos compliant each node in a Zone needs to execute a Byzantine Fault Tolerant (BFT) consensus protocol called Tandermint [54]. The *Hubs* are intermediate blockchains enabling

[9]Medium: `https://medium.com/plasma-group/on-to-new-beginnings-e9d76b170752`

[10]Application-Specific Blockchains: `https://github.com/cosmos/cosmos-sdk/blob/master/docs/intro/why-app-specific.md`

interoperability between the Zones it connects. Everyone can publish a Hub: the first Hub is called *Cosmos Hub* and has been deployed by the Cosmos team with a native token called *Atom*.

A project similar to Cosmos is called Polkadot [40, 49]. Polkadot proposes a main DLT, called *Relay Chain*, which connects and validates the transactions of the attached sidechains, also known as *parachains*, enabling a shared security model of the whole system[11]. Differently from Cosmos, where Zones are responsible in validating transactions, in Polkadot the whole validation process is performed by the nodes of the Relay Chain, and to each parachain is assigned a particualr set of nodes in the Relay Chain. The native token of the Relay Chain is called *Dot*.

Off-chain computation based solutions

The second version of the Interledger protocol in its original whitepaper [22], opposed to the Universal mode (see State Channels technologies) is called *Atomic mode*. In this mode a set of *notaries* are the source of truth to guarantee the success / failure of a payment. The nodes involved in a payment setup first a series of transfer commitments, then the recipient sends to the notaries the receipts, and finally the notaries must agree on the execution or the abortion of the transfers. The decision is received by the recipient, who will forward it to the payment chain.

Truebit [1] is an off-chain verification tool that aims to overcome the computational limitations in a decentralized network. The area of application of Truebit is the Ethereum network. In the Truebit protocol a computational heavy task is requested by a *Taskgiver* by storing an entry in a smart contract, *Solvers* can offer themselves to solve it in exchange of a reward, and *Verifiers* check the correctness of the result. However, Solvers need to store the hash of all the intermediate steps in a Merkle tree because a subset of Verifiers called *Challengers* can disagree with the Solver answer, and can query the Solvers asking for the state of the computation during intermediate steps. This procedure is an interactive game between Solver and Challenger.

ARPA [3] is a Multiparty Computation (MPC) network. In an MPC network a set of n parties, including an adversary, wish to learn the outcome of a function $y = f(x_1, x_2, ..., x_n)$ where the participants i know only their secret input x_i, and the outcome y must be correct [9]. The ARPA network

[11]Polkadot shared security: `https://wiki.polkadot.network/docs/en/learn-security`

is composed by a set of nodes performing secured and privacy preserving computation in a MPC fashion, communicating with any blockchain with a proxy smart contract storing ARPA computation requests. The goal is to provide a verifiable scheme to prove that a certain computation has been performed off-chain by the ARPA network, it is faithfully done and the privacy of the content and the participants is guaranteed. ARPA designs a reward system to keep the nodes online in their network.

ZoKrates [11] is a framework to build off-chain verifiable computation based on zero-knowledge proofs. To ease the programming phase, the framework provides a domain-specific high level language that will be compiled into a valid zkSNARK verification scheme. The *main* function expects a list of inputs that can either be public or private, and returns at least an output value. Once compiled, the resulting flattened code is used for two tasks: 1) witness generation, finding a correct assignment of all the private and public variables, called witness, to solve the program; 2) setup phase, producing a pair of public keys, a verification and a proving key. Combining the proving key and the result of the program is possible to build a proof that needs to be sent to the verification smart contract, generated by the verification key. The smart contract takes the proof, the program's public inputs and the result and checks their correctness writing the result, true or false, in the blockchain. Currently, the ZoKrates framework supports the generation of Solidity's smart contracts.

2.2.3 A detailed example: Bitcoin Lightning Network

In this section we introduce the basics to understand how the Bitcoin Lightning Network works. The main source of this section is the book "Mastering Bitcoin" [2]. The Lightning Network allows two users to create a payment channel (see Section 2.2.1, state channels) to freely exchange bitcoins off-chain with fast speed, high security and privacy without the need to create each time a Bitcoin transactions. The overall procedure requires only two Bitcoin transactions: one to open the channel, and one to close it.

The payment channel

In the following we describe two users exchange currency though a single payment channel, a specialization of a state channel [39]. To ease the explanation two example actors will be used from now on, named Alice (A) and Bob (B). A payment channel is a virtual connection between two users, indeed the communications in between may involve additional intermediate steps. As a term of comparison, a payment channel between Alice and Bob can be seen as a TCP connection. An exchange within a payment channel between Alice and Bob is called **commitment**. This name is suitable because any exchange within the channel acts as a *promissory note* between the two actors.

Alice and Bob can open a payment channel submitting a transaction on the Bitcoin blockchain called **funding transaction**. This transaction creates a 2-of-2 multisignature address between Alice and Bob initially funded by both parties, say X BTC from Alice and Y BTC from Bob. The **capacity** of the channel is therefore X+Y: the capacity cannot be modified in a second moment (a new channel needs to be created). As long as a party has funds on their side, they can create a commitment to send bitcoins to the other party. The commitments are never submitted to the blockchain, and therefore a scenario with a lot of open channels can reach a throughput higher than the Bitcoin network: all of these commitments are stored on Alice and Bob's wallets and can be finalized into the Bitcoin network by either party at any time when closing the channel. A payment channel can be either closed unilaterally or bilaterally. When both the parties agree on closing the channel they submit a **settlement transaction** indicating the latest commitment as output of the 2-of-2 multisignature address. The unilateral option is required since one of the two parties may disappear, and the settlement transaction is not available.

Example 2.1. Alice and Bob create a payment channel funding it 0.5 BTC each. During the channel life Alice and Bob create 3 commitments: 1) 0.3 BTC to Alice; 2) 0.5 BTC to Bob; 3) 0.2 BTC to Bob. The final state is 0.1 BTC to Alice and 0.9 BTC to Bob. Finally, Alice and Bob create a settlement transaction that sends 0.1 BTC to Alice and 0.9 BTC to Bob. The transactions included on the Bitcoin blockchain will only be two: the funding transaction and the settlement transaction.

What happens if Alice submits as settlement transaction a commitment that is not the latest? To prevent this issue, the Lightning Network exploits the concepts of **timelocks** and **hashlocks** (see Section 2.2.1, state channels). Anytime the two parties sign the i_{th} commitment on the channel, a new pair of secrets (s_iA, s_iB) is automatically created, and each output of the *ith* commitment is hashlocked with corresponding secret: H(s_iA) locks the amount of BTC to send to Alice, and H(s_iB) locks the amount of BTC to send to Bob. At the same time, the secrets ($s_{i-1}A$, $s_{i-1}B$) of the *i-1th* commitment will be revealed. In Lightning Network, these secrets are also known as *revocation keys*. As soon as Alice closes unilaterally the channel settling the commitment j, the operation automatically sends the funds to Bob according to commitment j, and Alice needs to wait a timelock of T blocks to get her funds. Bob can get Alice's funds if he provides the revocation key s_j. There are two possibilities:

1. if j is the latest commitment, Bob does not know the s_j;
2. if j is NOT the latest commitment, then Alice tried to cheat Bob, Bob knows s_j and can punish Alice, within the timelock, by unlocking taking the funds that are meant to go to her.

Example 2.2. In Example 2.1 each one of the three commitments has its associated revocation key pairs: (s_1A, s_1B), (s_2A, s_2B), (s_3A, s_3B). If Alice closes the channel settling the latest commitment, Bob gets immediately 0.9 BTC, and has no knowledge of s_3A because this revocation key has been generated together with the third commitment and it is not revealed. In the case Alice closes the channel settling the first commitment (the one which assigns 0.8 BTC to Alice) Bob gets immediately 0.2 BTC, and has time (determined by the timelock) to claim the 0.8 BTC as punishment to Alice because he kwows s_1A.

A network of payment channels

If Alice wishes to send BTC to Carl through a payment channel, they need to create a new one, but it requires other BTC that Alice might not have. However, it is be possible to find a path of payment channels that allows Alice to send BTC to Carl through intermediate payment steps: that's why this technology has the name of "Network". Unfortunately, routing a payment is not simply a problem of finding the shortest path. Users are allowed to ask for a fee for their service, and a payment of X BTC cannot go through a channel that has a capacity of Y < X. Finally, the nodes need to be online to be able to perform a payment.

Example 2.3. Alice wishes to send to Carl 0.5 BTC. Alice and Carl do not have an open payment channel, and they cannot open a new one. Therefore Alice queries the Lightning Network to find a path of payment channels from her to Carl. She finds two paths: a 3 hops path P1 and a 5 hops path P2. However, the total fee in P1 is higher than in P2.

Once Alice has decided the path to send BTC to Carl, the payment can start to take place. However, after a setup phase, the payment steps are performed backwards (from recipient to payer). This is necessary because if Alice pays Bob 0.5 BTC[12] and asks him to forward them to the next point, Bob can simply ignore this step and keep the BTC. The Bitcoin Lightning Network solves this problem by means of Hash Time Locked Contracts (HTLCs). An HTLC is an agreement between a payer and a recipient, allowing the recipient to claim the payment by unlocking the hashlock before the timelock expires (see Section 2.2.1, state channels). In the Lightning Network, after a path of payments has been chosen between Alice and Carl, it begins the setup of HTLCs between all the participants. Carl generates a secret s and its hash value h. Carl sends the h to Alice who creates an HTLC with hashlock h and timelock T with the next hop of the payment route (i.e. a user with whom Alice has an open payment channel). Each intermediate step follows the example of Alice, but decreasing the timelock of a value Δ at each step. The last intermediate step opens an HTLC with Carl with hashlock h and timelock $T - n\Delta > 0$ (assuming n intermediate hops).

Example 2.4. (Steps 2, 3 in Figure 3) Alice chooses the 3 hop route. The intermediate nodes are labeled as I1, I2 and I3. Carl sends h to Alice. Alice sets an HTLC(h, t) with I1; I1 sets an HTLC(h, $t - \Delta$) with I2; I2 sets an HTLC(h, $t - 2\Delta$) with I3; I3 sets an HTLC(h, $t - 3\Delta$) with Carl. The payment of each HTLC is of 0.5 BTC (fees are ignored for simplicity).

After all of the HTLC contracts have been set up, Carl reveals the value s to its predecessor that triggers the last payment to Carl. At this point this actor knows s, so they can unlock the payment from the previous step: they have the intention to do it since they already paid Carl. This operation goes backward the payment route until Alice receives s, finalizing the payment.

Example 2.5. (Steps 4.x in Figure 3) Carl knows s since he generated it. He unlocks the HTLC with I3 thus receiving 0.5 BTC from them. At this point I3 learns s and can unlock the payment promised by I2. I3 can do that also because the timelock between I2 and I3 is not expired since it is longer than the timelock between I3 and Carl (which did not expire either). I3 unlocks the payment and I2 learns s. I2 unlocks the payment from I1. I1 unlocks the payment from Alice. At this point Alice successfully paid Carl 0.5 BTC without the creation of any new payment channel.

Without considering the fees, the intermediate nodes did not globally gain / lose any funds. However, for each payment channel each user has

[12]The fee is omitted for simplicity.

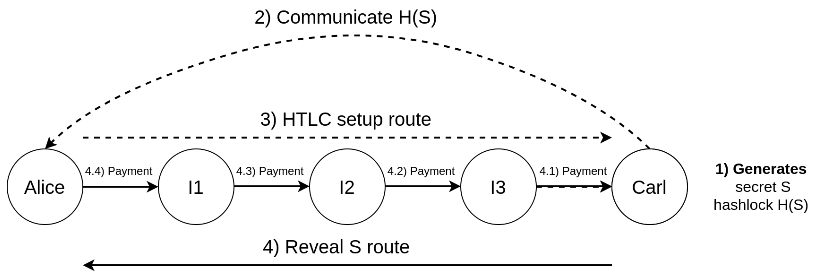


Figure 3: Payment from Alice to Carl crossing I1, I2 and I3. The dashed arrows represent the HTLC setup; the continuous arrows the payment.

now a different amount available. Say the route involves Alice, Bob, Carl in this order, then Bob gains +0.5 BTC from Alice, and loses -0.5 BTC to Carl. In order to preserve the privacy of the nodes within the routing, the network communications between the nodes follow the Onion routing protocol [44].

3 Thesis proposal

In this section we present some open problems that the blockchain technology is currently facing, and we describe the research activity we propose to solve these problems. In particular, Section 3.1 presents the open problems, Section 3.2 shows the research directions to take during the PhD, described in detail in Sections 3.2.1 and 3.2.2.

3.1 Open problems

Scalability Nowadays, blockchain-based applications suffer from issues that are inherited from the underlying architecture. The "blockchain trilemma" [14] states that a blockchain system cannot be fully decentralized, secure and scalable at the same time. For example, Bitcoin and Ethereum both lack of scalability because the consensus algorithm they run, PoW, focuses on preventing the spam of incorrect transactions rather than fast processing time. While this makes the network decentralized, because all the nodes run the same protocol, and secure, it comes at the cost of a low scalability, i.e. a low transaction per second (tx/s) rate, and adding a new node to the network does not improve it. Bitcoin implements a digital and decentralized

payment system that, nowadays, can process about 6 tx/s[13], very small if compared to popular payment systems like VISA; Ethereum builds a decentralized platform running DApps and can process about 13 tx/s[14], generated by all the DApps running on Ethereum, making the Ethereum network not a long-term scalable solution. However, in these 10 years of existence, Bitcoin never changed its consensus protocol because it seems to be the one to offer better decentralization and security guarantees: the findings presented in [15], that surveyed the scientific and industrial communities to classify in a [1-5] range the key properties of blockchain architectures, confirm this view, showing that the highest scores for "decentralization" and "security" belong to the architectures sharing the PoW consensus and a permissionless setting. Therefore, security and decentralization seem to have more importance over scalability in permissionless settings. Tweaking the protocol parameters, such as the block size or the average difficulty of PoW, can help but they would not drastically improve the scalability.
Improving the scalability of blockchain-based applications can be done by adopting permissioned architectures but, as stated by the trilemma, this choice requires to give up either security or decentralization. Another solution is to design and implement an alternative blockchain model from scratch (e.g. as planned with Ethereum 2.0 [14]). Layer-2 technologies, instead, tackle this problem by executing the operations off-chain, but a proof of execution must to be stored, or verified, on chain. This preserves the blockchain features and implementation (unless minor changes).

Privacy An important property of public blockchain is the transparency of the operations and the data. Indeed, the blockchain is typically considered as an immutable and transparent source of data, making it able to "create trusted setups in a trustless fashion" as pointed out in [35]. In Bitcoin and Ethereum this is a design choice that contributes to the security of the network, because any peer joining can verify the existing blocks and refuse to participate if the transactions are not correct. As a consequence, breaking the system in favour of a single peer is not viable because no other peer will join, and the cryptocurrency value will collapse nullifying the gains to the attacker. However, this comes at the cost in terms of privacy of the users. Every transaction is stored with the sender and receiver addresses, together with the sent amount, in the blockchain forever: to mitigate this, in Bitcoin

[13]`https://www.blockchain.com/charts/transactions-per-second`

[14]`https://blockchair.com/ethereum/charts/transactions-per-second?interval=1m`

it is suggested to use a different address for each transaction to limit the knowledge a malicious person can have on the wealth of a Bitcoin user. The pseudo-anonymity protects Bitcoin users, however it is possible to clusterize the Bitcoin addresses to understand whether they belong to the same user [32], and knowing the identity behind an address de-anonymizes the whole cluster [4]. Moreover, this limits the data that an application is allowed to store due to regulations protecting privacy-sensitive information.
In private networks this issue does not arise, but a private network typically comes with a permissioned architecture (see Table 1), making the system federated instead of P2P. In layer-2 technologies the data may be stored off-chain but they require a proof on-chain. The privacy depends on the format of such proof: in the Lightning Network the movements within a channel are private, but the amount to open the channel is in clear.

3.2 Research directions

Nowadays, an important contribution to blockchain-based systems would consist in finding a way to improve the scalability of the system and to guarantee the privacy of sensitive data, without requiring a drastical modification (e.g. an hard-fork) to the underlying architecture: this is the challenge that layer-2 technologies are tackling. There are two major use cases that involve the blockchain technology: the asset exchange, and the execution of decentralized programs (smart contracts). Both of them suffer from the problems of scalability and privacy as described above. It does not exist a standard way to implement a layer-2 model on a blockchain system because it strongly depends from the use case. The plan of this PhD is to fill this gap by 1) analyzing the Payment Channels, evaluate their weaknesses and propose a alternatives to solve them; 2) propose a methodological framework to integrate a layer-2 model to blockchain-based applications.

3.2.1 Analyzing the Payment Channels

This research direction will consist in investigating and improving existing layer-2 technologies that support a cryptocurrency exchange: the Lightning Network is a good candidate since the mainnet was launched in January 2018, and nowadays is composed by about 14.000 online nodes and 37.000 open channels exchanging an amount of about 1100 BTC worth 11 millions dollars[15]. Studying the topology of the network is indeed an interesting research direction [33, 47], that allows us to understand how the peers in a

[15]`https://1ml.com/statistics` (4 October 2020)

state channel-based network connect to each other, how it evolves over the time, and identify the presence of patterns that can be compared to other networks. The findings are also beneficial to evaluate the security: in [34] the authors show that the network is resistant to random node failure, but it is characterized by a few nodes with an high degree, behaving as intermediate payment hubs, that may disconnect the network if attacked. As a final example, in [21] the authors show that it is possible to disclose the balance a node has in a channel, an information that is secret to preserve privacy, asking to that node to forward many payments, that will not be finalized, forwarding wrong hashlocks. An additional investigation consists in the analysis of the routing protocol. The channels of the network are composed by both a capacity and a weight (a fee). A payment consists in solving a flow-optimization problem, because it needs to find a path that can either minimize the number of hops or the total fee to pay, while finding a route whose capacity can sustain the payment. The problem is more difficult: while the capacity of a channel is known, for privacy motivations the balance that a node actually has is not (unless disclosed, see [21]), therefore, a payment needs to be split in smaller ones. The authors in [18] lists the properties a routing algorithm in a payment channel network should satisfy: effectiveness, efficiency, scalability, cost-effectiveness, and privacy. Given the amount of variables, it may not be feasible to define an optimal protocol that works in any condition and satisfies all the requirements. For example, one extreme could require the shortest paths as possible: this could be solved by having a few highly connected hubs, but they would be vulnerable points of failure; another extreme could require to limit the points of failure by reducing degree of each node, then making the network more resistant, but such approach might not be useful for real-world users.

Road map The plan is to create a crawler, for example exploiting a Lightning Network node, to get several snapshots of the network. In this way, we can build an historical trace of the evolution of the network, that reflects the behavior of its users. With this dataset, the next steps can be the following:

1. As a preliminary step from a single snapshot, extract basic statistical properties of the channels to understand their characteristics;

2. Study the topology of a snapshot of the P2P overlay to answer to questions like "Is the overlay a structured or an unstructured network? Are there patterns? What are the security implications for the users?", and compare the results with other previous works;

3. Based on the findings on a single snapshot, we plan to exploit the historical trace to understand whether the topology changes over the time or if it is stationary, and understand the motivations that could be behind a dynamic or static network;

4. Design new routing protocols for the Lightning Network. As anticipated above, find a suitable routing over a payment channel network needs to take into consideration several restrictions that makes this problem difficult. The topological analysis will give us the insights to design a routing protocol that works on the realistic behavior of the network.

3.2.2 Definition of a methodological framework

In this research direction we plan to design a methodological framework to integrate layer-2 models to blockchain systems based on decentralized computation (for example, smart contracts). When developing a blockchain-based application, important decisions need to be taken to find a trade-off between auditability, security, scalability and privacy, therefore designing which operation, and data, goes on-chain and which off-chain: with on-chain we benefit auditability and security, while off-chain scalability and privacy. The introduction of a second layer complicates the design, but it can provide the scalability and privacy without relying on third-party platforms, keeping the overall architecture decentralized. However, there is an important point when designing a layer-2 architecture: finding a technique that can prove on-chain that an operation, or a data, has been computed, or stored, off-chain. The goal of the framework is to return the two-layered architectural composition that best fits a list of operations and requirements of an application. In order to develop the decisions the framework has to take, we believe it is important to study some test applications with their own requirements. During the first year of the PhD we have been working on two test applications: Recommender Systems (RS) and Responsible Disclosures (RD).
A RS [46] is a platform collecting user's opinions, called reviews, on products or services, called items, so that they can recommend similar items to the users based on their habits and tastes. Amazon, Tripadvisor and Yelp are a few examples. However, nowadays RSs are typically built on top of architectures with centralized governance, leading to problems like review manipulations as described in [48]: as an example, in Australia an hotel was fined 2.2 million dollars for violating consumer law by manipulating reviews

on Tripadvisor[16]. We consider RS to be a good test application since the research community is actively investigating in that direction [7, 48, 51], as well as the industry [16, 28, 45]: for example, in [48] the authors tackle the problem of review manipulation storing customer reviews in an architecture mixing Ethereum and the Inter-Planetary File System (IPFS), a BitTorrent-like file system.
A RD [13, 53] is a notorious procedure to manage the disclosure of vulnerabilities, either software or hardware, to the general public without damaging the vendor of the defected product. RD is a trade-off between Full Disclosure, the existence of a vulnerability is communicated to the public, and Non Disclosure, a vulnerability is communicated only to the vendor, and it is an agreement, between the finder and the vendor, which allows the finder to warn the general public about the vulnerability if the vendor does not fix it in time. However, as stated by [26], finders have been intimidated by vendors to not publish their findings, risking a lawsuit. This is an interesting use case to adopt a transparent and immutable ledger to track the steps of the RD procedure.

Road map We have been working on the RS use case since a year, that resulted on a publication about a general smart contract architecture for RSs in [29]. We have enriched the proposal with the addition of a rewarding and reputation systems to mitigate the fees a user has to pay [31]. We identified the solution not scalable, and limiting the amount of data that can be stored. For this reason, we are investigating a multi-chain approach to split the data and the operations between two chains, and connect the operations with the atomic-swap technique: currently, we implemented a single-chain prototype [30]. We plan to enrich the architecture designing a system to include a user trust network in order to limit the reviews only to the people belonging to the user's network. To express the trust connections, our idea is to use a framework to express connections like RT [27]. However, we expect that processing such connections, stored on a smart contract, to not be effective to do on-chain, and may require privacy-sensitive data that cannot either be stored on-chain. The research plan is the following:

1. Study the scalability and privacy factors of the current single-chain design, and provide a threat model;

2. Design a layer-2 solution based on multi-chain and off-chain computa-

[16]`https://edition.cnn.com/travel/article/australia-tripadvisor-hotel-fined-trnd/index.html`

tion, and evaluate its impact in terms of scalability and data privacy;

3. Evaluate the adoption of zero-knowledge proofs to support the inclusion of private data when processing the trust network.

The work on RD is a (remote) collaboration with the Communication and Networking department of Aalto University (Espoo, Finland), and it consists in supporting the RD procedure with a multi-chain architecture, and automating critical steps of the process, to preserve the finder's privacy. An high level design of the system can be found in [26]. The proposed steps are the following:

1. Build a multi-chain prototype out of the high level design;
2. Evaluate with a threat model, and show that it supports the RD procedure while protecting the finder's identity.

The experience acquired during these works will help us to define a general methodology for the development of scalable and privacy preserving layer-2 solutions for blockchains.

References

[1] *A scalable verification solution for blockchains.* URL: `https://people.cs.uchicago.edu/~teutsch/papers/truebit.pdf`. (accessed: 4 October 2020).

[2] Andreas M Antonopoulos. *Mastering bitcoin: Programming the open blockchain.* " O'Reilly Media, Inc.", 2017.

[3] *ARPA Whitepaper.* URL: `https://docsend.com/view/t69gzij`. (accessed: 4 October 2020).

[4] Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. "Deanonymisation of clients in Bitcoin P2P network". In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security.* 2014, pp. 15–29.

[5] *Bitcoin Liquid.* URL: `https://docs.blockstream.com/liquid/technical_overview.html`. (accessed: 4 October 2020).

[6] Vitalik Buterin. "Chain interoperability". In: *R3 Research Paper* (2016).

[7] Fran Casino and Constantinos Patsakis. "An efficient blockchain-based privacy-preserving collaborative filtering architecture". In: *IEEE Transactions on Engineering Management* (2019).

[8] *Cosmos*. URL: `https://cosmos.network/intro`. (accessed: 4 October 2020).

[9] Ronald Cramer, Ivan Bjerre Damgård, and Jesper Buus Nielsen. *Secure multiparty computation*. Cambridge University Press, 2015.

[10] Jacob Eberhardt and Jonathan Heiss. "Off-chaining models and approaches to off-chain computations". In: *Proceedings of the 2nd Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*. 2018, pp. 7–12.

[11] Jacob Eberhardt and Stefan Tai. "c". In: *2018 IEEE 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)*. IEEE. 2018, pp. 1084–1091.

[12] Jacob Eberhardt and Stefan Tai. "On or off the blockchain? Insights on off-chaining computation and data". In: *European Conference on Service-Oriented and Cloud Computing*. Springer. 2017, pp. 3–15.

[13] *Economics of Vulnerability Disclosure*. URL: `https://www.enisa.europa.eu/publications/economics-of-vulnerability-disclosure`. (accessed: 4 October 2020).

[14] *Ethereum Sharding*. URL: `https://eth.wiki/sharding/Sharding-FAQs`. (accessed: 4 October 2020).

[15] Martin Garriga et al. "Blockchain and cryptocurrencies: A classification and comparison of architecture drivers". In: *Concurrency and Computation: Practice and Experience* (2020), e5992.

[16] *Gastroadvisor whitepaper*. URL: `https://www.gastroadvisor.com/whitepaper.pdf`. (accessed: 4 October 2020).

[17] Arthur Gervais et al. "On the security and performance of proof of work blockchains". In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016, pp. 3–16.

[18] Lewis Gudgeon et al. "Sok: Layer-two blockchain protocols". In: *International Conference on Financial Cryptography and Data Security*. Springer. 2020, pp. 201–226.

[19] *Hash Time Locked Contracts*. URL: `https://en.bitcoin.it/wiki/Hash_Time_Locked_Contracts`. (accessed: 4 October 2020).

[20] Maurice Herlihy. "Atomic cross-chain swaps". In: *Proceedings of the 2018 ACM symposium on principles of distributed computing.* 2018, pp. 245–254.

[21] Jordi Herrera-Joancomarti et al. "On the difficulty of hiding the balance of lightning network channels". In: *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security.* 2019, pp. 602–612.

[22] *Interledger.* URL: `https://interledger.org/`. (accessed: 4 October 2020).

[23] Niclas Kannengießer et al. "Bridges Between Islands: Cross-Chain Technology for Distributed Ledger Technology". In: *Proceedings of the 53rd Hawaii International Conference on System Sciences.* 2020.

[24] Tommy Koens and Erik Poll. "Assessing interoperability solutions for distributed ledgers". In: *Pervasive and Mobile Computing* 59 (2019), p. 101079.

[25] Jae Kwon and Ethan Buchman. "Cosmos: A network of distributed ledgers". In: *URL https://cosmos. network/whitepaper* (2016).

[26] *Leveraging Interledger Functionality in Automated Responsible Disclosure of Vulnerabilities.* URL: `https://www.sofie-iot.eu/news/leveraging-interledger-functionality-in-automated-responsible-disclosure-of-vulnerabilities`. (accessed: 4 October 2020).

[27] Ninghui Li and John C Mitchell. "RT: A role-based trust-management framework". In: *Proceedings DARPA Information Survivability Conference and Exposition.* Vol. 1. IEEE. 2003, pp. 201–212.

[28] *Lina.Review.* URL: `https://lina.network/lina-review/`. (accessed: 4 October 2020).

[29] Andrea Lisi et al. "A Smart Contract Based Recommender System". In: *International Conference on the Economics of Grids, Clouds, Systems, and Services.* Springer. 2019, pp. 29–42.

[30] Andrea Lisi et al. "Practical application and evaluation of Atomic Swaps to Recommendation Systems (SUBMITTED)". In: *17th International Conference on Advanced and Trusted Computing (ATC).* IEEE. 2020.

[31] Andrea Lisi et al. "Rewarding reviews with tokens: an Ethereum-based approach (SUBMITTED)". In: *Future Generation Computer Systems* (2020).

[32] Damiano Di Francesco Maesa, Andrea Marino, and Laura Ricci. "Data-driven analysis of Bitcoin properties: exploiting the users graph". In: *International Journal of Data Science and Analytics* 6.1 (2018), pp. 63–80.

[33] Stefano Martinazzi. "The evolution of Lightning Network's Topology during its first year and the influence over its core values". In: *arXiv preprint arXiv:1902.07307* (2019).

[34] Stefano Martinazzi and Andrea Flori. "The evolving topology of the Lightning Network: Centralization, efficiency, robustness, synchronization, and anonymity". In: *Plos one* 15.1 (2020), e0225966.

[35] Daniele Mazzei et al. "A Blockchain Tokenizer for Industrial IOT trustless applications". In: *Future Generation Computer Systems* 105 (2020), pp. 432–445.

[36] Patrick McCorry et al. "Pisa: Arbitration outsourcing for state channels". In: *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*. 2019, pp. 16–30.

[37] Mahdi H Miraz and David C Donald. "Atomic cross-chain swaps: development, trajectory and potential of non-monetary digital token swap facilities". In: *Annals of Emerging Technologies in Computing (AETiC) Vol* 3 (2019).

[38] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. Tech. rep. Manubot, 2019.

[39] *Payment channels*. URL: `https://en.bitcoin.it/wiki/Payment_channels`. (accessed: 4 October 2020).

[40] *Polkadot*. URL: `https://wiki.polkadot.network/docs/en/`. (accessed: 4 October 2020).

[41] Joseph Poon and Vitalik Buterin. "Plasma: Scalable autonomous smart contracts". In: *White paper* (2017), pp. 1–47.

[42] Joseph Poon and Thaddeus Dryja. *The bitcoin lightning network: Scalable off-chain instant payments*. 2016.

[43] *Raiden Network*. URL: `https://docs.raiden.network/`. (accessed: 4 October 2020).

[44] Michael G Reed, Paul F Syverson, and David M Goldschlag. "Anonymous connections and onion routing". In: *IEEE Journal on Selected areas in Communications* 16.4 (1998), pp. 482–494.

[45] *Revain*. URL: `https://revain.org/`. (accessed: 4 October 2020).

[46] Francesco Ricci, Lior Rokach, and Bracha Shapira. "Recommender systems: introduction and challenges". In: *Recommender systems handbook*. Springer, 2015, pp. 1–34.

[47] Elias Rohrer, Julian Malliaris, and Florian Tschorsch. "Discharged Payment Channels: Quantifying the Lightning Network's Resilience to Topology-Based Attacks". In: *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE. 2019, pp. 347–356.

[48] K Salah, A Alfalasi, and M Alfalasi. "A Blockchain-based System for Online Consumer Reviews". In: *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE. 2019, pp. 853–858.

[49] Stefan Schulte et al. "Towards Blockchain Interoperability". In: *International Conference on Business Process Management*. Springer. 2019, pp. 3–10.

[50] *Segregated witness*. URL: `https://en.bitcoin.it/wiki/Segregated_Witness`. (accessed: 4 October 2020).

[51] Mike Sharples and John Domingue. "The blockchain and kudos: A distributed system for educational record, reputation and reward". In: *European Conference on Technology Enhanced Learning*. Springer. 2016, pp. 490–496.

[52] V. A. Siris et al. "Interledger Approaches". In: *IEEE Access* 7 (2019), pp. 89948–89966. ISSN: 2169-3536. DOI: `10.1109/ACCESS.2019.2926880`.

[53] Adam Stone. "Software flaws, to tell or not to tell?" In: *IEEE Software* 20.1 (2003), pp. 70–73.

[54] *Tandermint*. URL: `https://docs.tendermint.com/master/introduction/what-is-tendermint.html`. (accessed: 4 October 2020).

[55] Gavin Wood et al. "Ethereum: A secure decentralised generalised transaction ledger". In: *Ethereum project yellow paper* 151.2014 (2014), pp. 1–32.

[56] Karl Wüst and Arthur Gervais. "Do you need a blockchain?" In: *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*. IEEE. 2018, pp. 45–54.

[57] Qiheng Zhou et al. "Solutions to scalability of blockchain: A survey". In: *IEEE Access* 8 (2020), pp. 16440–16455.