A Circuit Approach to Constructing Blockchains on Blockchains

Ertem Nusret Tas
Stanford University, USA

David Tse
Stanford University, USA

Yifei Wang
Stanford University, USA

Abstract

Recent years have witnessed an explosion of blockchains, each with an open ledger that anyone can read from and write to. In this multi-chain world, an important question emerges: how can we build a more secure overlay blockchain by reading from and writing to a given set of blockchains? Drawing an analogy with switching circuits, we approach the problem by defining two basic compositional operations between blockchains, serial and triangular compositions, and use these operations as building blocks to construct general overlay blockchains. Under the partially synchronous setting, we have the following results: 1) the serial composition, between two certificate-producing blockchains, yields an overlay blockchain that is safe if at least one of the two underlay blockchains is safe and that is live if both of them are live; 2) the triangular composition between three blockchains, akin to parallel composition of switching circuits, yields an overlay blockchain that is safe if all underlay blockchains are safe and that is live if over half of them are live; 3) repeated composition of these two basic operations can yield all possible tradeoffs of safety and liveness for an overlay blockchain built on an arbitrary number of underlay chains. The results are also extended to the synchronous setting.

2012 ACM Subject Classification Security and privacy → Distributed systems security

Keywords and phrases interchain consensus protocols, serial composition, triangular composition, circuits

Acknowledgements We thank Dionysis Zindros for several insightful discussions on this project. This paper and the concurrent related work in which blockchains were analyzed as 'virtual parties' [48] both came out of many fruitful discussions about blockchain composability among Ertem Nusret Tas, David Tse, Yifei Wang and Dionysis Zindros, when they were all at Stanford University. This research was funded by a Research Hub Collaboration agreement with Input Output Global Inc. Ertem Nusret Tas is supported by the Stanford Center for Blockchain Research.

1 Introduction

1.1 Background

Bitcoin, invented by Nakamoto in 2008 [32], is the first blockchain with a public ledger, which anybody can read from and write arbitrary data. Since then, there has been a proliferation of such blockchains. Each of them is a consensus protocol run by its own set of validators. Together, these blockchains form a *multi-chain world*, communicating with each other through bridging protocols which read from and write to the blockchains.

As a consensus protocol, a fundamental property of a blockchain is its *security*: a blockchain is secure if the ledger it provides is safe and live. The security is supported by

The authors are listed alphabetically.

the blockchain's set of validators. In a multi-chain world, a natural question arises: given a set of existing blockchains, how to build a more secure protocol, an *overlay* blockchain, by only reading from and writing to the ledgers of the individual *underlay* blockchains? In other words, how to build a blockchain on blockchains?

This problem has received attention recently, and there have been two main approaches to this problem in the literature. The first approach is *interchain timestamping*. In the context of two blockchains, data on one blockchain is timestamped to another blockchain, and a more secure ledger is obtained by reading the ledger of the first chain using the timestamps on the second chain to resolve forks. An interchain timestamping protocol was proposed in [23] to allow a Proof-of-Stake (PoS) chain to borrow security from Bitcoin. In that work, Bitcoin is assumed to be secure and the problem was to determine the optimal security properties that can be achieved by the PoS chain. A more symmetric formulation is considered in [42], where none of the individual chains is assumed to be secure. Moreover, the interchain timetamping protocol is extended to more than 2 chains, where the timestamping proceeds in a *sequential* manner, where the chains are ordered and the first chain timestamps to a second chain which timestamps to a third chain, etc. The main security result in [42] is that the overlay blockchain is safe if at least one of the underlay blockchains is safe, and is live if all of the underlay blockchains are live.

In the second approach, an analogy is drawn between the multiple blockchains and the multiple validators in a blockchain, and an overlay blockchain is built by running a consensus protocol on top of the underlay blockchains by treating them as validators. This idea was first sketched out in Recursive Tendermint [3] in the context of the Cosmos ecosystem, consisting of numerous application specific blockchains each running the Tendermint consensus protocol [12]. Recently, this idea was made more precise and concrete by Trustboost [41], where the validator role of each underlay blockchain is instantiated by a specialized smart contract. These simulated validators send messages between the underlay blockchains via a cross-chain communication protocol (CCC) to implement a variant [9] of the Hotstuff consensus protocol [46]. The main security result in [41] is that, in a partially synchronous network, the overlay blockchain is secure (safe and live) if more than 2/3 of the underlay blockchains are secure.

1.2 Problem Motivation

Even though interchain timestamping and Trustboost both propose a construction of blockchains on blockchains, their security statements are quite different in nature. First, the conditions for safety and liveness are separate for the interchain timestamping protocol, while they are coupled in Trustboost. Since loss of safety and loss of liveness may have different impacts on a blockchain, separating out when safety and liveness are achieved is useful.

Second, when the safety condition of the overlay blockchain depends *only* on the safety of the underlay blockchains, one can immediately infer the *accountable safety* [14] (also known as the forensics property [40]) of the overlay blockchain in terms of the accountable safety of the underlay chains. Accountable safety states that if the adversary controls a large fraction of the validators and causes a safety violation, all protocol observers can irrefutably identify the adversarial validators responsible for the safety violation. It is thus a strengthening of the traditional safety guarantees of consensus protocols. When the overlay blockchain's safety depends only on the underlay chains' safety, a safety violation on the overlay would imply safety violations on (some of) the underlays. Therefore, if the underlay chains satisfy accountable safety, when the overlay's safety is violated, all protocol observers would identify the responsible adversarial validators of the underlay chains, implying accountable safety for the overlay blockchain.

Whereas there are protocols satisfying accountable safety [33, 40], adversarial validators responsible for liveness violations cannot be held accountable in the same sense as accountable safety [43]. Thus, to infer the accountable safety of an overlay blockchain, its safety should depend *only* on the safety but not the liveness of the underlay blockchains. Indeed, if the overlay blockchain loses safety due to liveness violations on the underlay blockchains, it might not be possible to identify the responsible adversarial validators.

Finally, separating safety and liveness of the overlay blockchain and characterizing their dependence on the safety and liveness of the underlay chains enables achieving greater resilience than when security is based on the number of secure underlay chains, with safety and liveness *coupled*. For illustration, Trustboost [41] shows security of the overlay blockchain only when over 2/3 of the underlay chains are secure, *i.e.*, *both* safe and live. Note that this is optimal if the overlay chain's security were to be based on the number of secure underlay chains. However, by separating safety and liveness, we can achieve safety for the overlay blockchain if over 2/3 of the underlay chains are safe, and liveness if over 2/3 of the underlay chains are live, where the sets of safe and live underlay chains need not be the *same*. This implies that the overlay blockchain can be secure, even when up to 2/3 of the underlay chains are not both safe and live, *i.e.*, secure! While this may sound puzzling, there are no hidden tricks at play here. Indeed, any two quorums of underlay chains required for the liveness of the overlay blockchain must intersect at an underlay chain whose safety is required for the overlay's safety. In contrast, using our notation, Trustboost would require both liveness quorums to intersect with a safety quorum at over 2/3 of the underlay chains.

Interchain timestamping protocols provide an inspiration for security statements separating out safety and liveness, but they only achieve one particular tradeoff between safety and liveness: they favor safety strongly over liveness. This is because safety of the overlay blockchain requires only one of the underlay blockchains to be safe, while liveness of the overlay blockchain requires all of the underlay blockchains to be live. Therefore, two natural questions arise: 1) What are all the tradeoffs between safety and liveness which can be achieved? 2) How can we construct overlay blockchains that can achieve all the tradeoffs? The main contributions of this paper are to answer these two questions.

1.3 Security Theorems

Consider overlay blockchains instantiated with k underlay chains (*cf.* Section 3 for a formal definition of overlay blockchains). We say a tuple (k, s, l) is achievable if one can construct an overlay blockchain such that
a. If s or more underlay blockchains are safe, the overlay blockchain is safe.
b. If l or more underlay blockchains are live with constant latency after the global stabilization time (GST), the overlay blockchain is live with constant latency after GST.
Going forward, when referring to the liveness of a blockchain, we mean liveness with constant latency after GST.

We identify all achievable tuples and provide a protocol achieving them (Fig. 1).

▶ **Theorem 1.** *Consider the partially synchronous setting. For any integers $k \geq 1$, l and s such that $\lfloor k/2 \rfloor + 1 \leq l \leq k$ and $s \geq 2(k-l)+1$, the tuple (k, s, l) is achievable.*

In particular, the tuple $(k, \lceil \frac{2k}{3} \rceil, \lceil \frac{2k}{3} \rceil)$ is achievable, *i.e.*, there is an overlay blockchain that is safe if more than 2/3 of the underlay chains are safe, and is live if more than 2/3 of the underlay chains are live. This implies that the overlay is safe and live if more than 2/3 of the underlay chains are safe and more than 2/3 of the underlay chains are live. Note that this is a strictly stronger security guarantee than Trustboost, which is guaranteed to be safe and

live if more than 2/3 of the underlay chains are both safe and live; *i.e.*, the *same* chains need to be safe and live in this latter statement. Moreover, Theorem 1 includes also asymmetric operating points where $s \neq \ell$.

The next theorem gives a matching impossibility result.

▶ **Theorem 2** (Informal, Theorem 14)**.** *Consider the partially synchronous network. For any integers $k \geq 1$, l and s such that $\lfloor k/2 \rfloor + 1 \leq l \leq k$ and $s < 2(k-l)+1$, no protocol can satisfy the following properties simultaneously:*
a. If s underlay blockchains are safe and all underlay blockchains are live*, the overlay blockchain is safe.*
b. If l underlay blockchains are live and all underlay blockchains are safe*, the overlay blockchain is live.*

The same result holds for any integers $k \geq 1$ and $l \leq k/2$.

Theorem 2 shows the optimality of the result in Theorem 1 in a strong sense: even if we allow the safety or liveness of the overlay blockchain to depend on *both* safety and liveness of the underlay chains (Fig. 1), the security guarantee of the overlay blockchain cannot be improved. In other words, under partial synchrony, liveness of the underlay chains have no effect on the safety of the overlay blockchain. Theorems 1 and 2 are proven in Sections 5.2 and 6.2 respectively.

We also characterize the security properties achievable in the *synchronous* network.

▶ **Theorem 3** (Informal, Theorems 17 and 19)**.** *Consider the synchronous network. For any integers $k \geq 1$, l, s and b, one can construct an overlay blockchain as described below if and only if $\lfloor k/2 \rfloor + 1 \leq l \leq k$, $s \geq 2(k-l)+1$, and $b \geq k-l+1$:*
a. If s underlay blockchains are safe, or *b underlay blockchains are both safe and live, the overlay blockchain is safe.*
b. If l underlay blockchains are live, the overlay blockchain is live.

Theorem 3 shows that under synchrony, unlike partial synchrony, the overlay blockchain has better safety guarantees when the underlay chains are *both* safe and live. On the other hand, Theorem 3 implies that if we require the safety (liveness) of the overlay blockchain to depend only on the number of safe (live) underlay chains (*i.e.*, restrict b to be 0), we cannot achieve any better resilience under synchrony compared to partial synchrony. Security in the synchronous network is discussed further in Sections 7 and 8.

1.4 Construction via Blockchain Circuits

We now give insight into our methods using the example of the (k, s, l) tuples under partial synchrony. For $k = 2$, the only achievable tuple in Theorem 1 is $(2, 1, 2)$, which can be achieved by timestamping. For $k = 3$, we have $(3, 1, 3)$ and $(3, 3, 2)$ as achievable tuples. $(3, 1, 3)$ can be achieved by sequential interchain timestamping across 3 chains. This is the strongly safety favoring overlay blockchain (extremal of the tradeoff in Figure 1). $(3, 3, 2)$ represents a liveness-favoring overlay blockchain: it is safe if *all* 3 underlay blockchains are safe, and is live if at least 2 of the 3 underlay blockchains are live. No existing construction is known to achieve this operating point. Our solution to achieve all tuples in Theorem 1 consists of two steps and described in Sections 4 and 5:
1. We provide a construction that achieves $(3, 3, 2)$ by drawing an analogy to Omission-Fault Tolerant (OFT) protocols, where validators only commit omission faults (analogous to loss of liveness of a safe underlay blockchain) but no Byzantine faults.

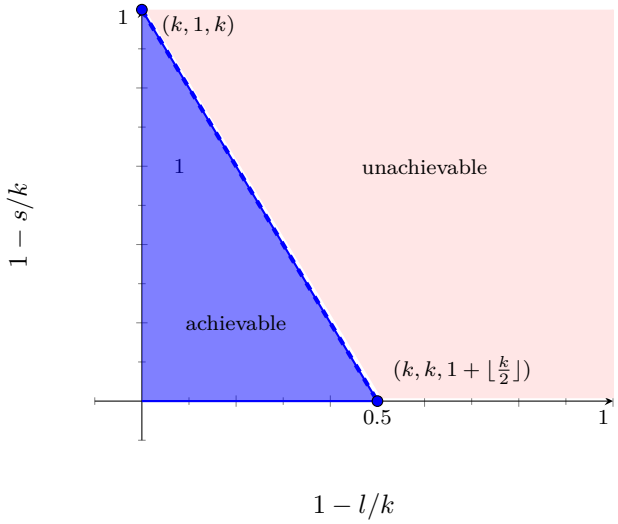

Figure 1 Region of safety-liveness guarantee. The integer grids in the blue area consists of all points which are achievable, while the integer points in the red area are not achievable under partial synchrony. We highlight the two extreme achievable tuples $(k, 1, k)$ and $(k, k, 1 + \lfloor \frac{k}{2} \rfloor)$.

2. We show that by repeatedly composing the $(2, 1, 2)$ and $(3, 3, 2)$ solutions, we can build overlay blockchains that achieve any tuple in Theorem 1.

As an inspiration to our approach, we can draw an analogy to switching circuit design in Claude E. Shannon's masters' thesis [39] (Table 1). In this spectacular masters' thesis, Shannon used serial and parallel composition of switches to create an OR and an AND gate respectively, and then use these gates as building blocks to create more complex circuits which can be designed using Boolean algebra. Drawing the analogy, the timestamping solution to $(2, 1, 2)$ can be viewed as a *serial composition* of two blockchains, and the OFT solution to $(3, 3, 2)$, called the triangular composition due to the use of three blockchains, can be viewed as a *parallel composition* of three blockchains for partial synchrony (Curiously, unlike switching circuits, no parallel composition of 2 blockchains can exist *under partial synchrony*, as ruled out by Theorem 2. See Section 4.2 for more discussion.)

Our serial and parallel compositions require the composed underlay blockchains to satisfy certain properties (*e.g.*, hosting smart contracts) outlined in Section 4. These properties are satisfied by blockchains that support general-purpose smart contracts (*e.g.*, EVM in Ethereum) and run on PBFT-style consensus protocols [16] such as Tendermint [12]. In this context, our circuit compositions can be readily implemented by Cosmos blockchains [1] that support CosmWasm smart contracts and run Tendermint as their consensus protocols.

1.5 Outline

Our paper is organized as follows. Related works are summarized in Section 2. We present preliminary definitions in Section 3. We describe the serial composition for achieving the tuple $(2, 1, 2)$ and the triangular composition for achieving $(3, 3, 2)$ in Section 4. Using them as *gates*, we build circuit compositions achieving all possible security properties under partial synchrony and synchrony in Sections 5 and 7. The converse results for unachievable properties under partial synchrony and synchrony are in Sections 6 and 8. Section 9 investigates scalability of large circuits based on serial and triangular compositions.

	Switching Circuits	Blockchain Circuits
Goal	Computation	Security
Basic components	switches	blockchains
Component state	$X \in \{0,1\}$ $X = 1$ iff switch is open	$X = (S, L) \in \{0,1\}^2$ $S = 1$ iff chain is safe $L = 1$ iff chain is live
Serial composition	$X_1 + X_2 = X_1$ OR X_2	$X_1 + X_2 = (S_1 + S_2, L_1 L_2)$
Parallel composition	$X_1 X_2 = X_1$ AND X_2	$X_1 X_2 X_3 = (S_1 S_2 S_3, L_1 L_2 + L_2 L_3 + L_3 L_1)$
Syntheis	Boolean formulas	generalized quorum systems
Completeness	All truth table assignments	All achievable compositions

Table 1 Comparison between switching circuits and blockchain circuits. We note that the parallel composition for blockchain circuits is more complicated than $X_1 X_2 = (S_1 S_2, L_1 + L_2)$, which would have been the natural analogue of the parallel composition. However, such a composition is impossible to achieve (Section 4.2).

2 Related Works

Timestamping. A timestamping protocol allows a *consumer* chain to obtain timestamps for its blocks by checkpointing [26, 21, 23, 43, 42] them on a *provider* chain; so that in case there is a fork in the consumer chain, the fork can be resolved by choosing the one with the earlier timestamp (other uses of timestamping include reducing the latency of Nakamoto consensus [20]). The provider chain is thus used as a timestamping server that provides security to the consumer chain. Examples of timestamping protocols include Polygon [7] checkpointing onto Ethereum, Stacks [8] and Pikachu [10] checkpointing to PoW Ethereum and Babylon [43] checkpointing to Bitcoin. Authors of [42] design an interchain timestamping protocol to achieve *mesh security* [5, 6], in which Cosmos zones provide and consume security to and from each other in a mesh architecture. The protocol strongly favors safety over liveness and cannot achieve all possible security properties.

Trustboost. Trustboost [41] proposes a family of protocols where multiple constituent blockchains interact to produce a combined ledger with boosted trust. Each blockchain runs a smart contract that emulates a validator of an overlay consensus protocol, Information Theoretic HotStuff (IT-HS) [9], that outputs the ledger with boosted trust. As long as over two-thirds of the blockchains are secure (safe and live), Trustboost satisfies security; thus its security guarantees are implied by our circuit construction. Trustboost is implemented using Cosmos zones as the underlay blockchains and the inter-blockchain communication protocol (IBC) as the method of cross-chain communication; so that the emulated validators can exchange messages. In this paper, we separate the safety/liveness conditions of the component blockchains for achieving safety/liveness guarantees of the interchain circuit construction. Trustboost does not make any claims when the number of chains $k \leq 3$ or when a chain loses just one of its security properties (either safety or liveness), while our blockchain circuit approach covers all possible choices of achievable (k, s, l) tuples, especially the two basic cases $(2, 1, 2)$ and $(3, 3, 2)$. Trustboost also relies on external bots/scripts to notify the constituent blockchains about the overlay protocol's timeouts, whereas our approach does not use any external parties beyond the underlay blockchains.

As one can trade-off the safety and liveness resilience of HotStuff by tuning its quorum size, a natural question is if a similar trade-off for (k, s, l) points can be achieved for Trustboost by tuning the quorum size of its overlay protocol (IT-HS). However, to achieve these points,

the overlay protocol must ensure liveness as long as l blockchains are live, without requiring their safety. This necessitates changing the overlay protocol to prioritize liveness in the case of safety violations[2]. Then, a new security analysis is needed for the modified overlay protocol so that Trustboost can continue to leverage its security. In contrast, the triangular composition of our circuits builds on a liveness-favoring protocol as is (*cf.* Section 4.2).

Cross-staking. Cross-staking was proposed as a technique to enhance the security of the Cosmos blockchains (zones) in the context of mesh security. A consumer zone allows validators of a provider zone to stake their tokens on the consumer zone via IBC and validate the consumer chain. However, this requires validators to run full nodes of multiple blockchains, thus resulting in a large overhead on that of interchain protocols and our blockchain circuit approach, where the validators of the constituent blockchains only run light clients.

Thunderella. Thunderella [37] is a SMR consensus protocol, composed of an asynchronous, quorum-based protocol and a synchronous, longest chain based protocol. The synchronous protocol ensures that Thunderella satisfies security, albeit with latency $O(\Delta)$, at all times with 1/2 resilience under the Δ-synchronous sleepy network model [36], whereas the asynchronous path helps achieve fast progress with latency dependent only on the actual network delay δ, if over 3/4 of the validators are honest and awake. Thus, its goal is to support different latency regimes under different assumptions by having the validators execute two protocols, rather than to improve security by combining different chains in a black-box manner (*cf.* interchain consensus protocols, Section 3).

Robust Combines. Our approach of combining existing underlay chains to design a more secure overlay protocol is conceptually related to cryptographic combiners [24, 22], which combine many instances of a cryptographic primitive to obtain a more secure candidate for the same primitive. The output satisfies correctness and security, if these properties are guaranteed for at least one of the original candidates. In contrast, our circuit composition decouples safety and liveness and analyzes the dependence of the overlay protocols' safety and liveness *separately* on the same properties of the underlay chains.

3 Preliminaries

In this section, we introduce several preliminary definitions. We use $[k]$ to represent the set $\{1, 2, \ldots, k\}$. We denote the elements within a sequence s of k non-negative integers by the indices $i \in [k]$: $s = (s_1, \ldots, s_k)$. For two such sequences, we write $s \leq s'$ if for all $i \in [k]$, $s_i \leq s'_i$. Similarly, $s < s'$ if $s \leq s'$ and there exists an index $i^* \in [k]$ such that $s_{i^*} < s'_{i^*}$. We denote a permutation function on the sequences s by σ. There are two types of participants in our model: validators and clients.

Validators and Clients. Validators take as input transactions from the environment $\mathcal{Z}$ and execute a blockchain protocol (also known as total order broadcast). Their goal is to ensure that the clients output a single sequence of transactions. Validators output consensus messages (*e.g.* blocks, votes), and upon query, send these messages to the clients. After receiving consensus messages from sufficiently many validators, each client individually outputs a sequence of *finalized* transactions, called the ledger and denoted by L. Clients can be thought of as external observers of the protocol, which can go online or offline at will.

[2] For instance, HotStuff must relax the liveness rule of the SafeNode function to return true as long as the view number of the prepareQC is larger than or *equal to* the locked view, which is different from the current specification in [46].

Blocks and Chains. Transactions are batched into blocks and the blockchain protocol orders these blocks instead of ordering transactions individually. Each block B_k at height k has the following structure: $B_k = (x_k, h_k)$, where x_k is the transaction data and $h_k = H(B_{k-1})$ is a parent pointer (*e.g.*, a hash) to another block. There is a genesis block $B_0 = (\perp, \perp)$ that is common knowledge. We say that B extends B', denoted by $B' \preceq B$, if B' is the same as B, or can be reached from B by following the parent pointers. Each block that extends the genesis block defines a unique chain. Two blocks B and B' (or the chains they define) are *consistent* if either $B \preceq B'$ or $B' \preceq B$. Consistency is a transitive relation.

A client cl *finalizes* a block B at some time t if it outputs B and the chain of blocks preceding B as its ledger at time t, *i.e.*, if cl's latest chain contains B for the first time at time t. The ledger in cl's view is determined by the order of the transactions in this chain.

A blockchain protocol is said to *proceed in epochs of fixed duration* if whenever the protocol is live, a new block is confirmed in the view of any client at a rate of at most one block every T seconds for some constant T, *i.e.*, the protocol has bounded chain growth rate. Such examples include Tendermint [12] and Streamlet [17], where a new block is proposed by an epoch leader every 2Δ time, where Δ is a protocol parameter. These protocols enable the clients to track time by inspecting the timestamps on the blocks. PBFT-style protocols such as PBFT [16] and HotStuff [46] can also be made to proceed in epochs of fixed duration (despite not being so) by artificially introducing delays before the proposal are broadcast.

Adversary. We consider a computationally-bounded adversary $\mathcal{A}$ that can corrupt a fraction of the validators called *adversarial*. The remaining ones that follow the protocol are called *honest*. Adversary controls message delivery subject to the network delay.

Networking. In a *partially synchronous* network [18], the adversary can delay messages arbitrarily until a global stablizaton time (GST) chosen by the adversary. After GST, the network becomes synchronous and the adversary must deliver messages sent by an honest validators to its recipients within Δ time, where $\Delta > 0$ is a known delay bound[3]. The network is called *synchronous* if GST is known and equal to zero.

Security. Let L_t^{cl} denote the ledger output by a client cl at time t. We say that a protocol is *safe* if for any times t, t' and clients $\mathsf{cl}, \mathsf{cl}'$, L_t^{cl} and $L_{t'}^{\mathsf{cl}'}$ are consistent, and for any client cl, $L_t^{\mathsf{cl}} \preceq L_{t'}^{\mathsf{cl}}$ for all $t' \geq t$. We say that a protocol is *live* if there is a time $t_{\text{fin}} > 0$ such that for any transaction tx input to all honest validator at some time t, it holds that $\text{tx} \in L_{t'}^{\mathsf{cl}}$ for any client cl and times $t' \geq \max(\text{GST}, t) + t_{\text{fin}}$. Note that a protocol satisfying liveness also ensures that clients keep outputting valid transactions; because clients refusing to output invalid transactions as part of their ledgers will not output *anything* after the first invalid transaction. When we talk about the ledger of a specific protocol Π_A output by a client cl at time t, we will use the notation $L_{A,t}^{\mathsf{cl}}$.

Certificates. We adopt the definition of certificates from [30].

▶ **Definition 4** (Definition 3.2 of [30]). *We say that a blockchain protocol with confirmation rule $C(.)$* generates certificates *if the following holds with probability $> 1 - \epsilon$ when the protocol is run with security parameter ϵ, under the conditions for which safety is satisfied: There do not exist conflicting ledgers L_1 and L_2, a time t and sets of consensus messages M_1 and M_2 broadcast by time t, such that L_i is a prefix of the confirmed ledger determined by $C(.)$ on M_i,* i.e., *$C(M_i)$ for $i \in \{1, 2\}$.*

[3] We assume synchronized clocks as bounded clock offset can be captured by the delay Δ, and clocks can be synchronized using the process in [18].

An example of a safety condition is over 2/3 of the validators being honest (*e.g.*, for PBFT [16]), whereas a confirmation rule example, applied to the consensus messages, is confirming a block if there are commit messages for it from over 2/3 of the validators. In a certificate-generating protocol, any client cl that finalizes a ledger L can convince any other client to finalize L by showing a subset of the consensus messages. These messages form a *certificate* for L.

All protocols that are safe under partial synchrony generate certificates [30]. For example, in PBFT-style protocols [16], Tendermint [12], HotStuff [46] and Streamlet [17], clients finalize a block upon observing a quorum of commit messages from over 2/3 of the validator set. This quorum of commit messages on the block acts as a certificate for the block. When these protocols are safe, there cannot be two quorums, *i.e.*, certificates, attesting to the finality of conflicting blocks. In contrast, Nakamoto consensus [32] does not generate certificates. As clients confirm a chain *only if* they do not receive a longer chain, no set of messages by themselves suffice to convince clients of the confirmation of a blockchain, as there might always exist a longer but hidden chain of blocks.

Interchain Consensus Protocols. An interchain consensus protocol (interchain protocol for short) is a blockchain protocol, called the *overlay* protocol, executed on top of existing blockchain protocols, called the underlay chains. Its participants are the clients and validators of the constituent underlay chains Π_i, $i \in [n]$. All clients and validators observe all underlay chains, but each validator is responsible for participating in the execution of one of these blockchain protocols, which ensures scalability. Clients and honest validators of each underlay chain run a client of every other chain, and can read from and write to the output ledgers of the other chains. This restricted communication is captured by the notion of *cross-chain communication (CCC)* [47, 41]: each underlay chain Π_i, $i \in [n]$, only exposes read and write functionalities to its finalized ledger. Clients and validators of every other chain, Π_j, $j \neq i$, verify the finality of Π_i's ledger via certificates (*e.g.*, by verifying a quorum of signatures on the finalized blocks in PBFT-style protocols [16]), whereas the internal mechanisms and the validator set of Π_i remain hidden from the interchain protocol, except as used by the CCC to validate certificates. They write to Π_i by broadcasting their transactions to all Π_i validators as input in the presence of a public-key infrastructure, or by using trustless relays that can produce a proof of transmission by collecting replies from sufficiently many Π_i validators. Clients of the interchain protocols use only their views of the finalized ledgers of the underlay chains to determine the overlay blockchain's ledger.

The CCC functionality can be implemented by a trusted controller that relays data across chains, or by committees subsampled from among the validators. A prominent CCC example is the Inter-Blockchain Communication protocol (IBC) of Cosmos [2], where the messages are transmitted by relayers [4] akin to controllers. However, IBC does not require the relayers to be trusted for safety, as it allows the receiver chain's validators to verify messages by inspecting if they were included in the finalized sender chain blocks.

4 Protocol Primitives

We build the overlay protocols by composing simpler protocols in two different ways: serial composition and triangular composition. In this section, we describe these compositions and their implications for security.

4.1 Serial Composition

We describe the safety-favoring serial composition Π_s with two constituent certificate-generating blockchain protocols, Π_A and Π_B (Fig. 2; *cf.* Alg. 1). The Π_A validators receive

Algorithm 1 The algorithm used by a bootstrapping client cl to output the ledger L_s of the serial composition Π_s instantiated with two constituent blockchains Π_A and Π_B at some time t. The algorithm takes $L^{\mathsf{cl}}_{B,t}$, the finalized Π_B ledger output by cl at time t, as its input and outputs the ledger L_s. The function GetSnapshots returns the snapshots of the Π_A ledger included in $L^{\mathsf{cl}}_{B,t}$ along with their certificates. The function isCertified returns true if the input ledger is accompanied by a valid certificate.

```
1:  function OutputChain(L^cl_{B,t})
2:      snp_1, ..., snp_m ← GetSnapshots(L^cl_{B,t})
3:      L_s ← ⊥
4:      for i = 1, ..., m do
5:          if isCertified(snp_i) then
6:              L_s ← Clean(L_s, snp_i)
7:          end if
8:      end for
9:      return L_s
10: end function
```

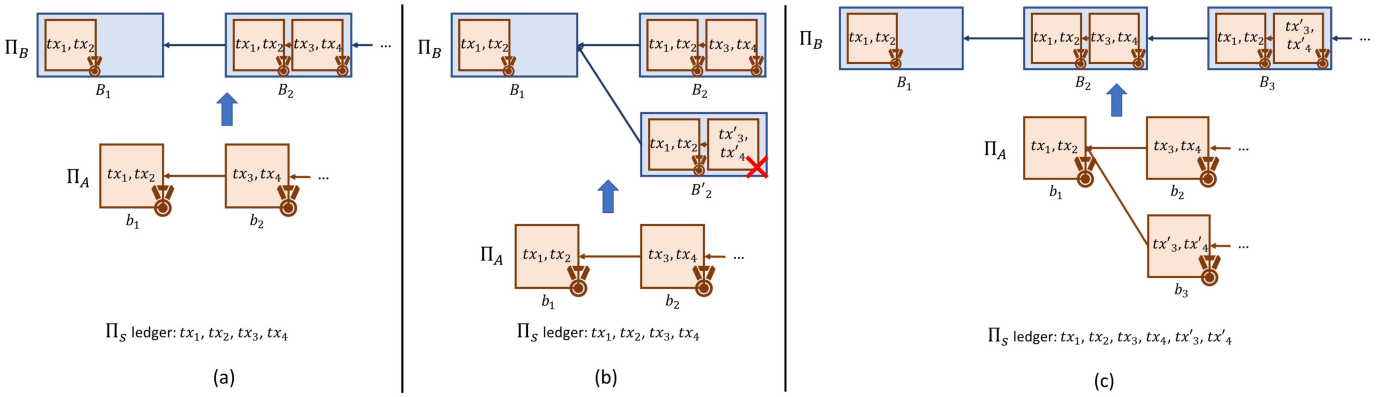

Figure 2 Serial composition. The Π_A blocks (brown) are denoted by $b_1, b_2, \ldots$ and the Π_B blocks (blue) are denoted by $B_1, B_2, \ldots$. Certificates of the Π_A blocks are denoted by the medals. **In (a)**, both Π_A and Π_B are safe. Thus, every client observes the same Π_B ledger with certified snapshots $\mathsf{snp}_1 = (\mathrm{tx}_1, \mathrm{tx}_2)$ and $\mathsf{snp}_2 = (\mathrm{tx}_1, \mathrm{tx}_2, \mathrm{tx}_3, \mathrm{tx}_4)$. Upon sanitizing the snapshots, clients obtain $\textsc{Clean}(\mathsf{snp}_1, \mathsf{snp}_2) = (\mathrm{tx}_1, \mathrm{tx}_2, \mathrm{tx}_3, \mathrm{tx}_4)$ as the Π_s ledger. **In (b)**, the Π_B ledger is not safe, and two clients x and y observe conflicting Π_B ledgers L^x_{B,t_1} and L^y_{B,t_2} with blocks B_1, B_2 and B_1, B'_2 respectively. The blocks B_1, B_2 and B'_2 contain the certified snapshots $\mathsf{snp}_1 = (\mathrm{tx}_1, \mathrm{tx}_2)$, $\mathsf{snp}_2 = (\mathrm{tx}_1, \mathrm{tx}_2, \mathrm{tx}_3, \mathrm{tx}_4)$ and $\mathsf{snp}'_2 = (\mathrm{tx}_1, \mathrm{tx}_2)$ respectively. Note that $(\mathrm{tx}'_3, \mathrm{tx}'_4)$ is not part of the certified snapshot snp'_2 as they are not included in a certified Π_A block. Upon sanitizing the snapshots, clients again obtain consistent Π_s ledgers $L^x_{B,t_1} = \textsc{Clean}(\mathsf{snp}_1, \mathsf{snp}_2) = (\mathrm{tx}_1, \mathrm{tx}_2, \mathrm{tx}_3, \mathrm{tx}_4)$ and $L^y_{B,t_2} = \textsc{Clean}(\mathsf{snp}_1, \mathsf{snp}'_2) = (\mathrm{tx}_1, \mathrm{tx}_2)$. **In (c)**, the Π_A ledger is not safe, and two clients x and y observe conflicting Π_A ledgers L^x_{A,t_1} and L^y_{A,t_2} with blocks b_1, b_2 and b_1, b_3 respectively. However, both clients observe the same Π_B ledger with blocks B_1, B_2, B_3 and their certified snapshots $\mathsf{snp}_1, \mathsf{snp}_2$, snp_3. Hence, upon sanitizing the snapshots, clients obtain the same (consistent) Π_s ledgers $L^x_{s,t_1} = L^y_{s,t_2} = \textsc{Clean}(\mathsf{snp}_1, \mathsf{snp}_2, \mathsf{snp}_3) = (\mathrm{tx}_1, \mathrm{tx}_2, \mathrm{tx}_3, \mathrm{tx}_4, \mathrm{tx}'_3, \mathrm{tx}'_4)$.

transactions from the environment and other validators, and the clients of Π_A output a certified Π_A ledger. Each Π_B validator acts as a client of Π_A, and consider the Π_A ledger

in its view, called a *snapshot*, and its certificate, as a *transaction input* to Π_B[4] (Fig. 2a). At any time step t, each client cl of the serial composition (which is a client of both Π_A and Π_B), online at time t, inspects the certified snapshots of the Π_A ledger within its Π_B ledger. Then, cl reads the certified Π_A snapshots in the order they appear in its Π_B ledger, copies these snapshots and finally eliminates the duplicate transactions appearing in multiple snapshots by calling a sanitization function. The sanitization function $\textsc{Clean}(L_A, L_B)$ takes two ledgers L_A and L_B, concatenates them, eliminates the duplicate transactions that appear in L_B and keeps their first occurrence in L_A (cf. [34] [42]). Finally, the client outputs the remaining transactions as *its* Π_s ledger (its view of the Π_s ledger at that time). The serial composition satisfies the following security properties:

▶ **Theorem 5.** *Consider the serial composition Π_s instantiated with the certificate-generating blockchain protocols Π_A and Π_B. Then, under partial synchrony,*

1. *Π_s satisfies safety if at least* one *of Π_A or Π_B is safe.*
2. *Π_s satisfies liveness with constant latency after GST if* both *Π_A and Π_B are live with constant latency after GST.*
3. *Π_s generates certificates.*
4. *Π_s proceeds in epochs of fixed duration if Π_A and Π_B proceed in epochs of fixed duration.*

Proof of Theorem 5 is given in Appendix F.1. Proof of the statements 1 and 2 are illustrated by Fig. 2 that covers the cases when Π_B and Π_A are not safe, yet Π_s is safe. Statements 3 and 4 are needed for further composability of the serial composition with other serial and triangular compositions (*cf.* the conditions on Theorems 5 and 6). Appendix B describes an attack against the serial composition when Π_A is not certificate-generating.

For the serial composition Π_s, we require liveness only for the transactions input to all honest Π_A validators. In general, liveness must be guaranteed only for the transactions input to *all* honest validators of the underlay protocols. If validators have access to a public-key infrastructure that identifies each other, then any transaction input to a single honest validator of an underlay protocol can be broadcast to all validators of all underlay protocols, and thus can be included in the ledgers.

4.2 Triangular Composition

A natural liveness-favoring analogue of the serial composition of two blockchains would be a composition that ensures liveness if either of the two chains is live, and safety if both chains are safe. However, no interchain protocol can satisfy these guarantees, even under synchrony. Below, we provide the intuition behind this result (*cf.* Theorems 18 and 19 for details).

Consider two blockchains Π_A and Π_B that are not live, but safe. Here, safety of a protocol (*e.g.*, Π_A) means that different clients' views of the Π_A ledger are consistent, yet it is possible that the Π_A ledger output by a client conflicts with a Π_B ledger output by another client. The protocol Π_A emulates the behavior of a live blockchain towards a client cl_1, whereas it is stalled in cl_2's view, *i.e.*, $L_{A,t}^{\mathsf{cl}_2} = \emptyset$ for all times t. In the meanwhile, Π_B emulates the behavior of a live blockchain towards a different client cl_2, whereas it is stalled in cl_1's view, *i.e.*, $L_{B,t}^{\mathsf{cl}_1} = \emptyset$ for all times t. Since the triangular composition is conjectured to be live when either of the blockchains is live, both cl_1 and cl_2 output transactions based on their observations of the Π_A and Π_B ledgers respectively (as far as cl_1 is concerned, Π_A *looks* live,

[4] For instance, if Π_B is a blockchain protocol, the snapshots and their certificates will be included in the blocks by the block proposers (*cf.* Section 9 for more efficient implementations).

and as far as cl_2 is concerned, Π_B *looks* live). However, when these Π_A and Π_B ledgers are different and conflicting, this implies a safety violation even though both Π_A and Π_B are safe, *i.e.*, cl_1 and cl_2's Π_A (Π_B) ledgers are consistent (as $\emptyset$ is a prefix of every ledger).

Given the example above which shows the impossibility of a composition that is live if either chain is live and safe if both are safe, we relax the properties expected of a liveness-favoring composition in two ways: (i) the *triangular* composition of 3 blockchains ensures liveness if 2 of the 3 constituent chains are live, and safety if all chains are safe, under partial synchrony (Theorem 6), whereas (ii) the *parallel* composition of 2 blockchains, *under synchrony*, ensures liveness if either of the constituent chains is live, and safety if *both* chains are safe and live (Section 7.1, Theorem 15). Here, we focus on the triangular composition.

For inspiration towards a minimal triangular composition with these guarantees, we consider a setting, where the protocol participants are validators rather than blockchains. We observe that a natural analogue of a blockchain that is not live, but safe, is a validator with *omission faults*. Since the triangular composition for blockchains requires the safety of all constituent protocols for safety, its analogue for validators would tolerate only omission faults. Thus, our triangular composition is motivated by omission fault tolerant (OFT) consensus protocols [28, 13, 35]. Before presenting the composition, we briefly describe these OFT protocols for validators, which we extend to the blockchain setting.

4.2.1 The OFT Protocol for Validators.

The OFT protocol is a leader-based blockchain protocol that generates certificates under a partially synchronous network. It is run by 3 validators mirroring the most basic triangular composition. It proceeds in epochs of fixed duration 3Δ. In a nutshell, it works as follows:

Each epoch v has a unique leader that proposes a block at the beginning of the epoch , *i.e.*, at time $3\Delta v$. Upon observing a proposal for epoch v, validators broadcast acknowledge messages for the proposed block at time $3\Delta v + \Delta$. Upon observing a *certificate* of 2 unique acknowledge messages from epoch v for the epoch's proposal, validators and clients finalize the proposed block and its prefix chain. If a validator does not observe a certificate of 2 acknowledge messages for an epoch v proposal by time $3\Delta v + 2\Delta$, it broadcasts a leader-down message for epoch v, where the message contains the block with the highest epoch number among the ones it previously voted for. Leader-down messages enable the leader of the next epoch to identify the correct block to propose on to preserve safety. A detailed protocol description is presented in Appendix D.

4.2.2 From OFT Protocol to the Triangular Composition.

We next describe a triangular composition for 3 blockchains. It consists of 3 underlay blockchain protocols, Π_A, Π_B and Π_C, run by validators and an overlay protocol, *i.e.*, the OFT protocol, run on top of these chains (Fig. 3). Each underlay protocol executes a smart contract that *emulates a validator* of the overlay OFT protocol (*cf.* Appendix A for a discussion on validator emulation). These emulated validators exchange messages via the CCC abstraction. There is a PKI that identifies on each underlay chain the 2 other chains emulating a validator (*e.g.*, by means of the public keys of the other chains' validators).

Blockchains. The triangular composition requires the underlay protocols to run general-purpose smart contracts and to proceed in epochs of fixed duration T. This is because the overlay OFT protocol requires each emulated validator to keep track of the time passed since it entered any given epoch. In general, it is impossible to emulate the validators of any overlay protocol secure under partial synchrony, if the underlay protocol has no means

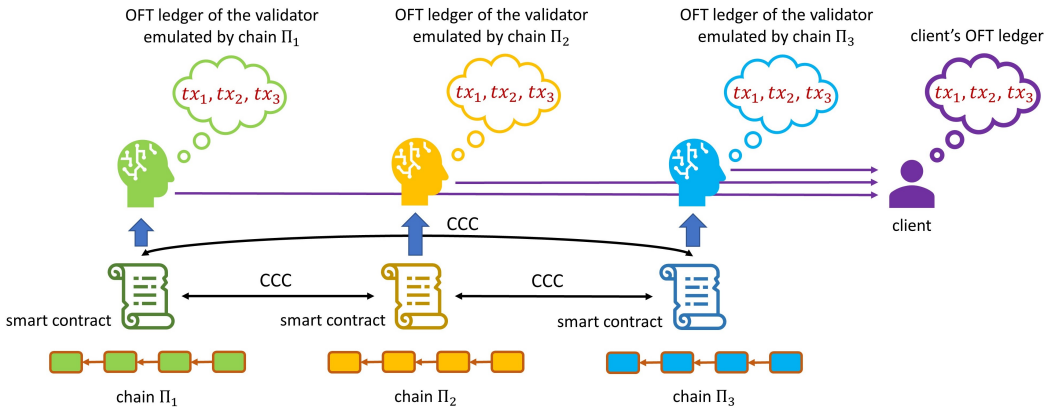

Figure 3 Triangular composition. An overlay OFT protocol run on top of 3 underlay blockchains. A smart contract on each of the underlays emulates a validator of the OFT protocol and outputs a finalized OFT ledger. The client reads the underlay chains' ledgers and outputs the OFT ledger finalized by a majority of the emulated validators.

of keeping track of the real time (*cf.* Appendix C). We achieve this functionality by using underlay protocols that proceed in epochs of fixed time duration such as Tendermint [12] or Streamlet [17] (*cf.* Appendix C). However, our triangular composition can also be instantiated with optimistically responsive protocols (*cf.* Section 10 for more discussion).

Using epoch numbers recorded in the underlay blocks, the smart contract tracks the time passed since it entered any given epoch of the overlay protocol. If it entered some epoch v of the overlay protocol at time t, it moves to epoch $v+1$ at an underlay block of an underlay epoch $3t_{\text{fin}}/T$, where t_{fin} is the cross-chain communication latency. Here, the 3Δ epoch of the overlay OFT protocol is replaced by a $3t_{\text{fin}}$ length epoch, since the messages exchanged by the *emulated* validators incur additional latency, including the finalization latency of the underlay chains besides network delay.

Clients. We next describe how clients of the triangular composition output a ledger for the overlay protocol using the ledgers of the underlay chains. Upon outputting a ledger L_A, L_B and L_C for each underlay protocol Π_A, Π_B and Π_C, at some time t, a client inspects the execution of the smart contracts as attested by these ledgers. If the execution trace on some ledger is invalid according to the rules of the smart contract, then the client discards the parts of the ledger starting with the first invalid transaction recorded on it, thus turning invalid execution into a liveness failure. For instance, sending a syntactically incorrect message is detectable by only inspecting the messages on a ledger. In contrast, sending two acknowledge messages for conflicting overlay blocks in the same overlay epoch might not be detected upon inspection, since these two messages can exist in separate execution traces emulating the same OFT validator, *i.e.*, on conflicting ledgers, observed by different clients (safety failure).

Once the client observes the execution traces for the validators emulated on valid portions of the ledgers L_A, L_B and L_C, it identifies the blocks of the overlay protocol committed by each emulated validator. It accepts and outputs an overlay block and its prefix chain if it was committed by 2 or more emulated validators (as attested by the ledgers of 2 or more underlay chains). If a client accepts and outputs an overlay block and its chain of height h, it never outputs a shorter overlay chain from that point on. If the client observes multiple conflicting L_A, L_B and L_C ledgers when the safety of underlay chains is violated, it considers all these ledgers, and might output conflicting overlay blocks as a result. However, this is

not a problem, as the proof of the next theorem shows that the client will continue to output blocks and retain liveness nevertheless.

The triangular composition satisfies the following:

▶ **Theorem 6.** *Consider the triangular composition Π_t instantiated with the protocols Π_A, Π_B and Π_C, that proceed in epochs of fixed duration. Then, under partial synchrony,*

1. *Π_t satisfies safety if all of Π_A, Π_B and Π_C are safe.*
2. *Π_t satisfies liveness with constant latency after GST if 2 blockchains among Π_A, Π_B and Π_C are live after GST with constant latency and proceed in epochs of fixed duration.*
3. *Π_t generates certificates if Π_A, Π_B and Π_C do so.*
4. *Π_t proceeds in epochs of fixed duration.*

Proof of Theorem 6 is given in Appendix F.2. Statements 1 and 2 are based on the proof of the original OFT protocol design for validators. Statements 3 and 4 are needed for further composability of the triangular composition with other serial and triangular composition (*cf.* the conditions on Theorem 5).

5 Circuits for Partial Synchrony

In this section, we construct overlay protocols via circuit composition achieving the security properties claimed by Theorem 1 and show optimality by proving Theorem 2. We also extend these results to all possible overlay protocols, akin to the generalization of security properties to quorum and fail-prone systems. Unlike the security claims for the protocol primitives, all of the proofs below are algebraic in nature.

5.1 Extended Serial and Triangular Constructions

We first build extended serial and triangular constructions as building block toward the full circuit composition.

▶ **Lemma 7.** *Let Π_i, $i \in [k]$ be k different blockchain protocols that generate certificates. Then, there exists a protocol, called the n-serial composition, satisfying the following properties:*

- *it is safe if at least one of Π_i, $i \in [k]$ is safe.*
- *it is live after GST with constant latency if all of Π_i, $i \in [k]$ are live after GST with constant latency.*
- *it generates certificates.*
- *it proceeds in epochs of fixed duration if all of Π_i, $i \in [k]$ do so.*

Lemma 7 follows directly from iteratively applying Theorem 5 on the protocols Π_i, $i \in [k]$.

▶ **Lemma 8.** *For any integer $f \geq 1$, let Π_i, $i \in [2f+1]$, be $2f+1$ different blockchain protocols that proceed in epochs of fixed duration. Then, there exists a protocol, called the $(2f+1)$-triangular composition, satisfying the following properties:*

- *it is safe if all Π_i, $i \in [2f+1]$ are safe.*
- *it is live after GST with some constant latency if at least $f+1$ of Π_i, $i \in [2f+1]$ are live after GST with some constant latency.*
- *it generates certificates if all Π_i, $i \in [2f+1]$ generate certificates.*
- *it proceeds in epochs of fixed duration.*

Note that the original triangular composition with three protocols (referred to as the triangular composition) would be called a 3-triangular composition. Proof of Lemma 8 is presented in Appendix F.6. It constructs an f-triangular composition via strong induction on the number f. The inductive step uses the $(f-1)$-triangular composition and the serial composition of Lemma 7, whereas the base case follows from the properties of the 3-triangular composition shown by Theorem 6.

5.2 Permutation Invariant Circuits for Partial Synchrony

We next prove Theorem 1, which characterizes the security of so-called permutation invariant overlay protocols. This class of protocols achieves safety (or liveness) as long as *any* subset of the underlay chains with a sufficient size provide the same set of security guarantees (*e.g.*, any subset of 5 out of 7 underlay chains is all safe and/or all live). In this sense, these protocols do not distinguish between the underlay chains.

Proof of Theorem 1 relies on Theorems 5 and 6. Recall that a tuple (k, s, l) was defined to be achievable if there exists an interchain protocol with k blockchains such that if at least s blockchains are safe, the protocol is safe, and if at least l blockchains are live, the protocol is live. By definition, a serial composition achieves the $(2, 1, 2)$ point (Theorem 5), and a triangular composition achieves the $(3, 3, 2)$ point (Theorem 6). Similarly, a $(2f+1)$-triangular composition achieves the $(2f+1, 2f+1, f+1)$ point, and there exist such compositions for any $f \geq 1$ by Lemma 8, which itself follows from Theorems 5 and 6.

For two blockchain protocols Π_A, Π_B, we denote by $\Pi_A \oplus \Pi_B$ the serial compositions of these two blockchains as described in Section 4.1. Consider k protocols $\Pi_1, \Pi_2, \ldots, \Pi_k$. We iteratively define $\oplus_{i=1}^{j+1}\Pi_i = \left(\oplus_{i=1}^{j}\Pi_i\right) \oplus \Pi_{j+1}$ for $j \in [k-1]$, and denote a protocol achieving the $(k, 1, k)$ point by $\pi^{(k,1,k)}(\Pi_1, \ldots, \Pi_k)$. and denote a protocol achieving the $(2f+1, 2f+1, f+1)$ point by $\pi^{(2f+1,2f+1,f+1)}(\Pi_1, \ldots, \Pi_{2f+1})$.

Towards the final result, the following lemma shows that we can construct a protocol achieving $(k+m, s, l+m)$ point using a protocol achieving (k, s, l).

▶ **Lemma 9.** *For any integer $m \geq 1$, if (k, s, l) is achievable using $\pi^{(k,s,l)}$ then, $(k+m, s, l+m)$ is achievable using*

$$\pi^{(k+m,s,l+m)}(\{\Pi_i\}_{i=1}^{k+m}) = \bigoplus_{\substack{S\subseteq[k+m]\\|S|=k}} \pi^{(k,s,l)}\left(\{\Pi_j\}_{j\in S}\right). \tag{1}$$

Proof. We first show that $\pi^{(k+m,s,l+m)}$ defined in (1) is safe if at least s of the blockchains are safe. There exists a subset $S_0 \subseteq [k+m]$ with $|S_0| = k$ such that at least s blockchains among $\{\Pi_j\}_{j\in S_0}$ are safe. This implies that $\pi^{(k,s,l)}\left(\{\Pi_j\}_{j\in S_0}\right)$ is safe. As we enumerate all subsets with size k in constructing $\pi^{(k,s,l)}\left(\{\Pi_j\}_{j\in S_0}\right)$, by Lemma 7, we observe that $\pi^{(k+m,s,l+m)}(\{\Pi_i\}_{i=1}^{k+m})$ is safe; as one of the blockchains in the serial composition, namely $\pi^{(k,s,l)}\left(\{\Pi_j\}_{j\in S_0}\right)$, is safe.

On the other hand, suppose that at least $l+m$ of the blockchains are live, which implies that at most $k-l$ blockchains are not live. Therefore, for any arbitrary choice of size-k subset $\{\Pi_j\}_{j\in S}$ from $\{\Pi_j\}_{j=1}^{k+m}$, at most $k-l$ blockchains are not live, or equivalently, at least l blockchains in $\{\Pi_j\}_{j\in S}$ are live. This implies that $\pi^{(k,s,l)}\left(\{\Pi_j\}_{j\in S}\right)$ is live for all possible choices of subset S with size k. Then, by Lemma 7, we observe that $\pi^{(k+m,s,l+m)}(\{\Pi_i\}_{i=1}^{k+m})$ is live; as all of the blockchains in the serial composition are live. ◀

Finally, we present the proof of Theorem 1.

Proof of Theorem 1. By Lemma 8, there are circuit compositions achieving the $(2f+1, 2f+1, f+1)$ point given copies of *any* two protocols achieving the $(2,1,2)$ and $(3,3,2)$ points respectively, via recursive compositions of these protocols. This in turn implies that for any given integers k, s, l such that $\lfloor k/2 \rfloor + 1 \leq l \leq n$ and $s = 2(k-l)+1$ (boundary of the achievable points on Fig. 1), the point $(s, s, k-l+1) = (2(k-l)+1, 2(k-l)+1, k-l+1)$ is achievable. Since $s = 2(k-l)+1$ for these boundary points, we have $l+s-k = k-l+1$. Therefore, by Lemma 9, $(k, s, l) = (s+(k-s), s, k-l+1+(k-s))$ is achievable. This implies that all points (k, s, l) such that $\lfloor k/2 \rfloor + 1 \leq l \leq n$ and $s \geq 2(k-l)+1$ are achievable. ◀

5.3 General Circuits for Partial Synchrony

We next present a general characterization of the security of the overlay protocol under partial synchrony as a function of the safety and liveness of the underlay chains. Note that a general characterization would include protocols that are not permutation invariant, *i.e.*, providing different security guarantees when two subsets of the underlay chains achieve the same set of security properties. For example, a non-permutation invariant overlay protocol with three underlay chains Π_i, $i \in [3]$, might be live when the underlay chains Π_1 and Π_2 are both live, but it might not be so when Π_1 and Π_3 are live. For such protocols, our notation of (k, s, l) tuples fall short of characterizing the security properties. Therefore, we develop a new model for the security of overlay protocols synthesized from k underlay chains.

5.3.1 The Model

As the security of a blockchain consists of safety and liveness, we use $s, l \in \{0,1\}^k$ to denote the list of predicates indicating which underlay chains are guaranteed to be safe and live. Specifically, $s_i = 1$ if the i-th underlay chain is guaranteed to be safe, and $s_i = 0$ if the i-th chain is not guaranteed to be safe. Then, the security properties of an overlay protocol Π can be characterized by two sets $V^S, V^L \subseteq 2^{\{0,1\}^{2k}}$, which express the dependence of Π's safety and liveness on the safety and liveness of the underlay chains. Namely, $(s, l) \in V^S$ if the overlay protocol Π is guaranteed to be safe when for all i such that $s_i = 1$, the i-th underlay chain is guaranteed safety, and for all j such that $l_j = 1$, the j-th chain is guaranteed liveness. Similarly, $(s, l) \in V^L$ if the overlay protocol Π is guaranteed to be live when for all i such that $s_i = 1$, the i-th underlay chain is guaranteed safety, and for all j such that $l_j = 1$, the j-th chain is guaranteed liveness. We hereafter use the (V^S, V^L) characterization of security in lieu of the (k, s, l) tuples. Given these definitions, any set V^S, V^L for an overlay protocol satisfies

(P1) If $v \in V$, $w \geq v$, then $w \in V$.

A sequence $v \in V$ is called an *extreme* element if there is no $w \in V$ such that $w < v$. Let $\mathrm{exm}(V)$ be the set containing all extreme elements in V. By property (P1), $\mathrm{exm}(V)$ uniquely describes V, and any protocol Π can be characterized by the two sets $E^S, E^L \subseteq 2^{\{0,1\}^{2k}}$ consisting of the extreme elements in V^S and V^L: $E^S = \mathrm{exm}(V^S)$ and $E^L = \mathrm{exm}(V^L)$.

5.3.2 The Result

Given the model above, the security properties achievable by overlay protocols under partial synchrony can be described as follows. For $s \in \{0,1\}^n$, let us define $\mathrm{ind}(s) = \{i : s_i = 1\}$.

▶ **Theorem 10.** *For any tuple $(E^S, E^L) \subseteq 2^{\{0,1\}^{2k}}$ such that*

1. *For all $(s, l) \in E^L$, $(s', l') \in E^S$, it holds that $s = l' = 0^k$.*

2. *For all* $(0^k, l^1), (0^k, l^2) \in E^L, (s, 0^k) \in E^S$*, it holds that* $ind(l^1) \cap ind(l^2) \cap ind(s) \neq \emptyset$.

there exists an overlay protocol characterized by a tuple dominating (E^S, E^L).

The proof is in Appendix F.3 and inductively constructs the desired overlay protocol. Intuitively, Theorem 10 states that safety (liveness) of the overlay protocol depends only on the safety (liveness) of the underlay chains, and any two quorums of underlay chains required for the liveness of the overlay protocol must intersect at a chain whose safety is required for the safety of the overlay protocol. Note that Theorem 10 implies Theorem 1, as Theorem 10 characterizes security for all types of overlay protocols, including permutation-invariant ones analyzed by Theorem 1. We opted to present Theorem 1 first for ease of understanding.

6 The Converse for Partial Synchrony

6.1 The Converse Result for Partial Synchrony

We start with the converse result that applies to all overlay protocols under partial synchrony, showing the optimality of our security characterization in Theorem 10.

▶ **Theorem 11.** *Let* Π *be an overlay blockchain protocol. Let* $(s^i, l^i) \in \{0,1\}^n$ *for* $i \in [3]$ *satisfy* $(s^1, l^1) \in V^L, (s^2, l^2) \in V^L, (s^3, l^3) \in V^S$*. Then, we have* $ind(l^1) \cap ind(l^2) \cap ind(s^3) \neq \emptyset$.

Note that the converse exactly matches the second clause of Theorem 10.

Proof. For contradiction, suppose $\text{ind}(l^1) \cap \text{ind}(l^2) \cap \text{ind}(s^3) = \emptyset$. Denote the k underlay blockchains by $\Pi_1, \ldots, \Pi_k$. There are two clients $\mathsf{cl}_1, \mathsf{cl}_2$. Consider the following three worlds.
World 1: All blockchains are safe. The underlay chains Π_i, $i \in \text{ind}(l^1)$ are live, and the others are stalled. The adversary sets $\text{GST} = 0$. Suppose that tx_1 is input to the protocol at time $t = 0$. As $(s^1, l^1) \in V^L$, the overlay blockchain is live. At time $t_1 = t_{\text{fin}}$, the client cl_1 outputs tx_1 as its interchain ledger: $L_{t_1}^{\mathsf{cl}_1} = [\text{tx}_1]$.
World 2: All blockchains are safe. The underlay chains Π_i, $i \in \text{ind}(l^2)$ are live, and the others are stalled. The adversary sets $\text{GST} = 0$. Suppose that tx_2 is input to the protocol at time $t = 0$. As $(s^2, l^2) \in V^L$, the overlay blockchain is live. At time $t_2 = t_{\text{fin}}$, the client cl_2 outputs tx_2 as its interchain ledger: $L_{t_2}^{\mathsf{cl}_2} = [\text{tx}_2]$.
World 3: All blockchains are live. The underlay chains Π_i, $i \in \text{ind}(l^1) \cap \text{ind}(l^2)$ are not safe, and the others, including those in s^3, are safe. For simplicity, let $Q = \text{ind}(l^1) \cap \text{ind}(l^2)$. The adversary sets $\text{GST} = 2t_{\text{fin}}$ and creates a network partition before GST such that client cl_1 can only communicate with the validators in Π_i, $i \in \text{ind}(l^1)$, and client cl_2 can only communicate with the validators in Π_i, $i \in \text{ind}(l^2)$. As a result, for client cl_1, the underlay chains Π_i, $i \notin \text{ind}(l^1)$ seem stalled until at least time $2t_{\text{fin}}$, and for client cl_2, the underlay chains Π_i, $i \notin \text{ind}(l^2)$ seem stalled until at least time $2t_{\text{fin}}$, and Suppose that tx_1, tx_2 are input to the protocol at time $t = 0$. However, the adversary reveals tx_1 only to the honest validators in Π_i for $i \in \text{ind}(l^1)/Q$ and tx_2 only to those in Π_i for $i \in \text{ind}(l^2)/Q$. Moreover, it delays any communication between the validators in Π_i for $i \in \text{ind}(l^1)/Q$ and those in $i \in \text{ind}(l^2)/Q$ until after GST.

As the chains Π_i, $i \in Q$ are not safe, they can simultaneously interact with cl_1 and the chains Π_i, $i \in \text{ind}(l^1)/Q$ as in World 1 and with cl_2 and the chains Π_i, $i \in \text{ind}(l^2)/Q$ as in World 2. To ensure this, the adversary delays any messages from the honest validators of the chains Π_i, $i \in Q$, to cl_1 and cl_2, except the certificates received by cl_1 and cl_2 in Worlds 1 and 2 respectively. As we assume Π_i, $i \in Q$, are not safe, such certificates attesting to conflicting ledgers must exist for the chains Π_i, $i \in Q$. Then, client cl_1 cannot distinguish World 1 and

World 3 before $2t_{\text{fin}}$, which implies that $L_{t_1}^{\mathsf{cl}_1} = [\text{tx}_1]$. Similarly, client cl_2 cannot distinguish World 2 and World 3, which implies that $L_{t_2}^{\mathsf{cl}_2} = [\text{tx}_2]$. However, $L_{t_1}^{\mathsf{cl}_1}$ and $L_{t_2}^{\mathsf{cl}_2}$ conflict with each other, which violates the safety of the overlay protocol. This is a contradiction; as all underlay chains are live, and those in s^3 are safe. ◀

6.2 The Converse Result for Permutation Invariant Protocols under Partial Synchrony

Before proving Theorem 2, we introduce a more comprehensive notation for permutation invariant overlay protocols to capture the fact that the safety (or liveness) of the overlay protocol can depend on *both* the safety and liveness of the underlay chains. Although Theorem 2 shows that the liveness of the underlay chains do not help achieve better safety properties for the overlay (and vice versa), we nevertheless need a notation that allows the *possibility* of such cross-dependence between safety and liveness to argue for the absence of this cross-dependence. Moreover, we will use the new notation for permutation invariant overlay protocols later to describe the achievable security guarantees under synchrony, where the safety of the overlay protocol *depend* on the liveness of the underlay chains.

6.2.1 The Model for Permutation Invariant Overlay Protocols

Permutation invariance means that the overlay blockchain treats the underlays identically.

▶ **Definition 12** (Permutation Invariance). *We say a protocol* Π *is permutation invariant if the sets* V^S *and* V^L *both satisfy that*

(P2) If $v \in V$*, then* $\sigma(v) \in V$ *for all permutations* σ*.*

Here, $v = (s, l)$*, and we define* $\sigma(v) = (\sigma(s), \sigma(l))$*, where* $\sigma(s), \sigma(l) \in \{0,1\}^k$*,* $\sigma(s)_i = s_{\sigma(i)}$*,* $\sigma(l)_i = l_{\sigma(i)}$ $\forall i \in [k]$*.*

By property (P2), we can create an equivalence relation '$\sim$' over the sets in V. We say $v \sim w$, if there exists a permutation $\sigma : [k] \to [k]$ such that $w = \sigma(v)$. The relation '$\sim$' defines a quotient set $V/\sim$, which is the set of equivalence classes in V. Given $v = (s, l)$ and

$$c_s(v) := \#\{i : s_i = 1\},\ c_l(v) := \#\{i : l_i = 1\}, c_{sl}(v) := \#\{i : s_i = l_i = 1\},$$

each equivalence class $\{\sigma(v) | \sigma : [k] \to [k] \text{ is a permutation}\}$ is uniquely represented by a tuple $(c_s(v), c_l(v), c_{sl}(v)) \in \mathbb{N}^3$. As the set $\text{exm}(V)$ (for either V^S or V^L) containing all extreme elements also satisfies (P2), we can also partition $\text{exm}(V)/\sim$ into equivalence classes, each represented by a tuple $(n_s, n_l, n_{sl}) \in \mathbb{N}^3$. Therefore, given property (P1), the set V can be represented by a set of tuples $P = \{(n_s^{(i)}, n_l^{(i)}, n_{sl}^{(i)}) | i \in \mathbb{N}\}$.

Finally, any permutation invariant protocol Π can be characterized by two sets $P^S, P^L \in 2^{\mathbb{N}^3}$, representing V^S and V^L respectively and interpreted as follows: For any $(n_s, n_l, n_{sl}) \in P^S$ and $(m_s, m_l, m_{sl}) \in P^L$, we have

- Π is safe if at least n_s blockchains are safe, n_l blockchains are live, and n_{sl} blockchains are safe and live.
- Π is live if at least m_s blockchains are safe, m_l blockchains are live, and m_{sl} blockchains are safe and live.

Let $V(P) := \{v | c_s(v) \geq n_s, c_l(v) \geq n_l, c_{sl}(v) \geq n_{sl}, (n_s, n_l, n_{sl}) \in P\}$.

For two set pairs (P^S, P^L) and $(\tilde{P}^S, \tilde{P}^L)$ characterizing permutation invariant overlay protocols, we define the partial order $(P^S, P^L) \succeq (\tilde{P}^S, \tilde{P}^L)$ to mean that $V(P^S) \supseteq V(\tilde{P}^S)$ and $V(P^L) \supseteq V(\tilde{P}^L)$. For $v \in \{0,1\}^k$, let us define $\text{ind}(v) = \{i : s_i = 1\}$.

▶ **Lemma 13.** *$(P^S, P^L) \succeq (\tilde{P}^S, \tilde{P}^L)$ if and only if for any $\tilde{p}_1 \in \tilde{P}^S, \tilde{p}_2 \in \tilde{P}^L$, there exists $p_1 \in P^S$ and $p_2 \in P^L$ such that $p_1 \leq \tilde{p}_1$ and $p_2 \leq \tilde{p}_2$.*

Proof. It is sufficient to show that $V(P) \supseteq V(\tilde{P})$ if and only if for any $\tilde{p} \in \tilde{P}$, there exists $p \in P$ such that $p \leq \tilde{p}$. Suppose that we have $V(P) \supseteq V(\tilde{P})$. For any $\tilde{p} = (\tilde{n}_s, \tilde{n}_l, \tilde{n}_{sl}) \in \tilde{P}$, consider $\tilde{v} \in \{0,1\}^{2k}$ such that $c_s(\tilde{v}) = \tilde{n}_s, c_l(v) = \tilde{n}_l, c_{sl}(v) = \tilde{n}_{sl}$. Then, $\tilde{v} \in V(P)$. There exists an extreme element $v \in V$ such that $v \leq \tilde{v}$. Defining $p = (c_s(v), c_l(v), c_{sl}(v))$, we can conclude that $p \leq \tilde{p}$. Suppose that for any $\tilde{p} \in \tilde{P}$, there exists $p \in P$ such that $p \leq \tilde{p}$. For any $\tilde{v} \in V(\tilde{P})$, there exists $\tilde{p} = (\tilde{n}_s, \tilde{n}_l, \tilde{n}_{sl}) \in \tilde{P}$ such that $c_s(\tilde{v}) = \tilde{n}_s, c_l(\tilde{v}) = \tilde{n}_l, c_{sl}(\tilde{v}) = \tilde{n}_{sl}$. Let $p \in P$ such that $p \leq \tilde{p}$. From the definition of $V(P)$, we have $\tilde{v} \in V(P)$. ◀

6.2.2 The Result

The converse result for permutation invariant overlay protocols under partial synchrony, Theorem 2, follows as a corollary of Theorem 11. It shows the optimality of our security characterization in Theorem 1. Its proof is presented in Appendix F.4.

▶ **Theorem 14** (Theorem 2). *Let Π be a permutation invariant overlay blockchain protocol characterized by (P^S, P^L). Consider the tuples $(m_s, m_l, m_{sl}) \in P^L, (n_s, n_l, n_{sl}) \in P^S$. Then, it holds that $n_s \geq 2(k - m_l) + 1$ and $m_l > k/2$.*

7 Circuits for Synchrony

In this section, we construct overlay protocols via circuit composition achieving the security properties claimed by Theorem 3, and show their optimality. As the properties achievable under synchrony are stronger than those achievable under partial synchrony, to bridge the gap between partial synchrony and synchrony, we first introduce the parallel composition as a new protocol primitive in addition to the serial and triangular compositions (*cf.* Section 4.2 for a discussion of the triangular and parallel compositions). We then state the security result for general overlay protocols using the model in Section 5.3.1. Equipped with the model in Section 6.2.1, we subsequently show the security properties claimed for permutation invariant overlay protocols by Theorem 3 as a corollary of the general result. We end with a proof of optimality for both results.

7.1 Parallel Composition

We now describe the parallel composition with two underlay chains, Π_A and Π_B (Alg. 2). Upon getting a transaction from the environment, every honest Π_A and Π_B validator broadcasts the transaction to every other validator.

Let $L^{\mathsf{cl}}_{A,t}$, $L^{\mathsf{cl}}_{B,t}$ and $L^{\mathsf{cl}}_{p,t}$ denote the Π_A, Π_B ledgers and the ledger of the parallel overlay protocol in the view of a client cl at time t. Consider a client cl that has been online for at least t_{fin} time. It obtains the parallel ledger as a function of the Π_A and Π_B ledgers. For this purpose, cl first checks if every transaction in $L^{\mathsf{cl}}_{A,t-t_{\text{fin}}}$ appears in $L^{\mathsf{cl}}_{B,t}$, and if every transaction in $L^{\mathsf{cl}}_{B,t-t_{\text{fin}}}$ appears within $L^{\mathsf{cl}}_{A,t}$ (*the interleaving condition*). If so, it *interleaves* the prefixes of the Π_A and Π_B ledgers to construct the Π_p ledger:

$$L^{\mathsf{cl}}_{p,t} := \textsc{Interleave}(L^{\mathsf{cl}}_{A,t}[:\ell], L^{\mathsf{cl}}_{B,t}[:\ell]), \tag{2}$$

where $\ell := \min(|L^{\mathsf{cl}}_{A,t-t_{\text{fin}}}|, |L^{\mathsf{cl}}_{B,t-t_{\text{fin}}}|)$. $\textsc{Interleave}$ function applied on equal size ledgers L_1, L_2 outputs a ledger L_o such that $L_o[2i-1] = L_1[i]$ and $L_o[2i] = L_2[i]$ for all $i \in [|L_1|]$.

Algorithm 2 The algorithm used by a client cl, online for at least t_{fin} time, to output the ledger $L^{\mathsf{cl}}_{p,t}$ of the parallel composition Π_p instantiated with two constituent blockchains Π_A and Π_B at some time t. The algorithm takes the ledgers $L^{\mathsf{cl}}_{A,t-t_{\text{fin}}}$, $L^{\mathsf{cl}}_{A,t}$, $L^{\mathsf{cl}}_{B,t-t_{\text{fin}}}$ and $L^{\mathsf{cl}}_{B,t}$ output by cl at time t and outputs the ledger $L^{\mathsf{cl}}_{p,t}$. The function $\textsc{Interleave}(L_1, L_2)$ with inputs of same length returns the interleaved ledger L_o such that $L_o[2i-1] = L_1[i]$ and $L_o[2i] = L_2[i]$ for all $i \in [|L_1|]$.

function $\textsc{OutputChain}(L^{\mathsf{cl}}_{A,t-t_{\text{fin}}}, L^{\mathsf{cl}}_{A,t}, L^{\mathsf{cl}}_{B,t-t_{\text{fin}}}, L^{\mathsf{cl}}_{B,t})$
  $\ell \leftarrow \min(|L^{\mathsf{cl}}_{A,t-t_{\text{fin}}}|, |L^{\mathsf{cl}}_{B,t-t_{\text{fin}}}|)$
  if $L^{\mathsf{cl}}_{A,t-t_{\text{fin}}} \subseteq L^{\mathsf{cl}}_{B,t} \wedge L^{\mathsf{cl}}_{B,t-t_{\text{fin}}} \subseteq L^{\mathsf{cl}}_{A,t}$ **then**
    $L^{\mathsf{cl}}_{p,t} \leftarrow \textsc{Interleave}(L^{\mathsf{cl}}_{A,t}[:\ell], L^{\mathsf{cl}}_{B,t}[:\ell])$
  else
    $i^* \leftarrow \mathsf{argmax}_{i\in\{A,B\}} |L^{\mathsf{cl}}_{i,t-t_{\text{fin}}}|$
    $L^{\mathsf{cl}}_{p,t} \leftarrow \textsc{Interleave}(L^{\mathsf{cl}}_{A,t}[:\ell], L^{\mathsf{cl}}_{B,t}[:\ell]) || L^{\mathsf{cl}}_{i^*,t}[\ell:]$
  end if
  return $L^{\mathsf{cl}}_{p,t}$
end function

If the interleaving condition fails, then cl interleaves the prefixes of the two ledgers and outputs the remainder of the longer ledger: defining $i^* = \mathsf{argmax}_{i\in\{A,B\}} |L^{\mathsf{cl}}_{i,t-t_{\text{fin}}}|$, it sets

$$L^{\mathsf{cl}}_{p,t} := \textsc{Interleave}(L^{\mathsf{cl}}_{A,t}[:\ell], L^{\mathsf{cl}}_{B,t}[:\ell]) || L^{\mathsf{cl}}_{i^*,t}[\ell:]. \tag{3}$$

The parallel composition satisfies the security properties below:

▶ **Theorem 15.** *Consider the parallel composition Π_p instantiated with the blockchain protocols Π_A and Π_B. Then, under synchrony,*

1. *Π_p satisfies safety if* both *Π_A and Π_B are safe and live.*
2. *Π_p satisfies liveness with constant latency if* either *Π_A or Π_B is live with constant latency.*
3. *Π_p generates certificates if both Π_A and Π_B do so.*
4. *Π_t proceeds in epochs of fixed duration if Π_A and Π_B do so.*

Proof of Theorem 15 is presented in Appendix F.5. It shows that if both chains are safe and live, the interleaving condition is satisfied, ensuring the safety of the Π_p ledger. If either chain is live, then all transactions up to the length of the longer chain is output, ensuring the liveness of the Π_p ledger. Note that the parallel composition does not satisfy accountable safety despite satisfying safety. This is not too surprising since its safety requires both the safety and liveness of the underlay chains.

7.2 General Circuits for Synchrony

▶ **Theorem 16.** *For any tuple $(E^S, E^L) \subseteq 2^{\{0,1\}^{2k}}$ such that*

1. *For all $(s, l) \in E^L$ it holds that $s = 0^k$,*
2. *For all $(0^k, l^1), (0^k, l^2) \in E^L, (s, l) \in E^S$, it holds that*
   a. *either there are indices $i \in ind(l^1)$ and $j \in ind(l^2)$ such that $(s_i, l_i, s_j, l_j) = (1, 1, 1, 1)$,*
   b. *or $ind(l^1) \cap ind(l^2) \cap ind(s) \neq \emptyset$,*

there exists an overlay protocol characterized by a tuple dominating (E^S, E^L).

We present the proof of Theorem 16 in Appendix F.8. It shows achievability by constructing a circuit very similar to that constructed by Theorem 10. Intuitively, Theorem 16 states that for the safety of the overlay protocol, either any two quorums of underlay chains required for the liveness of the overlay protocol must both contain at least one safe and live chain (which can be different), or these quorums must intersect at a safe chain.

7.3 Permutation Invariant Circuits for Synchrony

Following lemma proves the achievability guarantees claimed by Theorem 3. It uses the notation of Section 6.2.1 and follows as a corollary of Theorem 16.

▶ **Theorem 17** (Theorem 3, Achievability). *If a protocol is characterized by* (P^S, P^L) *within*

$$\{(\{(2(k-m_l)+1,0,0),(0,0,k-m_l+1)\},\{(0,m_l,0)\})|k/2<m_l\leq k\},$$

then there exists a permutation invariant overlay protocol characterized by a tuple dominating (P^S, P^L) *under synchrony.*

8 The Converse for Synchrony

We now show the optimality of our security characterization in Theorem 3.

8.1 The Converse Result for Synchrony

We start with the converse result that applies to all overlay protocols under synchrony, showing the optimality of our security characterization in Theorem 16.

▶ **Theorem 18.** *Let* Π *be an overlay blockchain protocol. Let* $(s^i, l^i) \in \{0,1\}^n$ *for* $i \in [3]$ *satisfy* $(s^1, l^1) \in V^L, (s^2, l^2) \in V^L, (s^3, l^3) \in V^S$*. Then, it holds that*

1. *either there are indices* $i \in ind(l^1)$ *and* $j \in ind(l^2)$ *such that* $(s_i, l_i, s_j, l_j) = (1,1,1,1)$*,*
2. *or* $ind(l^1) \cap ind(l^2) \cap ind(s) \neq \emptyset$*.*

Proof of Theorem 18 is presented in Appendix F.9, and relies on an indistinguishability argument between different worlds like the proof of Theorem 11. Note that the converse exactly matches the clause of Theorem 16.

Theorem 18 is reduced to Theorem 11 if the set of security functions are restricted to those, where safety of the overlay protocol depends only on the safety of the underlay chains, and the liveness of the overlay protocol depends only on the liveness of the underlay chains. This is consistent with [40, Theorem B.1], which proves that the accountable safety-liveness tradeoff under synchrony is the same as the safety-liveness tradeoff under partial synchrony.

8.2 The Converse Result for Permutation Invariant Protocols under Synchrony

The converse result for permutation invariant overlay protocols under synchrony as claimed in Theorem 3, follows as a corollary of Theorem 18. It shows the optimality of our security characterization in Theorem 3.

▶ **Theorem 19** (Theorem 3, Converse). *Let* Π *be a permutation invariant overlay blockchain protocol characterized by* (P^S, P^L)*. Consider the tuples* $(m_s, m_l, m_{sl}) \in P^L, (n_s, n_l, n_{sl}) \in P^S$*. Then, it holds that either* $n_{sl} \geq k - m_l + 1$ *or* $n_s \geq 2(k-m_l)+1$ *and* $m_l > k/2$*.*

Theorem 19 follows as a corollary of Theorem 18, and its proof is in Appendix F.10.

Appendix E summarizes all of the results in Sections 5, 6, 7 and 8 by identifying all pareto-optimal protocols under partial synchrony and synchrony using the language developed in Sections 5.3.1 and 6.2.1.

9 Efficiency

Our model of interchain consensus protocols in Section 3 allows the validators of the underlay blockchains to read the ledgers of the other underlays. For each validator, this implies a communication load proportional to the number of underlay chains, *i.e.*, low scalability. However, all of our compositions (serial, triangular and parallel) can be modified to retain their security properties when the validators merely run light clients of the other chains. For instance, the serial composition in Section 4.1 can be instantiated with *succinct timestamps* as in [42]; so that the constituent protocols receive and order timestamps of the blocks of the other protocols rather than snapshots of the whole ledger. Here, timestamps can even be made less frequent for more efficiency (albeit at the expense of latency). When the timestamps are implemented with binding hash functions, their ordering suffices to resolve forks on the other chains and ensure safety as long as any chain is safe.

The triangular construction in Section 4.2 requires the validators of each underlay chain to only follow a smart contract, dedicated to executing the OFT protocol, on the other chains to react to their OFT protocol messages. This again warrants at most a light client functionality. Finally, in the parallel construction of Section 7.1, the validators of the underlay blockchains need not communicate at all except broadcasting the circuit-composition related transactions. In turn, external observers (clients) are responsible for interleaving their ledgers. Therefore, all three composition methods, and by induction, our circuit constructions can work with underlay validators running light clients of each other's chains. Implementation of these constructions with light clients is left as future work.

10 Conclusion

In this work, we have analyzed the security of interchain consensus protocols under synchrony and partial synchrony. We next outline a few open questions and future directions implied by our work. As our serial composition requires the underlay chains to produce certificates and the protocols secure under the sleepy network model [36] (dynamic availability [27], unsized setting [30]) do not generate certificates [30], our results do not extend to underlay chains secure under the (synchronous) sleepy network model (no protocol can be secure under both partial synchrony and the sleepy network model [29]). It is thus an open question to design a serial composition for underlay chains that do not generate certificates.

Although we have instantiated the triangular composition with underlay chains that proceed in fixed time durations, the composition can also work with *optimistically responsive* protocols. These protocols achieve latency that is $O(\delta)$, where δ is the real-time network delay, under optimistic conditions. They can keep track of time with the help of an oracle committee of so-called *time keepers* [41] that input the real time into the protocol. Another alternative that does not require any trust in oracles is for the smart contracts on the underlay chains to adaptively estimate time. For instance, if the contracts notice that the overlay protocol has not made progress while the underlay protocols have, it can slow down the underlay protocols. It is future work to formalize the details of these solutions.

Our recursive compositions of circuits could require an underlay blockchain to appear in exponentially many sub-circuits. Our goal in this work was to show the achievability of the properties proven for the interchain consensus protocols. For small numbers of underlay chains, our results coupled with the optimizations in Section 9 still yield practical constructions for the safety-favoring points. It is an open question to design more scalable interchain consensus protocols for all points.

References

1 Cosmos: The Internet of Blockchains. URL: https://cosmos.network/.

2 The Inter-Blockchain Communication protocol. Website. URL: https://cosmos.network/ibc/.

3 ICS ?: Recursive Tendermint, 2019. URL: https://github.com/cosmos/ibc/issues/547.

4 cosmos/relayer: An IBC relayer for IBC-Go. Website, 2023. URL: https://github.com/cosmos/relayer.

5 Mesh security, 2023. URL: https://github.com/osmosis-labs/mesh-security.

6 Mesh security. Youtube, 2023. URL: https://www.youtube.com/watch?v=GjX4ejD_cRA&t=4670s.

7 Polygon 2.0: Protocol Architecture, 2023. URL: https://polygon.technology/blog/polygon-2-0-protocol-vision-and-architecture.

8 Stacks - A Bitcoin Layer for Smart Contracts, DeFi, NFTs, and Apps, 2023. URL: https://www.stacks.co.

9 Ittai Abraham and Gilad Stern. Information Theoretic HotStuff. In *OPODIS*, volume 184 of *LIPIcs*, pages 11:1–11:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

10 Sarah Azouvi and Marko Vukolic. Pikachu: Securing pos blockchains from long-range attacks by checkpointing into bitcoin pow using taproot. In *ConsensusDay@CCS*, pages 53–65. ACM, 2022.

11 Fabrice Benhamouda, Craig Gentry, Sergey Gorbunov, Shai Halevi, Hugo Krawczyk, Chengyu Lin, Tal Rabin, and Leonid Reyzin. Can a public blockchain keep a secret? In *TCC (1)*, volume 12550 of *Lecture Notes in Computer Science*, pages 260–290. Springer, 2020.

12 Ethan Buchman, Jae Kwon, and Zarko Milosevic. The latest gossip on BFT consensus. *arXiv:1807.04938*, 2018. URL: https://arxiv.org/abs/1807.04938.

13 Navin Budhiraja, Keith Marzullo, Fred B. Schneider, and Sam Toueg. Optimal primary-backup protocols. In *WDAG*, volume 647 of *Lecture Notes in Computer Science*, pages 362–378. Springer, 1992.

14 Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv:1710.09437*, 2019. URL: https://arxiv.org/abs/1710.09437.

15 Christian Cachin, Klaus Kursawe, and Victor Shoup. Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography. *J. Cryptol.*, 18(3):219–246, 2005.

16 Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *OSDI*, pages 173–186. USENIX Association, 1999.

17 Benjamin Y. Chan and Elaine Shi. Streamlet: Textbook streamlined blockchains. In *AFT*, pages 1–11. ACM, 2020.

18 Cynthia Dwork, Nancy A. Lynch, and Larry J. Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, 1988.

19 Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.

20 Matthias Fitzi, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ledger combiners for fast settlement. In *TCC (1)*, volume 12550 of *Lecture Notes in Computer Science*, pages 322–352. Springer, 2020.

21 Bela Gipp, Norman Meuschke, and Andre Gernandt. Decentralized trusted timestamping using the crypto currency bitcoin. In *Proceedings of the iConference 2015*, 2015.

22 Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On robust combiners for oblivious transfer and other primitives. In *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 96–113. Springer, 2005.

23 Thomas Hepp, Patrick Wortner, Alexander Schönhals, and Bela Gipp. Securing physical assets on the blockchain: Linking a novel object identification concept with distributed ledgers. In *CRYBLOCK@MobiSys*, pages 60–65. ACM, 2018.

24 Amir Herzberg. On tolerant cryptographic constructions. In *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 172–190. Springer, 2005.

25 Manuel Huber, Julian Horsch, and Sascha Wessel. Protecting suspended devices from memory attacks. In *EUROSEC*, pages 10:1–10:6. ACM, 2017.

26 Dimitris Karakostas and Aggelos Kiayias. Securing proof-of-work ledgers via checkpointing. In *IEEE ICBC*, pages 1–5. IEEE, 2021.

27 Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *CRYPTO (1)*, volume 10401 of *Lecture Notes in Computer Science*, pages 357–388. Springer, 2017.

28 Leslie Lamport. The part-time parliament. In *Concurrency: the Works of Leslie Lamport*, pages 277–317. ACM, 2019.

29 Andrew Lewis-Pye and Tim Roughgarden. Resource pools and the cap theorem. *arXiv:2006.10698*, 2020. URL: `https://arxiv.org/abs/2006.10698`.

30 Andrew Lewis-Pye and Tim Roughgarden. How does blockchain security dictate blockchain implementation? In *CCS*, pages 1006–1019. ACM, 2021.

31 Sinisa Matetic, Mansoor Ahmed, Kari Kostiainen, Aritra Dhar, David M. Sommer, Arthur Gervais, Ari Juels, and Srdjan Capkun. ROTE: rollback protection for trusted execution. In *USENIX Security Symposium*, pages 1289–1306. USENIX Association, 2017.

32 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. URL: `https://bitcoin.org/bitcoin.pdf`.

33 Joachim Neu, Ertem Nusret Tas, and David Tse. Snap-and-Chat protocols: System aspects. *arXiv:2010.10447*, 2020. URL: `https://arxiv.org/abs/2010.10447`.

34 Joachim Neu, Ertem Nusret Tas, and David Tse. Ebb-and-flow protocols: A resolution of the availability-finality dilemma. In *IEEE Symposium on Security and Privacy*, pages 446–465. IEEE, 2021.

35 Brian M. Oki and Barbara Liskov. Viewstamped replication: A general primary copy. In *PODC*, pages 8–17. ACM, 1988.

36 Rafael Pass and Elaine Shi. The sleepy model of consensus. In *ASIACRYPT (2)*, volume 10625 of *Lecture Notes in Computer Science*, pages 380–409. Springer, 2017.

37 Rafael Pass and Elaine Shi. Thunderella: Blockchains with optimistic instant confirmation. In *EUROCRYPT (2)*, volume 10821 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2018.

38 Michael O. Rabin. Randomized byzantine generals. In *FOCS*, pages 403–409. IEEE Computer Society, 1983.

39 Claude E Shannon. A symbolic analysis of relay and switching circuits. *Electrical Engineering*, 57(12):713–723, 1938.

40 Peiyao Sheng, Gerui Wang, Kartik Nayak, Sreeram Kannan, and Pramod Viswanath. BFT protocol forensics. In *CCS*, pages 1722–1743. ACM, 2021.

41 Peiyao Sheng, Xuechao Wang, Sreeram Kannan, Kartik Nayak, and Pramod Viswanath. Trustboost: Boosting trust among interoperable blockchains. In *CCS*, pages 1571–1584. ACM, 2023.

42 Ertem Nusret Tas, Runchao Han, David Tse, and Mingchao Yu. Interchain timestamping for mesh security. In *CCS*, pages 1585–1599. ACM, 2023.

43 Ertem Nusret Tas, David Tse, Fangyu Gai, Sreeram Kannan, Mohammad Ali Maddah-Ali, and Fisher Yu. Bitcoin-enhanced proof-of-stake security: Possibilities and impossibilities. In *SP*, pages 126–145. IEEE, 2023.

44 Ertem Nusret Tas, David Tse, and Yifei Wang. A circuit approach to constructing blockchains on blockchains. *arXiv:2402.00220*, 2024. URL: `https://arxiv.org/abs/2402.00220`.

45 Wenbin Wang, Chaoshu Yang, Runyu Zhang, Shun Nie, Xianzhang Chen, and Duo Liu. Themis: Malicious wear detection and defense for persistent memory file systems. In *ICPADS*, pages 140–147. IEEE, 2020.

46 Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan-Gueta, and Ittai Abraham. Hotstuff: BFT consensus with linearity and responsiveness. In *PODC*, pages 347–356. ACM, 2019.

47 Alexei Zamyatin, Mustafa Al-Bassam, Dionysis Zindros, Eleftherios Kokoris-Kogias, Pedro Moreno-Sanchez, Aggelos Kiayias, and William J. Knottenbelt. Sok: Communication across distributed ledgers. In *Financial Cryptography (2)*, volume 12675 of *Lecture Notes in Computer Science*, pages 3–36. Springer, 2021.

48 Dionysis Zindros, Apostolos Tzinas, and David Tse. Rollerblade: Replicated distributed protocol emulation on top of ledgers. *IACR Cryptol. ePrint Arch.*, page 210, 2024.

A Blockchains vs. Validators

In this section, we investigate an analogue of our interchain circuit construction for validators as protocol participants rather than blockchains. This connection was first made by Recursive Tendermint [3], which proposed implementing a consensus protocol, originally developed for validators, on top of Tendermint-based blockchains representing those validators and using IBC communication for message passing. As our construction was developed for blockchains that produce certificates, our validators can be assumed to have authenticated communication, *i.e.*, all messages are accompanied with unforgeable signatures. As argued in Section 4.2, a natural analogue of a chain that is safe, but not live is a validator with omission faults that can drop incoming and outgoing messages arbitrarily, but cannot sign conflicting messages. In this context, what should be the fault model for a validator corresponding to a blockchain that is not safe, but live? For such a validator, we take inspiration from the rollback attacks on trusted execution environments (TEEs), where the TEE equivocates by rolling its state back to a previous point and creating two conflicting execution traces from that point onward [25, 31, 45]. In the same vein, a validator corresponding to a not safe but live chain emulates the behavior of multiple fictitious validators simultaneously. Each fictitious validator creates an execution trace that is computationally indistinguishable from the trace of an honest validator, yet might be conflicting with the traces of other fictitious validators. We call a validator with this fault model a *split-brain* validator.

As in Section 4.2, when the consensus protocol has an execution environment (*e.g.*, the Ethereum Virtual Machine), we use omission faults to also capture the behavior of a validator that outputs a *single* invalid execution trace. This is because an invalid trace is detectable and can be rejected by other validators, essentially reducing this validator to an omission-fault validator.

A natural conclusion of the above model is that a blockchain that is neither safe nor live should represent a validator that is simultaneously a split-brain and omission-fault validator. As such, it emulates multiple fictitious validators with conflicting executions, each of which, taken independently, is an omission-fault validator. Note that this is a strictly weaker model of protocol violation than a Byzantine adversary that can deviate from the protocol arbitrarily. A funny, but all the same, real example of a Byzantine validator is a computer that runs the protocol code honestly along with a software that draws diagrams of blockchains. Although such validators are not captured by the notion of a split-brain and omission-fault validator, we argue that for almost all consensus protocols, a Byzantine validator cannot do any action that is not accessible to a split-brain omission-fault validator and has an effect on the security of the consensus protocol. Therefore, we refer to a split-brain, omission-fault validator as a Byzantine validator.

Finally, given our mapping from blockchains to validators, we can reinterpret our results in Sections 4 and 5 in the context of Figure 1. The x-axis of the figure, which used to denote the fraction of non-live blockchains now identifies the total fraction of Byzantine or

omission-fault validators. Similarly, the y-axis of the figure, which used to denote the fraction of non-safe blockchains now identifies the total fraction of Byzantine or split-brain validators. Based on the results in Sections 4, 5, we conjecture that any point marked blue is achievable by a consensus protocol, whereas no consensus protocol can achieve the red points.

B Attack on Serial Composition

Here, we present an attack on the serial composition of non certificate-generating protocols. For the proof of Theorem 5, it is crucial that the protocol Π_A generates certificates. To illustrate this, suppose Π_A is Nakamoto consensus, which does not generate certificates. Then, even if there is no safety violation on Π_A, the adversary can create two forks of the Π_A chain, one shorter, *i.e.*, with less proof-of-work (PoW), than the other. Let L_{shrt} and L_{long} denote the shorter and the longer forks of the Π_A chain respectively. When Π_B loses safety and the clients observe multiple conflicting forks of the Π_B ledger, the adversary can include snapshots of L_{shrt} and L_{long} in different Π_B forks. Note that there is no safety violation in Π_A, as upon observing both L_{shrt} and L_{long}, clients would unanimously output L_{long} as the finalized Π_A chain. However, since Π_B is no longer safe, clients observing the Π_B fork containing L_{short} now output L_{short} as the Π_s ledger, and the clients observing the Π_B fork containing L_{long} output L_{long} as the Π_s ledger. This implies a safety violation even though at least one of the blockchains, namely Π_A, is still safe, contradicting with Theorem 5.

C Impossibility of Partially Synchronous Overlay Protocols on Responsive Underlay Chains

It is impossible to emulate the validators of an overlay protocol that is secure under partial synchrony on underlay protocols that do not proceed in epochs of fixed duration (*e.g.*, on responsive [15] or optimistically responsive [46] protocols). If an overlay protocol is secure under partial synchrony and relies on timeouts, then the underlay protocols must be able to track time via epochs, *i.e.*, should not be responsive. On the other hand, if an overlay protocol is secure under partial synchrony and does not use timeouts, it must then be secure in an asynchronous network. By the celebrated FLP result, if this protocol is deterministic, it must have a non-terminating execution that violates its liveness [19]. To overcome the implications of the FLP result and ensure liveness under asynchrony, most protocols employ a common coin that generates a sequence of unpredictable coin tosses [38, 15]. However, the unpredictability of the future tosses requires validators to keep a secret and reveal it at the time of the coin toss (*e.g.*, when common coins are implemented via threshold signatures, the validators must keep their secret signing keys private and contribute signatures at the time of the toss). When such validators are emulated on a smart contract running on a public ledger, it is not possible to embed a secret into the contract that cannot be read until the time of the coin toss.

Although it is possible to design underlay blockchains that can keep a secret, such designs require secret sharing among the validators of the underlay blockchains as part of the consensus protocol, *i.e.*, a redesigning of the underlays [11]. In this case, an overlay protocol would rely on the validators of the underlay blockchains to reveal their secrets so that the emulated validators of the overlay can contribute unpredictable bits to the common coin. However, this scheme violates the definition of interchain consensus protocol that only uses the blockchain ledgers and do not involve the validators of the underlay blockchains as part of the protocol apart from verifying the finality of the ledgers of the underlay blockchains

(*cf.* Section 3). Therefore, in the interchain protocol paradigm, it is impossible to emulate validators of an overlay protocol that is secure under asynchrony via smart contracts on underlay blockchains that expose public ledgers. This implies that the overlay protocol that is secure under partial synchrony must use timeouts and the underlay protocols must proceed in epochs of fixed duration.

D Detailed Description of the OFT Protocol for Validators

The OFT protocol is a leader-based protocol that proceeds in epochs of 3Δ time. It uses three types of messages: propose, acknowledge, and leader-down. Let $n = 2f + 1$ denote the total number of validators, where f is a protocol parameter. For the parallel composition in Section 4.2, $f = 1$ is sufficient. A collection of acknowledge messages $\langle \text{ack}(B, v) \rangle$ for a block B and epoch v from $f + 1$ validators is called an epoch v certificate for block B and denoted by $C_v(B)$. We assume that the genesis block B_0 is a certified block at epoch 0 for all validators. Each validator keeps track of the proposal it has observed with the highest epoch number.

In each epoch v, a leader L_v is selected via round-robin, *i.e.*, the index of the leader L_v is (v mod n). At the beginning of the epoch, L_v builds a new block depending on the way it entered epoch v:

1. If L_v has previously received acknowledge messages (*i.e.*, $\langle \text{ack}(B_{k-1}, v-1) \rangle$) for a block B_{k-1} and the previous epoch $v - 1$ from $f + 1$ validators, *i.e.*, an epoch $v - 1$ certificate $C_{v-1}(B_{k-1})$ for block B_{k-1}, it builds and proposes a block B_k with the parent pointer $h_k = H(B_{k-1})$.
2. If L_v has previously received leader-down messages (*i.e.*, $\langle \text{leader-down}(B, v-1) \rangle$) for the previous epoch $v - 1$ from $f + 1$ validators, where each message carries a block B, it builds and proposes a block B_k with the parent pointer $h_k = H(B_{k-1})$, where B_{k-1} is the block with the maximal epoch number among the blocks within the $f+1$ leader-down messages.
3. If L_v has not received $f + 1$ acknowledge or leader-down messages by the beginning of epoch v, it does not build or propose a new block.

As any leader must have observed either $f + 1$ acknowledge (*i.e.*, a certificate) or $f + 1$ leader-down messages for epoch $v - 1$ to build a block for epoch v, we call them *tickets* for epoch v.

The detailed protocol description is given below:

1. **Enter.** Epoch v starts at time $3\Delta v$. Upon entering the epoch, L_v builds a block B_k and broadcasts a proposal message $\langle \text{prop}(B_k, v) \rangle$ as described above, along with the ticket. If L_v has not received $f + 1$ acknowledge or leader-down messages by the beginning of the epoch, it does not build or propose a new block.
2. **Acknowledge.** At time $3\Delta v + \Delta$, if a validator has observed a proposal message $\langle \text{prop}(B_k, v) \rangle$ by the epoch leader L_v, it broadcasts an acknowledge message $\langle \text{ack}(B_k, v) \rangle$ for the proposed block B_k and epoch v.
3. **Leader-down.** At time $3\Delta v + 2\Delta$, if a validator has not yet observed $f + 1$ acknowledge messages $\langle \text{ack}(B_k, v) \rangle$ for a block B_k proposed at epoch v by L_v, it sends a leader-down message $\langle \text{leader-down}(B_{k'}, v) \rangle$ for epoch v, where $B_{k'}$ is the block associated with the maximal epoch number v' at which the validator previously sent an acknowledge message $\langle \text{ack}(B_{k'}, v') \rangle$.

All validators **commit** a block B proposed at some epoch v by the correct leader L_v and its prefix chain upon observing the proposal $\langle \text{prop}(B, v) \rangle$ and a certificate $C_v(B)$ for block B and epoch v.

E Pareto-Optimal Overlay Protocols

We define pareto-optimality and explicitly identify all pareto-optimal protocols under partial synchrony and synchrony using the language developed in Sections 5.3.1 and 6.2.1 and combining the earlier results.

▶ **Definition 20** (Pareto-Optimality). *We say that an overlay protocol* Π dominates *an overlay protocol* Π' *(denoted by* $\Pi \succeq \Pi'$*) if the associated set pairs* (V^S_Π, V^L_Π) *and* $(V^S_{\Pi'}, V^L_{\Pi'})$ *satisfy* $V^S_{\Pi'} \subseteq V^S_\Pi$, $V^L_{\Pi'} \subseteq V^L_\Pi$. *We say that* Π strictly dominates Π' *(denoted by* $\Pi \succ \Pi'$*) if the associated set pairs* (V^S_Π, V^L_Π) *and* $(V^S_{\Pi'}, V^L_{\Pi'})$ *satisfy* $V^S_{\Pi'} \subseteq V^S_\Pi$, $V^L_{\Pi'} \subseteq V^L_\Pi$, *and either* $V^S_{\Pi'} \subset V^S_\Pi$ *or* $V^L_{\Pi'} \subset V^L_\Pi$. *We say that an overlay blockchain protocol* Π *is* pareto-optimal *if there exists no* Π' *such that* $\Pi' \succ \Pi$.

▶ **Theorem 21** (Partial Synchrony). *Consider the set of tuples* $(E^S, E^L) \subseteq 2^{\{0,1\}^{2k}}$ *that satisfy the following properties:*

1. *for all* $(s, l) \in E^S, (s', l') \in E^L$, *it holds that* $s' = l = 0^k$,
2. *for all* $(0^k, l^1), (0^k, l^2) \in E^L, (s, 0^k) \in E^S$, *it holds that* $ind(l^1) \cap ind(l^2) \cap ind(s) \neq \emptyset$, *and*
3. *for all* $(s, 0^k) \in E^S$, *for all* $s' < s$, *there exist* $(0^k, l^1), (0^k, l^2) \in E^L$ *such that* $ind(l^1) \cap ind(l^2) \cap ind(s') = \emptyset$.

Under a partially synchronous network, all pareto-optimal overlay protocols are characterized by the tuples (E^S, E^L) *within this set.*

Theorem 21 follows from Theorems 10 and 11. Theorem 10 constructs overlay protocols characterized by the (E^S, E^L) tuples in the set described above under partial synchrony. Then, Theorem 11 shows that no overlay protocol can achieve the security properties required of any element outside this set under partial synchrony.

▶ **Theorem 22** (Synchrony). *Consider the set of tuples* $(E^S, E^L) \subseteq 2^{\{0,1\}^{2k}}$ *that satisfy the following properties:*

1. *For all* $(s, l) \in E^L$ *it holds that* $s = 0^k$.
2. *For all* $(0^k, l^1), (0^k, l^2) \in E^L, (s, l) \in E^S$, *it holds that*
   a. *either there are indices* $i \in ind(l^1)$, $j \in ind(l^2)$ *such that* $(s_i, l_i, s_j, l_j) = (1, 1, 1, 1)$,
   b. *or* $ind(l^1) \cap ind(l^2) \cap ind(s) \neq \emptyset$.
3. *For all* $(s, l) \in E^S$, *for all* $(s', l') < (s, l)$, *there exist* $(0^k, l^1)$, $(0^k, l^2) \in E^L$ *such that*
   a. *for any* (i, j) *such that* $iind(l^1)$, $j \in ind(l^2)$, $(s'_i, l'_i, s'_j, l'_j) \neq (1, 1, 1, 1)$,
   b. *and* $ind(l^1) \cap ind(l^2) \cap ind(s') = \emptyset$.

Under a partially synchronous network, all pareto-optimal overlay protocols are uniquely described by (P^S, P^L) *within this set.*

Theorem 22 follows from Theorems 16 and 18. Theorem 16 constructs overlay protocols characterized by the (E^S, E^L) tuples in the set described above under synchrony. Then, Theorem 18 shows that no overlay protocol can achieve the security properties required of any element outside this set under partial synchrony.

▶ **Theorem 23** (Permutation Invariance, Partial Synchrony)**.** *Under a partially synchronous network, all pareto-optimal permutation invariant overlay protocols are characterized by* (P^S, P^L) *within:*

$$\{(\{(2(k-m_l)+1,0,0)\},\{(0,m_l,0)\})|k/2 < m_l \leq k\}$$

Theorem 23 follows from Theorems 1 and 2, *i.e.*, Theorem 14. Theorem 2 constructs overlay protocols for all (P^S, P^L) in the set $\{(\{(2(k-m_l)+1,0,0)\},\{(0,m_l,0)\})|k/2 < m_l \leq k\}$ under partial synchrony. Theorem 2 then shows that no overlay protocol can achieve the security properties required of any element outside this set under partial synchrony.

▶ **Theorem 24** (Permutation Invariance, Synchrony)**.** *Under a synchronous network, all pareto-optimal permutation invariant overlay protocols are characterized by* (P^S, P^L) *within:*

$$\{(\{(2(k-m_l)+1,0,0),(0,0,k-m_l+1)\},\{(0,m_l,0)\})|k/2 < m_l \leq k\}$$

Theorem 24 follows from Theorem 3, *i.e.*, Theorems 17 and 19. Theorem 17 constructs overlay protocols for all extreme elements in $(P^S, P^L) = \{(\{(2(k-m_l)+1,0,0),(0,0,k-m_l+1)\},\{(0,m_l,0)\})|k/2 < m_l \leq k\}$ under synchrony. Theorem 19 shows that no overlay protocol can achieve the security properties of the elements outside the set under synchrony.

F Security Proofs

F.1 Proof of Theorem 5

The proof uses the following propositions:

▶ **Proposition 25.** *Consider two different ledgers* L^x *and* L^y*. If* $L_x \preceq L_y$*, then* $L_x \preceq \textsc{Clean}(L^x, L^y)$*.*

Proof follows from the definition of the sanitization function $\textsc{Clean}(.,.)$.

▶ **Proposition 26.** *Consider three different ledgers* L^x*,* L^y *and* L^z*. If the ledgers* L^x*,* L^y *and* L^z *are all consistent with each other, then* $\textsc{Clean}(L^x, L^y)$ *is consistent with* L^z*.*

Proof. Without loss of generality, we can assume that $L^x \preceq L^y$. Then, we have $\textsc{Clean}(L^x, L^y) = L^y$. As L^y and L^z are consistent, it follows that $\textsc{Clean}(L^x, L^y)$ is consistent with L^z. ◀

Proof of Theorem 5. We first prove the safety claim. Suppose Π_A satisfies safety (Fig. 2b). Consider two clients x and y (which can be the same client) who observe the (potentially conflicting) Π_B ledgers L^x_{B,t_1} and L^y_{B,t_2} at times t_1 and t_2 respectively. As Π_A generates certificates, all Π_A ledgers with certificates of finality are consistent with each other. Therefore, all certified snapshots snp_i, $i = 1, 2, \ldots$, of the Π_A ledger appearing within L^x_{B,t_1} are consistent with each other and the certified snapshots snp'_i, $i = 1, 2, \ldots$, of the Π_A ledger appearing within L^y_{B,t_2} (and vice versa). Note that to obtain the Π_s ledgers L^x_{s,t_1} and L^y_{s,t_2}, clients x and y iteratively run the sanitization step at Line 5 of Alg. 1 on the snapshots snp_i, $i = 1, 2, \ldots$, and snp'_i, $i = 1, 2, \ldots$, respectively. Then, by iteratively applying Proposition 26 on these snapshots, we observe that the Π_s ledger L^x_{s,t_1} obtained by client x at time t_1 is consistent with the Π_s ledger L^y_{s,t_2} obtained by client y at time t_2. Hence, for any clients x and y and times t_1 and t_2, the Π_s ledgers in the view of x and y at times t_1 and t_2 are consistent.

Now, suppose Π_B satisfies safety (Fig. 2c). Then, for any two clients x and y and times t_1 and t_2, the Π_B ledgers L^x_{B,t_1} and L^y_{B,t_2} are consistent. This implies that the sequence of certified snapshots $(\mathsf{snp}_1, \ldots)$ of the Π_A ledger appearing within L^x_{B,t_1} is a prefix of the

sequence of certified snapshots ($\mathsf{snp}'_1, \ldots$) of the Π_A ledger appearing within L^y_{B,t_2} (or vice versa). Again, to obtain the Π_s ledgers L^x_{s,t_1} and L^y_{s,t_2}, clients x and y iteratively run the sanitization step at Line 5 of Alg. 1 on the snapshots snp_i, $i = 1, 2, \ldots$, and snp'_i, $i = 1, 2, \ldots$, respectively. Then, by iteratively applying Proposition 25 on these snapshots, we observe that the Π_s ledger L^x_{s,t_1} obtained by client x at time t_1 is consistent with the Π_s ledger L^y_{s,t_2} obtained by client y at time t_2. Hence, it holds that for any clients x and y and times t_1 and t_2, the Π_s ledgers in the view of x and y at times t_1 and t_2 are consistent.

Finally, when both Π_A and Π_B are live with parameter t_{fin} after GST, if the environment inputs Π_s a transaction tx at time t, the transaction appears and stays in the certified Π_A ledger of an honest Π_A validator by time $\max(\text{GST}, t) + t_{\text{fin}}$ and is input to Π_B. Then, it appears and stays in the Π_B ledger of every client as part of a certified Π_A snapshot by time $\max(\text{GST}, t) + 2t_{\text{fin}}$. Note that the first instance of tx cannot be sanitized while clients generate their Π_s ledgers from their Π_B ledgers. Hence, tx appears and stays in the Π_s ledgers of all clients by time $\max(\text{GST}, t) + 2t_{\text{fin}}$.

When both Π_A and Π_B generate certificates, so does Π_s; since the Π_s ledger is extracted from the certified Π_B ledger and the certified Π_A snapshots within that ledger. Moreover, when both Π_A and Π_B proceed in epochs of fixed duration, so does Π_s; as both the Π_B ledger and the snapshots therein grow in discrete time steps. ◀

F.2 Proof of Theorem 6

When all blockchains generate certificates, so does Π_p; since the overlay protocol generates certificates (by virtue of being secure under partial synchrony). Moreover, when all blockchains proceed in epochs of fixed duration, so does Π_p; as the overlay protocol proceeds in epochs of fixed duration.

Given the above observations for points (3) and (4) in Theorem 6, proof of Theorem 6 follows from Theorems 30 and 32. In the subsequent proofs, we only consider validators emulated by smart contracts on the finalized and valid Π_i ledgers observed by clients (*i.e.*, emulated overlay validators). When we talk about two separate actions by a validator, these actions might have appeared in different execution traces observed by different clients or the same client at different times. We use the same notation as in Appendix D for the overlay protocol specific messages exchanged among the validators via the CCC abstraction. All blocks referred below are blocks of the overlay protocol.

▶ **Proposition 27.** *If a blockchain is safe, the validator emulated by the smart contract running on this blockchain does not send two conflicting vote messages $\langle vote(B, v)\rangle$ and $\langle vote(B', v)\rangle$ with the same epoch v, in the views of any (potentially the same) clients. Similarly, no leader L_v of any epoch v proposes two conflicting blocks for v, in the views of any (potentially the same) clients.*

This proposition directly follows from the safety of the Π_i ledgers.

▶ **Proposition 28.** *No validator creates an invalid execution of the overlay protocol in the view of any client.*

This proposition directly follows from the fact that the clients refuse to output ledgers with invalid executions according to the external validity rules.

▶ **Lemma 29.** *Suppose that every blockchain is safe. Then, no two validators commit two conflicting overlay blocks in the views of any (potentially the same) clients.*

Proof. Suppose two validators val and val' commit two overlay blocks B and B' at epochs v and $v' \geq v$ respectively in the views of two (potentially the same) clients. We next show that $B \preceq B'$:

- Suppose $v = v'$. Since the leader L_v proposes a single block for its epoch by Proposition 27, $B = B'$.
- We show the $v' > v$ case by induction:

  Suppose $v' = v+1$. Then, val must have received $f+1$ acknowledge messages $\langle \text{ack}(B, v) \rangle$ by unique validators for block B. If L_{v+1} received a certificate $C_v(B)$ for block B and epoch v before entering epoch $v+1$, then by Proposition 28, the proposed block B' extends B. If L_{v+1} received $f+1$ leader-down messages before entering epoch $v+1$, by quorum intersection, at least one of them must have been sent by a validator who sent an acknowledge message $\langle \text{ack}(B, v) \rangle$ for block B and epoch v. By Proposition 27, if a leader-down message contains a block from epoch v, that block must be B. Then, by Proposition 28, the leader L_{v+1} proposes $B' = B_{k+1}$ extending B.

  Suppose that for epochs less than v', the block proposed by the leader extends or is equal to B. For epoch v', if $L_{v'}$ received a certificate $C_{v'-1}(B'')$ for some block B'' and epoch $v'-1$ before entering epoch v', then by Proposition 28, the proposed block $B' = B_{k'}$ extends B'' and thus B. If $L_{v'}$ received $f+1$ leader-down messages for epoch $v'-1$ before entering epoch v', by quorum intersection, at least one of them must have been sent by a validator who sent an acknowledge message $\langle \text{ack}(B, v) \rangle$ for block B and epoch v. Thus, at least one of the leader-down messages must have included a block B'' from an epoch $v'' \geq v$. By Proposition 27, if a leader-down message contains a block from epoch v'', that block must be B'', and without loss of generality, let B'' be the block with the highest epoch that is included among these leader-down messages. Then, by Proposition 28, the leader $L_{v'}$ proposes $B_{k'}$ extending B''. From the assumption, every block proposed by the leaders in epochs in $[v, v'-1]$ extends or is equal to B. This implies that $B_{k'}$ extends B. Finally, since the leader $L_{v'}$ proposes a single block for epoch v' by Proposition 27, $B' = B_{k'}$ extends B, *i.e.*, $B \preceq B'$.

◀

Finally, we prove safety of the overlay protocol when every constituent blockchain is safe.

▶ **Theorem 30.** *Suppose that every blockchain is safe. Then, the parallel composition is safe.*

Proof. Recall that a client accepts an overlay block and its prefix chain if it is part of the committed chains of 2 or more validators. By Lemma 29, all blocks committed by all validators in the clients' views are consistent. Hence, for any two overlay blocks B and B' accepted by the clients cl and cl' at times t and t', it holds that B and B' are consistent. Moreover, once a client outputs a block and its chain, it does not ever output a shorter chain. Thus, the parallel composition satisfies safety. ◀

We next prove liveness for the emulated validators.

▶ **Lemma 31.** *Suppose that at least* 2 *blockchains are* t_{fin}*-live. Then, all validators emulated on the live blockchains,* i.e.*, live validators, commit a new block proposed by a live validator soon after GST.*

Proof. Suppose the leader L_v of an epoch v starting at some time $t > \text{GST} + t_{\text{fin}}$ is a live validator. By the leader selection rule, there exists such a first epoch with $t < \text{GST} + 4t_{\text{fin}}$.

If L_v observes a certificate $C_{v-1}(B_{k-1})$ for a block B_{k-1} from epoch $v-1$ before entering epoch v, since it is live, it builds a block B_k extending B_{k-1} and broadcasts B_{k-1} along with its ticket. Otherwise, it must have received $f+1$ leader-down messages for epoch $v-1$, as epoch v starts t_{fin} time after GST. Then, L_v builds a block B_k by extending the block $B_{k'}$ associated with the maximal epoch v' among those contained by the $f+1$ leader-down messages, and broadcasts B_k along with its ticket. Hence, in either case, L_v broadcasts a proposal and its ticket.

Upon receiving L_v's proposal within t_{fin} time after its broadcast, all live validators broadcast an acknowledge messages $\langle \text{ack}(B_k, v) \rangle$ for the proposed block. Therefore, by time $t + 2t_{\text{fin}}$, all live validators receive a certificate for the live leader's block, committing it by time $t + 2t_{\text{fin}}$. Thus, all live validators commit a new block proposed by a live validator soon after GST. ◀

Finally, we prove liveness of the overlay protocol when $f+1$ of the constituent blockchains are live.

▶ **Theorem 32.** *Suppose that at least $f+1$ blockchains are t_{fin}-live. Then, the parallel composition is live with constant latency.*

Proof. Recall that a client accepts an overlay block and its chain if it is part of the committed chains of $f+1$ or more validators. By Lemma 31, all live validators, $f+1$ in total, commit a new block proposed by a live validator soon after GST, *i.e.*, within $t = f(3t_{\text{fin}}) + 3t_{\text{fin}}$ time. Hence, all clients observing the parallel composition after time t accept an overlay block proposed by a live validator and its chain. To ensure that this block persists in their ledger, even if it is conflicting with the past outputted ledgers, the clients add the block and its prefix chain into their ledgers and sanitize the duplicate transactions. Since the overlay block proposed by a live validator contains all outstanding transactions received by honest validators of the underlay protocols, liveness is satisfied for clients with constant latency, *i.e.*, $f(3t_{\text{fin}}) + 3t_{\text{fin}}$. ◀

F.3 Proof of Theorem 10

Proof of Theorem 10. Fix a tuple (E^S, E^L) as described by the theorem statement. Let $A := A(E^L) = \{\text{ind}(l) | (0^k, l) \in E^L\}$, and $Q_1, ..., Q_m$ denote the sets of indices within A. Here, m is a finite number, bounded by 2^n. We inductively build the overlay protocol characterized by the tuple (E^S, E^L):

Induction hypothesis: For any (E^S, E^L) as described by the theorem such that $A(E^L)$ consists of m sets (quorums) $Q_1, \ldots, Q_m$, there exists an overlay protocol characterized by a tuple dominating (E^S, E^L).

Base case: Suppose $m = 1$, *i.e.*, $A = \{Q_1\}$. Then, each $(s, l) \in E^S$ requires the safety of one of the protocols in $\{\Pi_i | i \in Q_1\}$ by clause (2). Therefore, the serial composition of the protocols $\{\Pi_i | i \in Q_1\}$ is characterized by a tuple dominating (E^S, E^L). This follows from the iterative application of Theorem 5 on the protocols $\{\Pi_i, i \in Q_1\}$.

Inductive step: Suppose $A = \{Q_1, \ldots, Q_m, Q_{m+1}\}$. By the induction hypothesis, for any $(\tilde{E}^S, \tilde{E}^L)$ as described by the theorem statement such that $A := A(\tilde{E}^L) = \{Q_1, \ldots, Q_m\}$, there exists an overlay protocol Π characterized by a tuple dominating (E^S, E^L). Then, using Π and the triangular composition, we construct the following protocol Π':

- The first protocol of the triangular composition is Π itself.
- The second protocol of the triangular composition is the serial composition of the protocols $\{\Pi_i, i \in Q_{m+1}\}$.

- The third protocol of the triangular composition is the $(2f+1)$-triangular composition, where $f = m-1$, *i.e.*, a $(2m-1)$-triangular composition:
  (i) Of the $2m-1$ protocols composed, m are as follows: For each $i \in [m]$, the i-th composition is the serial compositions of all protocols within $Q_{i,m+1} := Q_i \cap Q_{m+1}$.
  (ii) Remaining $m-1$ of $2m-1$ protocols are copies of Π.

First, we show that Π' is live if there exists $Q \in A$ such that Π_i is live for all $i \in Q$. Suppose there is a $j \in [m]$ such that all Π_i, $i \in Q_j$, are live. Then, by the induction hypothesis, Π is live, *i.e.*, the first protocol of the triangular composition is live. By Theorem 5, all protocols within $Q_{j,m+1}$ are live. As at least m components in the $(2m-1)$-triangular composition are live, the third protocol of the triangular composition is live by Lemma 8 (which can be achieved by a circuit protocol as shown in Appendix F.6). Hence, by Theorem 6, Π' is live.

Now, suppose all Π_i, $i \in Q_{m+1}$, are live. By Theorem 5, the second protocol of the triangular composition is live. Again, by Theorem 5, all protocols within $Q_{j,m+1}$ are live. Therefore, again, the third protocol of the triangular composition is live by the definition of the $(2m-1)$-triangular composition. By Theorem 6, this implies that Π' is live.

We next prove that Π' is safe if for every pair $Q_1, Q_2 \in A \cup \{Q_{m+1}\}$, at least one chain Π_i for $i \in Q_1 \cap Q_2$ is safe. Note that the conditions on $\tilde{E}^S$ require Π to be safe if for every pair $Q_1, Q_2 \in A$, at least one chain Π_i, $i \in Q_1 \cap Q_2$ is safe. As the conditions for the safety of Π' imply those for Π, if they hold, Π is safe. Moreover, these conditions for the safety of Π' imply that at least one chain in Q_{m+1} is safe, which by Theorem 5 implies that the second protocol of the triangular composition is safe. Finally, by the same conditions, for each $i \in [m]$, there exists a protocol that is safe within one of the sets $Q_{i,m+1} := Q_i \cap Q_{m+1}, i \in [m]$. Thus, all of the $2m-1$ protocols within the $(2m-1)$-triangular composition are safe, which implies that the third protocol of the triangular composition is safe by Lemma 8. Consequently, by Theorem 6, Π' is safe. Combining the safety and liveness results for Π', we can conclude that Π' is characterized by a tuple dominating (E^S, E^L). ◀

F.4 Proof of Theorem 14

Proof of Theorem 14. For contradiction, suppose there are tuples $(m_s, m_l, m_{sl}) \in p^L$, $(n_s, n_l, n_{sl}) \in P^S$ such that $m_l > k/2$, but $n_s \leq 2(k-m_l)$. Let $l^1 = 1^{m_l}0^{k-m_l}$, $l^2 = 0^{k-m_l}1^{m_l}$ and $s^3 = 1^{k-m_l}0^{2m_l-k}1^{k-m_l}$. Since the number of 1's in s^3 is $2(k-m_l)$, by definition of permutation invariant protocols, there exist $s^1, s^2, l^3 \in \{0,1\}^k$ such that $(s^1, l^1) \in V^L, (s^2, l^2) \in V^L, (s^3, l^3) \in V^S$. However, $\mathrm{ind}(l^1) \cap \mathrm{ind}(l^2) \cap \mathrm{ind}(s^3) = \emptyset$, which is a contradiction.

Now, suppose there exist a tuple $(m_s, m_l, m_{sl}) \in p^L$ such that $m_l \leq k/2$. Let $l^1 = 1^{m_l}0^{k-m_l}$ and $l^2 = 0^{k-m_l}1^{m_l}$. Since the number of 1's in l^3 and l^2 are at most $k/2$, by definition of permutation invariant protocols, there exist $s^1, s^2 \in \{0,1\}^k$ such that $(s^1, l^1) \in V^L, (s^2, l^2) \in V^L$. However, $\mathrm{ind}(l^1) \cap \mathrm{ind}(l^2) = \emptyset$, which is a contradiction. ◀

F.5 Proof of Theorem 15

Proof of Theorem 15. Without loss of generality, suppose Π_A is live with some parameter t_{fin}. Then, once a transaction tx is input to an honest validator at some time t, it appears and stays in the Π_A ledgers of all clients by time $t + t_{\mathrm{fin}}$. If the interleaving condition is satisfied in the view of a client cl observing the protocol at some time $t' \geq t + 2t_{\mathrm{fin}}$, then $\mathrm{tx} \in L^{\mathsf{cl}}_{B,t'}$. Thus for any such client cl and time $t' \geq t + 2t_{\mathrm{fin}}$, this implies $\mathrm{tx} \in L^{\mathsf{cl}}_{p,t'}$ per eq. (2). Therefore, tx appears (and stays) in the Π_p ledgers of all such clients by time $t + 2t_{\mathrm{fin}}$. If the interleaving condition is not satisfied in the view of a client cl observing the protocol

at some time $t' \geq t + 2t_{\text{fin}}$, then it is still the case that $\text{tx} \in L^{\mathsf{cl}}_{p,t'}$, since $L^{\mathsf{cl}}_{p,t'}$ would then contain every transaction in $L^{\mathsf{cl}}_{A,t'-t_{\text{fin}}}$ per eq. (3). This proves the liveness clause.

Suppose both Π_A and Π_B are safe and live with parameter t_{fin}. Then, for any client cl and time t, it holds that any transaction that appears in the ledger $L^{\mathsf{cl}}_{A,t}$ or $L^{\mathsf{cl}}_{B,t}$ is in the Π_A and Π_B ledgers of all clients by time $t + t_{\text{fin}}$. Therefore the interleaving condition is satisfied in the view of all clients. Then, for any client cl and time t, $L^{\mathsf{cl}}_{p,t}$ is as defined by eq. 2, which is an interleaving of two ledgers. Moreover, by the safety of the chains Π_A and Π_B, for all positive integers ℓ and ℓ', clients cl and cl', and times t and t', the ledgers $\textsc{Interleave}(L^{\mathsf{cl}}_{A,t}[:\ell], L^{\mathsf{cl}}_{B,t}[:\ell])$ and $\textsc{Interleave}(L^{\mathsf{cl}'}_{A,t'}[:\ell'], L^{\mathsf{cl}'}_{B,t'}[:\ell'])$ are consistent with each other. This proves the safety clause.

Finally, if both Π_A and Π_B generate certificates, since the Π_p ledger is a deterministic function of the Π_A and Π_B ledgers when Π_p is safe, the tuple of certificates for the Π_A and Π_B ledgers act as a certificate for the Π_p ledger. Thus, Π_p generates certificates. If both Π_A and Π_B proceed in epochs of fixed duration, chain growth rate of the Π_p ledger is bounded as well, implying that it proceeds in epochs of fixed duration. ◀

F.6 Proof of Lemma 8

Proof relies on Lemma 9 from Section 5.2.

Proof of Lemma 8. We prove by mathematical induction. Suppose that for any $g < f$, $(2g+1)$-triangular is achievable by using $\pi^{(2g+1)}$. Note that the base case is $g = 1$, and we assume that there exists a protocol achieving the $(3, 3, 2)$ point.

By Lemma 9, we note that $\pi^{(2g+2)}$ is also achievable. Let $\Pi_1, \ldots, \Pi_{2f+1}$ be $2f + 1$ different blockchains. Consider all subsets of $\Pi_1, \ldots, \Pi_{2f+1}$ with $2f - 1$ blockchains, which are $f(2f + 1)$ subsets in total. We use $\pi^{(2f-1)}$ for these subsets and obtain $f(2f + 1)$ blockchains, say Π^1_i with $i = 1, \ldots, f(2f + 1)$.

If f is an odd number, then we split $f(2f + 1)$ blockchains into $f + 2$ groups, where $f + 1$ groups has $2f - 1$ blockchains and the remaining group has 1 blockchain. Then, we run $\pi^{(2f-1)}$ on the first $f + 1$ groups and obtain $f + 1$ output blockchains. Finally, we run $\pi^{(f+2)}$ to obtain the final output blockchain.

If f is an even number, then we split $f(f + 1)$ blockchains into $f + 1$ groups, where f groups have $2f - 1$ blockchains and the remaining group has $2f$ blockchain. Then, we run $\pi^{(2f-1)}$ on the first f groups and run $\pi^{(2f,2f-1,f+1)}$ on the last group. Hence, we obtain $f + 1$ output blockchains. Finally, we run $\pi^{(f+1)}$ to obtain the final output blockchain.

It is easy to show that if $\Pi_1, \ldots, \Pi_{2f+1}$ are safe then the protocol is safe. Assuming that at least $f + 1$ of $\Pi_1, \ldots, \Pi_{2f+1}$ are live, we will show that the protocol is live. We prove by contradiction. Suppose that the protocol is not live. First, we note that as at least $f + 1$ of $\Pi_1, \ldots, \Pi_{2f+1}$ are live, at most $f(f + 1)/2$ blockchains among $\{\Pi^1_i\}_{i=1}^{f(f+1)}$ output by $\pi^{(2f-1)}$ running on size $2f - 1$ subsets are not live. This is because to obtain a not-live blockchain Π^1_i as output of $\pi^{(2f-1)}$, at least $f - 1$ of input blockchains need to be not-live, which implies that this subset with size $2f - 1$ needs to drop 2 live blockchains from $\Pi_1, \ldots, \Pi_{2f+1}$.

Now, if f is an odd number, as the protocol is not live, at least $(f + 3)/2$ blockchains as input of $\pi^{(f+2)}$ are not live. This implies that at least $f(f + 1)/2 + 1$ blockchains among $\{\Pi^1_i\}_{i=1}^{f(2f+1)}$ are not live. As $f(f+1)/2+1 > f(f+1)/2$, this leads to a contradiction. On the other hand, if f is an even number, as the protocol is not live, at least $(f + 2)/2$ blockchains as input of $\pi^{(f+1)}$ are not live. This implies that at least $(f + 2)f/2$ blockchains among $\{\Pi^1_i\}_{i=1}^{f(2f+1)}$ are not live. As $(f + 2)f/2 > (f + 1)f/2$, this leads to a contradiction. ◀

F.7 Alternative Proof for Theorem 1

The following is an analogue of Theorem 17 for partial synchrony and stated here for completeness:

▶ **Theorem 33.** *If a protocol is characterized by* (P^S, P^L) *in the set*

$$\{(\{(2(k-m_l)+1,0,0)\},\{(0,m_l,0)\})|k/2<m_l\leq k\},$$

then there exists a permutation invariant overlay protocol characterized by a tuple dominating (P^S, P^L) *under partial synchrony.*

Theorem 33 follows as a corollary of Theorem 10.

Proof. Fix some m_l, $k/2 < m_l \leq k$. Consider the tuple $P = (P^S, P^L) = (\{(n_s = 2(k - m_l) + 1, n_l = 0, n_{sl} = 0)\}, \{(m_s = 0, m_l, m_{sl} = 0)\})$. Let $(V^S, V^L) = V(P)$, $E^S = \text{exm}(V^S)$ and $E^L = \text{exm}(V^L)$.

1. Since $n_l = n_{sl} = m_s = m_{sl} = 0$, for all $(s, l) \in E^L$, $(s', l') \in E^S$, it holds that $s = l' = 0^k$.
2. Since $n_s = 2(k-m_l)+1$, by quorum intersection, for all $(0^k, l^1), (0^k, l^2) \in V^L, (s, 0^k) \in E^S$, it holds that $\text{ind}(l^1) \cap \text{ind}(l^2) \cap \text{ind}(s) \neq \emptyset$.

Therefore, by Theorem 33, there exists an overlay protocol characterized by a tuple dominating (P^S, P^L). Moreover, the overlay protocol suggested by the construction in the proof of Theorem 33 is permutation invariant. ◀

F.8 Proof of Theorem 16

Proof of Theorem 16. Fix a tuple (E^S, E^L) as described by the theorem statement. Let $A := A(E^L) = \{\text{ind}(l)|(0^k, l) \in E^L\}$, and $Q_1, ..., Q_m$ denote the sets of indices within A. Here, m is a finite number, bounded by 2^n. We inductively build the overlay protocol characterized by the tuple (E^S, E^L):

Induction hypothesis: For any (E^S, E^L) as described by the theorem statement such that $A(E^L)$ consists of m sets (quorums) $Q_1, \ldots, Q_m$, there exists an overlay protocol characterized by a tuple dominating (E^S, E^L).

Base case: Suppose $m = 1$, *i.e.*, $A = \{Q_1\}$. Then, each $(s, l) \in E^S$ requires the safety of one of the protocols in $\{\Pi_i|i \in Q_1\}$. Therefore, the serial composition of the protocols $\{\Pi_i|i \in Q_1\}$ is characterized by a tuple dominating (E^S, E^L). This follows from the iterative application of Theorem 5 on the protocols $\{\Pi_i, i \in Q_1\}$.

Inductive step: Suppose $A = \{Q_1, \ldots, Q_m, Q_{m+1}\}$. By the induction hypothesis, for any $(\tilde{E}^S, \tilde{E}^L)$ as described by the theorem statement such that $A := A(E^L) = \{Q_1, \ldots, Q_m\}$, there exists an overlay protocol Π that dominates the protocol characterized by (E^S, E^L). Then, using Π, we construct the following protocol Π' using the triangular composition:

- The first protocol of the triangular composition is Π itself.
- The second protocol of the triangular composition is the serial composition of the protocols $\{\Pi_i, i \in Q_{m+1}\}$.
- The third protocol of the triangular composition is the $(2f + 1)$-triangular composition, where $f = m - 1$, *i.e.*, a $(2m - 1, 2m - 1, m - 1)$ composition:
  (i) Of the $2m - 1$ protocols composed, m are as follows: For each $i \in [m]$, the i-th composition is the serial compositions of all protocols within $Q_{i,m+1} := Q_i \cap Q_{m+1}$ and the parallel compositions of all tuples in $\{(\Pi_{j_1}, \Pi_{j_2})|j_1 \in Q_i, j_2 \in Q_{m+1}\})$.
  (ii) Remaining $m - 1$ of $2m - 1$ protocols are copies of Π.

First, we show that Π' is live if there exists $Q \in A$ such that Π_i is live for all $i \in Q$. Suppose there is a $j \in [m]$ such that all Π_i, $i \in Q_j$, are live. Then, by the induction hypothesis, Π is live, *i.e.*, the first protocol of the triangular composition is live. By Theorem 15, all protocols within $\{\text{parallel}(\Pi_{j_1}, \Pi_{j_2}) | j_1 \in Q_j, j_2 \in Q_{m+1}\}$ are live. By Theorem 5, all protocols within $Q_{j,m+1}$ are live. As at least m components in the $(2m-1, 2m-1, m-1)$ composition are live, the third protocol of the triangular composition is live by the definition of the $(2f+1)$-triangular composition (which can be achieved by a circuit protocol as shown in Appendix F.6). Hence, by Theorem 6, Π' is live.

Now, suppose all Π_i, $i \in Q_{m+1}$, are live. By Theorem 5, the second protocol of the triangular composition is live. Again, by Theorem 15, all protocols within $\{\text{parallel}(\Pi_{j_1}, \Pi_{j_2}) | j_1 \in Q_j, j_2 \in Q_{m+1}\}$ are live, and by Theorem 5, all protocols within $Q_{j,m+1}$ are live. Therefore, again, the third protocol of the triangular composition is live by the definition of the $(2f+1)$-triangular composition. By Theorem 6, this implies that Π' is live.

We next prove that Π' is safe if for every pair $Q_1, Q_2 \in A \cup \{Q_{m+1}\}$, either at least one chain Π_i for $i \in Q_1 \cap Q_2$ is safe or at least two chains Π_{j_1}, Π_{j_2}, $j_1 \in Q_1$, $j_2 \in Q_2$ are safe and live. Note that the conditions on $\tilde{E}^S$ require Π to be safe if for every pair $Q_1, Q_2 \in A$, either at least one chain Π_i, $i \in Q_1 \cap Q_2$ is safe, or at least two chains Π_{j_1}, Π_{j_2}, $j_1 \in Q_1$, $j_2 \in Q_2$ are safe and live. As the conditions for the safety of Π' imply those for Π, if they hold, Π is safe. Moreover, these conditions for the safety of Π' imply that at least one chain in Q_{m+1} is safe, which by Theorem 5 implies that the second protocol of the triangular composition is safe. Finally, by the same conditions, for each $i \in [m]$, there exists a protocol that is safe within one of the sets $Q_{i,m+1} := Q_i \cap Q_{m+1}, i \in [m]$ or $\{\text{parallel}(\Pi_{j_1}, \Pi_{j_2}) | j_1 \in Q_i, j_2 \in Q_{m+1}\}$ by Theorem 15. Thus, all of the $2m-1$ protocola within the $(2m-1, m-1, m-1)$ composition are safe, which implies that the third protocol of the triangular composition is safe by the definition of the $(2f+1)$-triangular composition. Consequently, by Theorem 6, Π' is safe. Combining the safety and liveness results for Π', we can conclude that Π' is characterized by a tuple dominating (E^S, E^L). ◀

F.9 Proof of Theorem 18

Proof of Theorem 18. For contradiction, suppose $\text{ind}(l^1) \cap \text{ind}(l^2) \cap \text{ind}(s^3) = \emptyset$ and without loss of generality, there is no index $i \in \text{ind}(l^1)$ such that $s_i = l_i = 1$. Denote the k underlay blockchains by $\Pi_1, \ldots, \Pi_k$. There are two clients $\mathsf{cl}_1, \mathsf{cl}_2$. Consider the following three worlds.
World 1: All blockchains are safe. The underlay chains Π_i, $i \in \text{ind}(l^1)$, are live. The chains Π_i, $i \in \text{ind}(l^2)$ are not live, and the rest are stalled. Suppose transactions tx_1 and tx_2 are input to the protocol at time $t = 0$. The chains Π_i, $i \in \text{ind}(l^2)$, start with the transactions and output ledgers in cl_1's view. However, they seem stalled to the validators of the chains in some set $P_l \subseteq \{\Pi_i | i \in \text{ind}(l^1) \setminus \text{ind}(l^2)\}$. As $f_L^\Pi(s^1, l^1) = 1$, the overlay blockchain is live. At time $t_1 = t_{\text{fin}}$, the client cl_1 outputs its interchain ledger containing tx_1 and tx_2. Without loss of generality, suppose tx_1 appears as the first entry: $L_{t_1}^{\mathsf{cl}_1} = [\text{tx}_1, \text{tx}_2]$.
World 2: All blockchains are safe. The underlay chains Π_i, $i \in \text{ind}(l^2)$, are live. The chains in the set P_l are not live, but those in $\{\Pi_i | i \in \text{ind}(l^1)\} \setminus P_l$ are live. The rest are stalled.

Suppose that tx_1 and tx_2 are input to the protocol at time $t = 0$. The chains in P_l seem stalled to cl_2 and the validators of the chains Π_i, $i \in \text{ind}(l^2) \setminus \text{ind}(l^1)$. As $f_L^\Pi(s^2, l^2) = 1$, the overlay blockchain is live. At time $t_1 = t_{\text{fin}}$, cl_2 outputs its interchain ledger containing tx_1 and tx_2. Suppose tx_2 appears as the first entry: $L_{t_1}^{\mathsf{cl}_2} = [\text{tx}_2, \text{tx}_1]$.
World 3: None of the chains in $\text{ind}(l^1)$ is both safe and live. Here, P_s and P_l denote the chains Π_i, $i \in \text{ind}(l^1)$, that are *not* safe and live respectively. The chains Π_i, $i \in \text{ind}(l^1) \cap \text{ind}(l^2)$ are not safe (but live), and Π_i, $i \in \text{ind}(l^2) \setminus \text{ind}(l^1)$, are both safe and live. For simplicity, let

$Q = \text{ind}(l^1) \cap \text{ind}(l^2)$.

Suppose that tx_1, tx_2 are input to the protocol at time $t = 0$. The chains in P_l seem stalled to cl_2 and the validators of Π_i, $i \in \text{ind}(l^2)/Q$ until at least time t_{fin}, and pretend like the chains Π_i, $i \in \text{ind}(l^2)/Q$ are stalled. The chains in P_s expose two conflicting ledgers, the ledgers in world 1 to client cl_1 and the validators of Π_i for $i \in \text{ind}(l^1)/Q$, and the ledgers in world 2 to client cl_2 and the validators of Π_i for $i \in \text{ind}(l^2)/Q$. As the chains Π_i, $i \in Q$ are not safe, they simultaneously interact with cl_1 and the chains Π_i, $i \in \text{ind}(l^1)/Q$ as in World 1 and with cl_2 and the chains Π_i, $i \in \text{ind}(l^2)/Q$ as in World 2. Then, as the chains in P_s cover all chains Π_i, $i \in \text{ind}(l^1)/Q$ that are live, client cl_1 cannot distinguish World 1 and World 3 before t_{fin}, which implies that $L^{\mathsf{cl}_1}_{t_1} = [\text{tx}_1, \text{tx}_2]$. Similarly, client cl_2 cannot distinguish World 2 and World 3, which implies that $L^{\mathsf{cl}_2}_{t_2} = [\text{tx}_2, \text{tx}_1]$. However, $L^{\mathsf{cl}_1}_{t_1}$ and $L^{\mathsf{cl}_2}_{t_2}$ conflict with each other, which violates the safety of the overlay protocol. This is a contradiction. ◀

F.10 Proof of Theorem 19

Proof of Theorem 19. Suppose there are tuples $(m_s, m_l, m_{sl}) \in p^L, (n_s, n_l, n_{sl}) \in P^S$ such that $m_l > k/2$, but $n_s \leq 2(k - m_l)$ and $n_{sl} \leq k - m_l$ for contradiction. Let $l^1 = 1^{m_l}0^{k-m_l}$, $l^2 = 0^{k-m_l}1^{m_l}$, $s^3 = 1^{k-m_l}0^{2m_l-k}1^{k-m_l}$ and $l^3 = 1^{k-m_l}0^{2m_l-k}1^{k-m_l}$. Since the number of 1's in s^3 is $2(k - m_l)$ and the number of indices that are 1 in both s^3 and l^3 is $k - m_l$, by definition of permutation invariant protocols, there exist $s^1, s^2 \in \{0,1\}^k$ such that $(s^1, l^1) \in V^L, (s^2, l^2) \in V^L, (s^3, l^3) \in V^S$. However, in this case, $\text{ind}(l^1) \cap \text{ind}(l^2) \cap \text{ind}(s^3) = \emptyset$, and there is no index $j \in \text{ind}(l^2)$ such that $s^3_j = 1$ and $l^3_j = 1$, which is a contradiction.

Now, suppose there exist a tuple $(m_s, m_l, m_{sl}) \in p^L$ such that $m_l \leq k/2$ and $n_{sl} \leq k - m_l$. Let $l^1 = 1^{m_l}0^{k-m_l}$, $l^2 = 0^{k-m_l}1^{m_l}$, $s^3 = 1^k$ and $l^3 = 1^{m_l}0^{k-m_l}$ Since the number of 1's in l^3 and l^2 are at most $k/2$, by definition of permutation invariant protocols, there exist $s^1, s^2 \in \{0,1\}^k$ such that $(s^1, l^1) \in V^L, (s^2, l^2) \in V^L, (s^3, l^3) \in V^S$. However, in this case, $\text{ind}(l^1) \cap \text{ind}(l^2) = \emptyset$, and there is no index $j \in \text{ind}(l^2)$ such that $s^3_j = 1$ and $l^3_j = 1$, which is a contradiction. ◀

F.11 Alternative Proof for Theorem 17

Here, we give an alternative proof for Theorem 17 using Sync Streamlet as an overlay protocol in the fashion of Trustboost [41]. Consider synchronous Streamlet [17] with the quorom size q and denote it as Π. We claim that Π is safe if $n_s \geq 2(k - q) + 1$ or $n_{sl} \geq k - q + 1$.

▶ **Lemma 34.** *Suppose $n_s \geq 2(k-q)+1$ or $n_{sl} \geq k-q+1$. If two blocks B_1, B_2 get notarized in consecutive epoch $e, e+1$ and B_1 is with length l, then, no block with length l and epoch greater than $e+2$ is notarized in honest view.*

Proof. Suppose that a block B' with length l and epoch e' is notarized where $e' \geq e+3$. Let Q_1 be the set of blockchains which vote for B_2 in epoch $e+1$, let Q_2 be the set of blockchains which vote for B' in epoch e' and let S be the set of blockchain which is safe and live.

If we have $n_s + n_{sl} > 2(k-q)$, as $|Q_1 \cap Q_2| \geq 2q - n$, this implies that $k - n_s - n_{sl} < 2q - k$. Therefore, at least one blockchain in $Q_1 \cap Q_2$ is safe. Let Π_i be a safe blockchain in $Q_1 \cap Q_2$. We note that Π_i has observed B_1 in epoch $e+1$. As the length of B' is l, the voting action for B' from Π_i is invalid. This contradicts with Proposition 28.

If we have $n_{sl} > k - q$, then there exists a safe and live blockchain Π_i that vote for B' in epoch e'. As B_2 get notarized in epoch $e+1$, Π_i has observed B_2 in the beginning of epoch $e+3$. As the length of B' is less than the length of B_2, the voting action for B' from Π_i is invalid. This contradicts with Proposition 28.

◀

▶ **Theorem 35** (Consistency)**.** *Suppose that* $\max\{s + b_s, 2b_s\} > 2(n - q)$*. Suppose that two notarized chains* **chain** *and* **chain'** *of lengths* l *and* l' *respectively both triggered the finalization rule in some honest node's view, that is,* $\texttt{chain}[:l-5]$ *and* $\texttt{chain'}[:l'-5]$ *are finalized in honest view. Without loss of generality, suppose that* $|\texttt{chain}| \leq |\texttt{chain'}|$*. It must be that* $\texttt{chain}[:l-5] \preceq \texttt{chain'}[:l'-5]$