COOL: A Responsive Distributed Ledger Network Architecture

www.cool.network

Abstract

We introduce COOL, a distributed ledger network architecture engineered to address the shortcomings of current general-purpose blockchains concerning *flexibility*, *expandability*, and *unity*. Aiming to establish a resilient and trustworthy neutral groundwork for a more open, auditable, and participatory Internet, COOL utilizes an innovative architecture consisting of three separate yet interconnected tiers:

1. A Foundation Tier, realized through a primary canonical distributed ledger designated the coreChain, which functions as a consensus layer enabling network participants to establish agreement on the system's universal state;
2. An Execution Tier, realized through a multitude of specialized validity rollups designated execChains, which operates as a versatile and varied application hosting framework accommodating an extensive array of application infrastructure demands relating to supported transaction capacity, confirmation speed, computational environments, confidentiality configurations, cost structures, and other adjustable characteristics; and,
3. An Intelligence Tier, realized through an optimization-focused canonical distributed ledger designated the opChain, which offers a mechanism for the network to adjust independently and perpetually to satisfy the evolving requirements of applications by guaranteeing that applications are matched with suitable execChain infrastructure in alignment with each application's stated requirements and within the framework of wider network circumstances to accomplish a system-wide optimal arrangement.

Proof-of-Stake supports the network's financial security and motivates the involvement of *Validators*. Transaction charges for network activities and compensation for infrastructure operators are expressed in the network's fixed-supply native asset, COOL, which additionally functions as a mechanism through which the network is directed by the collective of its holders.

Important Notice: This whitepaper is a preliminary document that is presently in DRAFT status. Multiple additions and modifications are actively being developed, and will be incorporated in the document prior to public release. Particular subjects not yet addressed that will be covered in the publishable document include: (1) specifications of the Core Services, including services for derivatives and spot trading; (2) specifics of the financial mechanisms of Core Services; (3) revised information of the inflation schedule of the COOL token will be incorporated. Additionally, observe that multiple diagrams and graphical depictions of concepts will be incorporated, and that the analysis concerning the Intelligence Tier will be considerably updated to incorporate our most recent research endeavors (as will be distributed in a distinct DRAFT research document).

1 Introduction

Bitcoin served as the inaugural protocol to exhibit how open-source distributed computing systems possessing suitable mathematical and game-theoretic characteristics can deliver trust assurances that eliminate the requirement for centralized intermediaries. Currently, approximately 16 years after its introduction, the Bitcoin network's indigenous currency ranks as the globe's sixth most valuable monetary instrument — trailing only gold, the United States Dollar, the Euro, the Chinese Yuan, and the Japanese Yen.

Building upon the foundation of Satoshi Nakamoto, Bitcoin's anonymous originator, numerous endeavors have extended beyond Bitcoin's initial peer-to-peer electronic cash application to construct Turing-complete general-purpose distributed ledger protocols. A principal impetus behind these endeavors has been the acknowledgment that the contemporary Internet is progressively governed by centralized intermediaries that participate in censorship and extractive conduct, and that are vulnerable to pressure by governmental entities. The aspiration has persistently been that, similarly to how Bitcoin functions as a decentralized alternative to government-issued currency, general-purpose distributed ledger infrastructure could eventually function as a decentralized alternative to centralized web infrastructure, consequently delivering a trustworthy neutral foundation for a more open, auditable, and participatory Internet. Nevertheless, approximately a decade subsequent to the introduction of the initial general-purpose distributed ledger, Ethereum, the domain has struggled to draw a substantial community of application creators and participants.

1.1 Shortcomings of Current Methodologies

The insufficient embrace of general-purpose public distributed ledger infrastructure is chiefly ascribable to the shortcomings of current implementations, encompassing their being *inflexible*, *non-expandable*, and *divided*:

- By inflexible, we signify that general-purpose distributed ledgers are not engineered to adjust independently to satisfy the evolving requirements of applications and participants, and the evolving capacities of infrastructure operators. Despite multi-chain and rollup-focused distributed ledger networks accommodating varied infrastructure demands, their infrastructure arrangements remain predominantly fixed without time-intensive and development-heavy manual interventions, and the responsibility of manually recruiting and sustaining a collection of infrastructure operators.
- By non-expandable, we signify that general-purpose distributed ledgers that claim to have addressed the expandability dilemma (i.e., providing rapid transaction handling and elevated transaction capacity) have usually achieved this solely by having compromised on other fundamental network characteristics, encompassing unity and decentralization.
- By divided, we signify that general-purpose distributed ledgers pursuing sharding, multi-chain, or rollup-focused expansion approaches experience network division. Contingent on the specific expansion approach utilized, this division can present itself as isolated liquidity, dispersed developer attention allocation, and weakened financial security.

Given these shortcomings and others, no current general-purpose distributed ledger infrastructure is prepared to function as a feasible alternative to centralized web infrastructure at a significant magnitude. The aspiration of an accessible web will solely be achieved subsequent to the emergence, actualization, and embrace of a fundamentally novel design that surmounts these inadequacies.

1.2 Principal Contribution

We introduce COOL, a *flexible*, *expandable*, and *unified* distributed ledger network protocol that embraces an innovative design to surmount the shortcomings of current general-purpose distributed ledger protocols. Possessing the ultimate objective of functioning as a trustworthy neutral groundwork for a more open, auditable, and participatory Internet, COOL tackles two principal applications:

4. Delivering a decentralized application hosting infrastructure that addresses the varied and evolving requirements of applications spanning an extensive array of sectors; and,
5. Delivering an integrated ecosystem for the generation, movement, and employment of digital assets.

By design, COOL is a *flexible*, *expandable*, and *unified* system:

- By flexible, we signify that COOL develops its arrangement perpetually to guarantee that the broad-ranging infrastructure demands of applications are satisfied in a resource-efficient fashion;
- By expandable, we signify that COOL facilitates rapid transaction handling, and essentially unlimited computation, data, and transaction capacity without compromising other fundamental characteristics; and,
- By unified, we signify that COOL functions as a singular integrated infrastructure, despite being comprised of a multitude of varied sub-networks.

COOL embraces an innovative structure that distinguishes three separate yet interconnected tiers of capability:

- A Foundation Tier, comprised of a canonical distributed ledger designated the coreChain, which delivers a sturdy consensus layer servicing the remaining tiers of the network;
- An Execution Tier, comprised of a multitude of validity rollups designated execChains, which delivers an array of hosting environments that embrace varied arrangements to satisfy applications' assorted (and continuously evolving) demands; and,
- An Intelligence Tier, comprised of a canonical distributed ledger designated the opChain, which delivers a mechanism for examining network data and subsequently carrying out actions independently to propel the network toward a system-wide optimal arrangement.

Through its innovative structure, COOL constitutes a departure from current distributed ledger network protocols in numerous significant respects that, collectively, deliver a fundamentally novel paradigm for application creators, end-participants, and infrastructure operators.

6. Initially, COOL is application-cognizant and execChain-cognizant, signifying that the protocol itself acknowledges applications and execChains as entities instead of perceiving them simply as smart contracts.
7. Additionally, COOL is self-configuring, signifying that the execChains delivering a hosting environment for applications are generated, arranged, launched, and designated

infrastructure operators independently in reaction to applications' developing requirements.

To facilitate this, network economics and governance are employed; specifically, transaction charges in COOL are imposed on most network activities and are expressed in the network's fixed-supply native asset, the COOL token. On the coreChain, COOL is utilized in a Proof-of-Stake (PoS) Sybil-resistance mechanism that motivates coreChain Validators to operate coreChain nodes and achieve consensus on the network's state. On execChains, COOL tokens are utilized in Prover-Sequencer Separation schemes that motivate the conduct of execChain operators, customarily Sequencers and Provers, while guaranteeing sufficient security. On the opChain, a PoS mechanism is utilized to motivate opChain Validators to execute actions resulting in enhanced network arrangements. Ultimately, COOL is utilized in the governance mechanisms that allow the network protocol to be enhanced in a fashion reflecting the consolidated preferences of the stakeholders.

1.3 Document Structure

Following sections of this document are structured as follows. In Section 2, we present the system's elevated-level structure, participants and responsibilities, and security framework, prior to outlining characteristic operational sequences of the protocol. In Section 3, Section 4, and Section 5, we elaborate on the elements and sub-protocols of the network's Foundation Tier, Execution Tier, and Intelligence Tier, accordingly. In Section 6, we introduce the network's native asset, its utilities, and its supply mechanics. In Section 7, we elaborate on the network's governance mechanisms. In Section 8, we conclude by examining directions of prospective work.

2 Synopsis

The objective of this section is to introduce the COOL network protocol's elevated-level structure, its participants and responsibilities, and its security premises, prior to assembling these by outlining a sequence of the protocol's characteristic operational and transactional sequences.

2.1 Structural Overview

An instantiation of the COOL network protocol is comprised of the subsequent tiers:

- Foundation Tier: (Section 3) Realized in a canonical primary distributed ledger designated the coreChain, the Foundation Tier functions as the network's primary distributed ledger and delivers a mechanism for execChains and the opChain to resolve transactions and to communicate with other execChains, for coreChain Validators to achieve consensus on the network's universal state, and for coreChain Provers to generate validity attestations attesting to the accuracy of the coreChain ledger's record.
- Execution Tier: (Section 4) Realized in a multitude of specialized validity rollups designated execChains, each of which functions as an autonomous sub-network, the Execution Tier embraces varied design arrangements to address the assorted infrastructure demands of Applications functioning across diverse sectors.
- Intelligence Tier: (Section 5) Realized in an optimization-focused canonical distributed ledger designated the opChain, the Intelligence Tier is assigned the responsibility of gathering, handling, and examining data from all tiers, recognizing prospects for satisfying application requirements, and carrying out modifications that propel the network's arrangement toward a universal optimum, all while resolving to the coreChain.

The protocol's Economics and Incentives (Section 6), and Governance mechanisms (Section 7), traverse each of these three tiers.

Foundation Tier: The function of the Foundation Tier, as realized in the coreChain, is to secure the network-at-large and to deliver the remaining tiers of the network with a mechanism to resolve transactions. The coreChain is comprised of the subsequent core elements:

- A Peer-to-Peer element facilitates communication among coreChain nodes (peers) for the objective of disseminating messages and enabling consensus.
- A Consensus element enables nodes to achieve agreement on the network's state by executing a consensus mechanism and a finality sub-protocol.
- An Execution element enables a WASM-based virtual machine for general-purpose onchain computation to process transactions, chiefly those pertaining to System Contracts – native programs that encode segments of the system's logic and enable the enhancement of core protocol capability without necessitating a hardfork.

System Contracts are deployed on the coreChain and are employed for a broad array of protocol-related objectives, either serving the network-at-large or specific execChains. Capability facilitated by the System Contracts servicing the network-at-large encompasses the subsequent:

- Registries capability delivers database-style constructs cataloging the execChains, execChain Operators, and Applications, rendering key metadata accessible to other facets of the system.
- Staking and Delegation capability enables Validators to deposit a financial stake required for engaging in consensus when operating node infrastructure, and enables individual token-holders, as Delegators, to supplement additional security to Validators by committing stake.
- Proving capability enables coreChain Provers to produce validity attestations attesting to the accuracy of the coreChain ledger record (i.e., to execute Proof-of-Valid-Work) for the objective of allowing nodes entering the network to initialize efficiently.
- Minting and Issuance capability enables the generation of the network's native token and the ensuing distribution to Validators, Delegators, and coreChain Provers.
- Data Availability capability enables participants to acquire the raw state, transactions, state diff for any of the execChains operating atop the coreChain infrastructure.
- External Bridges capability enables participants to move tokens from the coreChain to external networks (e.g., Ethereum) and conversely.
- Intelligence Control capability enables the receipt and validation of instructions originating from the Intelligence Tier, and the directing of pertinent execChains to implement modifications determined by the Intelligence Tier.
- Governance capability enables the progression of the network protocol over time at the discretion of a multitude of the network's stakeholders.

Execution Tier: The function of the Execution Tier is to deliver a dynamic and varied hosting environment for Applications and digital assets, and to manage the processing of transactions relating to their utilization. The Execution Tier is realized in a multitude of validity rollups, execChains, which depend on the coreChain for security, resolution, and data availability. execChains embrace varied arrangements regarding transaction capacity, confirmation speed, computational environments, confidentiality configurations, cost structures, and other adjustable characteristics. Despite execChain arrangements varying broadly, each execChain employs a variant of each of the subsequent core elements:

- A Peer-to-Peer element enables communication among the nodes of an execChain (peers) and enables them to receive and disseminate end-participant transactions, sequence transaction batches, generate new blocks processing the transactions, and produce validity attestations for those blocks.
- A Consensus element implemented employing a Prover Sequencer Separation scheme enables an execChain to sequence transactions into batches, in a singular ordered fashion, and to produce a validity attestation for each batch confirming the accuracy of the new state. Sequencing and proving is enabled by Prover-Sequencer Separation schemes that govern an execChain's sequence generation, attestation production, and staking services in a permissionless and decentralized configuration, encompassing the division of charges between Provers and Sequencers.
- An Execution element delivers execChains with an environment for processing transactions and producing outputs before the proving procedure. Distinct execChains can embrace distinct types of execution environments – Ethereum Virtual Machine (EVM), Sealevel Virtual Machine (SVM), WASM-based virtual machine, or an alternative, encompassing native execution (i.e., no virtual machine).

- An Attestation Generation element delivers a circuit enabling an execChain's Provers to generate attestations. This involves accepting as an input the execChain's most recent state root, the sequence of transactions, and the new state root, and subsequently producing a validity attestation for the execution.
- A Pre-Confirmation element enables end-participants of Applications deployed by an execChain to be furnished with near-instantaneous execChain-level transaction assurances during the interval before the execChain transactions have resolved to the coreChain.

Each execChain is depicted on the coreChain by a collection of System Contracts that augment the execChain's capabilities beyond those delivered by the core elements alone. Capabilities facilitated by the coreChain-deployed System Contracts servicing execChains particularly encompass the subsequent:

- State Update capability enables an execChain's state data storage, retrieval, and modification, the validity attestation verification during the state modifications, and the storage of an execChain's validity attestation logic.
- Deposits and Withdrawals capability enables an execChain to facilitate deposits and withdrawals. A deposit leads to the securing of tokens on coreChain and transmitting a signal to mint (or release) the corresponding quantity of this token to the execChain. A withdrawal is executed upon a signal on coreChain following the release of the tokens on the execChain, connected to its state modification, signifying a destruction (or securing) of the token. This contract encompasses provisions for an execChain's escape mechanism enabling end-participants to execChain access assets securely through the coreChain in the circumstance that the execChain experiences an outage.
- Staking and Delegation capability enables an execChain to implement a staking and delegation service involving the securing of tokens (stake) utilizing coreChain's Staking and Delegation services.
- Interoperability capability enables Applications to engage with other Applications regardless of the execChains that each Application is deployed on. It is accomplished in two manners: the bridging of assets from one execChain to another; and, the transmission of arbitrary messages among execChains.
- Intelligence Dispatcher capability enables an execChain to receive command directives from the coreChain's Intelligence Controller capability in order to execute any opChain-directed network arrangement.
- Oracle capability delivers a shared service for execChain's to retrieve off-chain data (e.g., a price feed from an exchange operating on an external network).

Intelligence Tier: The function of the Intelligence Tier, as realized in the opChain, is to gather data from all chains; to handle and examine the data; to ascertain a suitable collection of actions that will enhance the network's health through arrangement modifications – where health is defined concerning a collection of objectives pertaining to resource employment, satisfaction of Applications' demands, operator stability, and other factors; and, to direct the execution of those actions by conveying control commands to the coreChain's Intelligence Control System Contract. The opChain is comprised of the subsequent core elements:

- A Peer-to-Peer element enables communication among opChain nodes (peers) for the objective of disseminating messages and enabling consensus.

- A Consensus element enables the opChain's Change Executors to achieve agreement on the collection of necessary modifications to the network arrangement that are proposed by the opChain's Data Optimizers.
- An Execution element enables the processing of opChain transactions through a WASM-based virtual machine.
- A Data Gathering and Analysis element enables the gathering of data from the coreChain and all execChains, root cause determination, the ensuing handling of this data in order to identify necessary modifications to the network's arrangement, and the proposition of optimizations.
- An Issue Resolution element enables the prioritization of prospective enhancements to the network arrangement in accordance with the opChain's economic mechanisms.
- A Change Execution element enables the execution of enhancements founded on the necessary modifications proposed by the Data Gathering and Analysis element and the priorities established by the Issue Resolution element.

The opChain's operation does not depend directly on any coreChain-deployed System Contracts aside from the Intelligence Control and Intelligence Dispatcher capability discussed previously in relation to the coreChain and the execChains, accordingly.

2.2 Participants and Responsibilities

A node is a network-level piece of software that processes certain logic and is interconnected with other nodes in the same sub-network – either the coreChain, a specific execChain, or the opChain. Nodes execute an implementation of the protocol that is configured for a certain Network Instance ID (e.g., Devnet, Testnet, or Mainnet). For all of the tiers of the COOL network, we presume several Node Types that govern the block data storage levels and non-block data storage levels being sustained by a given participant. The subsequent Node Types apply across all sub-networks:

- A Full Node denotes a node that sustains a subset of the complete block data for recent blocks, in addition to the block header.
- A Light Node denotes a node that sustains block headers for recent blocks.
- An Archival Node denotes a node that sustains historical blocks beyond the recent data sustained by Full Nodes.
- A Superlight Node denotes a node that sustains some data other than block data (e.g., data pertaining to staking status, equivocations in the consensus procedure, or attestations).

The COOL network protocol is multi-function in the sense that it necessitates actors to behave differently contingent on the duties designated to them or assumed voluntarily. Responsibilities are the protocol-level aliases that denote the named collection of duties that the actor executes at a given moment in time. An actor's collection of duties may change over time, and responsibilities are not always mutually exclusive so an actor may undertake more than one responsibility at a time. Contingent on the responsibility's nature, it may or may not necessitate an actor undertaking this responsibility to operate a node or another piece of software. Protocol-level participants on the coreChain, the execChains, and the opChain can undertake distinct responsibilities.

coreChain Responsibilities:

- coreChain Validators collectively achieve agreement on the state of the coreChain and, by extension, all execChains and the opChain. Each coreChain Validator operates a coreChain Full Node and deposits a sufficient stake of the network's native token as a prerequisite for engaging in consensus and receiving rewards.
- coreChain Delegators help secure the network by delegating native tokens to Validators in exchange for a portion of the staking rewards. coreChain Delegators are not required to operate a node of any type.
- Data Availability Challengers ensure adequate service delivery of the Data Availability capability embedded in the System Contracts. Data Availability Challengers operate a coreChain Light Node and ensure data availability by continuously challenging Data Availability Coordinators' knowledge of random chunks of data.

execChain Responsibilities:

- execChain Valiators (also known as Sequencers) combine multiple execChain transactions into batches and then handle the execution of transactions. Sequencers run an execChain Full Node. Depending on the prover-sequencer separation scheme employed by the execChain, the Sequencer role may be complemented by a separate role called Transaction Executor for handling the execution of transactions specifically. More generally, Sequencers can be referred to as execChain Operators (together with Provers).
- Provers generate attestations demonstrating that execChain transactions were correctly executed by the Sequencers. Provers runs an execChain Full Node. For execChains that involve recursive attestations or folding schemes, and depending on the prover-sequencer separation scheme employed, the Prover role may be complemented by a separate role called Attestation Recursion/Folding Coordinator that is responsible for coordinating the attestation generation process. More generally, Provers can be referred to as execChain Operators (together with Sequencers).
- execChain Delegators delegate stake to execChain Operators in order to help secure the execChain in exchange for a share of the execChain's staking rewards. execChain Delegators are not required to run a node of any type.
- Data Availability Coordinators coordinate the execChain's Data Availability solution by forwarding attestations to the Data Availability Challenger on the coreChain. Data Availability Coordinator runs an execChain Light Node.

opChain Responsibilities:

- opChain Validators execute reconfigurations of the network by submitting control commands to the Intelligence Controller on the coreChain, and collectively build opChain blocks and reach agreement on the opChain's state. opChain Validators run an opChain Full Node.
- Data Optimizers collect raw data from execChains and the coreChain, process raw data collected from execChains and the coreChain in order to identify opportunities for improved network configurations based on the opChain's Intelligence Models, then put forward appropriate solutions to the opChain Validators. Data Optimizers run an opChain Full Node.

2.3 Security Framework

The coreChain employs proof-of-stake (PoS) as a Sybil control mechanism. This mechanism requires that coreChain Validators (i.e., consensus-participating node operators) make a financial commitment to the network as a precondition for participation in consensus. In order to participate in staking rewards, coreChain Validators are required to place an amount of stake at risk over a specified staking duration. coreChain Validators that engage in misbehavior, whether maliciously or through negligence, have their stakes slashed. The classical Byzantine assumption for PoS networks states that the security of a network rests on the assumption that reliable consensus can be guaranteed provided only that no more than one-third of the stake is pledged by coreChain Validators that are adversarial. Several prominent existing networks, however, have devised mechanisms that allow for the relaxation of this assumption, to a degree, and the coreChain similarly relies on economic mechanisms to ensure that the coreChain is safe in a practical sense rather than solving for safety in the theoretical sense under the pure Byzantine assumption.

Each execChain is an independent validity rollup whose operators process the execChain's transactions while mirroring the execChain's state to the coreChain periodically. The validity rollup design employed by the Execution Tier provides cryptographic guarantees that make it cryptographically-impossible for an execChain's operators to report an invalid execution to the coreChain. Consequently, each execChain adopts an honest minority assumption: the participation of a single pair of execChain Operators (one Sequencer and one Prover) is sufficient to provide an execChain with the same level of safety as the coreChain. Although a single pair of execChain Operators is sufficient in terms of safety guarantees, reliance on a single pair of execChain Operators gives rise to a single point of failure which may result in liveness violations in the event of an execChain Operator's failure or malicious conduct. To address this liveness issue, to provide redundancy, and to counter the risk of transaction censorship on the part of Sequencers, execChains employ a relaxed variant of consensus. Each execChain has multiple execChain Operators of each type (i.e., multiple Sequencers and multiple Provers) which pledge stake as collateral in a manner similar to a canonical PoS blockchain. This execChain consensus also provides a form of intermediate security during the periods when an execChain is waiting for its state to be updated and finalized over the coreChain.

Although the opChain relies on the coreChain as a settlement layer, the opChain is a canonical blockchain that employs PoS as a Sybil control mechanism and adopts an honest majority assumption. Reliable consensus among the opChain Validators can be guaranteed provided only that no more than one-third of the stake is pledged by opChain Validators that are adversarial. Similarly to the coreChain, however, the opChain relies on economic mechanisms to ensure that the opChain is safe in a practical rather than theoretical sense.

2.4 Protocol

To conclude this synopsis we trace the typical operational flow of the COOL network protocol for each of the three tiers – the Foundation Tier (coreChain), the Execution Tier (execChains), and the Intelligence Tier (opChain).

coreChain operational sequence:

1. coreChain Operators collect transactions:
   (a) Transactions are submitted by end-participants, execChain nodes, Applications, and opChain nodes.
   (b) A routing scheme is employed to deliver transactions to the correct destination.
   (c) coreChain transactions are distributed among all coreChain nodes.
2. A block producer collects transactions to form a block. A consensus mechanism is then employed to achieve agreement on the new block.
3. The new block is distributed to all coreChain nodes.
4. The finality protocol is employed by coreChain nodes to cast votes in order to finalize the block:
   (d) Votes are added to coreChain blocks.
5. A coreChain transaction may be a cross-chain transaction involving two execChains, in which case the destination execChain has access to this information via the coreChain.

execChain operational sequence:

6. The execChain receives transactions related to the Applications and assets hosted by it. This includes transactions from end-participants targeting Applications residing on the execChain, and any other execChain-specific transactions.
7. Transactions are routed via the routing scheme provided by the coreChain.
8. In accordance with the execChain's Prover-Sequencer Separation (PSS) scheme, Sequencers build blocks and Provers provide validity attestations for the new state.
9. The execChain periodically sends to the coreChain a state update containing the transactions, the new state, and a validity attestation.
10. The coreChain Operators employ the consensus mechanism to construct new coreChain blocks and to execute the finality protocol.
    (e) When a block of the coreChain that contains the state update is finalized, the execChain history it describes is deemed final and irreversible.

opChain operational sequence:

11. The opChain's Data Optimizers collect data from all execChains and the coreChain:
    (f) Data Optimizers are assigned to execChains and are responsible for gathering information at run-time.
    (g) The collected data is filtered and processed by the Data Optimizers.
    (h) The processed data is distributed to all opChain nodes.
12. opChain's nodes run an analysis process in order to detect issues and possible lines of action:
    (i) The opChain consensus mechanism is employed to agree on the list of issues and proposed actions.
    (j) The list is distributed to all nodes that are listening to the opChain.
13. A bidding scheme is employed to agree on what actions to take, and at what cost:
    (k) The bidding is done by Applications.
    (l) The bidding process defines the subset of actions that are to be executed; and the cost each Application should pay for the upcoming actions.
14. The opChain Validators execute the agreed upon actions. This is done in collaboration with the coreChain. Notice that the opChain actions do not interfere with the operation of execChains, and the two can make progress in parallel.

(m) The actions to execute are delivered to coreChain in batches, composed of multiple opChain blocks. Each such batch is accompanied with a validity attestation, attesting to the correctness of it and actions within it.

3 Foundation Tier

The objective of this section is to elaborate on the design of the COOL network protocol's Foundation Tier distributed ledger, the coreChain. We commence by examining the coreChain's core elements and then the collection of capabilities that it hosts as System Contracts – native programs that can be enhanced without necessitating a fork. We conclude this section by elaborating on a typical coreChain transaction sequence.

3.1 Core Elements

The coreChain is comprised of core elements that support its Peer-to-Peer, Consensus, and Execution functions.

3.1.1 Peer-to-Peer

The coreChain's peer-to-peer (P2P) element enables communication among nodes (peers) to facilitate consensus and the propagation of external actors' messages, including end-participant transactions. This element is powered by LibP2P, a modular system of protocols, specifications, and libraries that enable the development of peer-to-peer network applications. LibP2P has been proven in a range of centralized and decentralized systems, supports major programming languages, supports multiple transport protocols (e.g., TCP, UDP, QUIC), and provides rich options for efficient data propagation, including solutions for spam and DoS resilience. While LibP2P has no out-of-the-box support for peer discovery fitting the COOL network protocol's requirements, support for peer discovery can be constructed by combining several LibP2P plugins:

8. The Identity plugin facilitates the exchange of peers' basic information in the network (e.g., addresses, public keys).
9. The Bootnodes plugin is a P2P layer entry point that provides a list of well-known nodes in the network which can be connected to and used to discover other nodes.
10. The Kademlia DHT plugin is a distributed hash table (DHT) that allows peer and content routing based on the concept of XOR-distance.

The COOL network protocol carries two kinds of messages over the coreChain's P2P network. Because of their different nature, they are treated slightly differently:

- Consensus-related messages facilitate nodes in a committee to reach consensus for block production and finalization. To save resources and network bandwidth, nodes in a committee send these messages to one another directly, leveraging broadcast via LibP2P's Gossipsub plugin only to propagate resulting block headers. Having obtained the block headers, all interested parties can obtain full blocks separately, as needed. From the time that the consensus protocol adopts multiple committees (as discussed in Section 3.2.2 Consensus), for better liveness and block propagation peer discovery will be constructed such that, with a high enough probability, all committee members will have an open connection with at least one member of all other committees.
- Non-consensus messages are generated by actors external to the consensus mechanism, such as end-participants' submitting transactions and cross-chain interoperability artifacts. These messages are propagated in the network using the Gossipsub plugin in a

bandwidth-efficient manner. When a message is gossiped, the full message is only sent to a subset of nodes which are known to the sender. The remainder of the nodes receive only a lightweight 'fingerprint' of the message but can request the full message only if they have never heard of the message before.

3.1.2 Consensus

The coreChain's consensus element enables coreChain Validators, formed into a single committee, to produce blocks by following a one-step consensus protocol — every block produced can be optimistically considered as final as soon as the next consensus round attempts to extend the previous block. All coreChain Validators in the network must communicate with each other, and at least two thirds of the coreChain Validators (by stake weight) are required to produce a block.

Time on the coreChain is split into slots which are grouped into epochs, and any given slot can have at most one finalized block associated with it. For each slot, a single coreChain Validator is chosen from among all Validators to act as leader, while the remaining coreChain Validators act as attesters who verify the leader's work in that slot.

The consensus element employs a modified version of the HotStuff consensus protocol – it was somewhat modified to keep the coreChain forward-compatible. HotStuff was selected as the basis for the coreChain's consensus because it is well-known, formally defined and analyzed, and well-established, and it supports the changing of the leader in each slot in a way that is efficient. Although it is a one-step protocol, it is employed in such a way as to mimic a two-step protocol for block creation and finalization. This is followed by a third phase called finality attestation stabilization.

- Block creation: When a new slot begins, the leader waits for a supermajority of attesters to send their proposals. Based on the leader's own view, the collected proposals, and a restricted variant of HotStuff's Fork Choice Rule (FCR) that prioritizes a longer chain, the leader chooses which parent block to extend. The leader then extends the chosen parent block, seals it using an EdDSA signature, and sends the signature for the attesters for validation and finalization.
- Block finalization: For the remainder of the slot, several rounds of communication occur between the leader and the attesters. After each round, the leader generates an aggregated BLS signature called the Quorum Certificate (QC), and broadcasts the QC to the attesters and other peers. Every QC carries votes of at least 2/3rds of the attesters (by stake weight). The FCR relies heavily on these QCs, and with each subsequent round, the resulting QC renders more certainty with respect to the block's finality. The QC resulting from the final round of communication serves as cryptographic attestation that the block has been validated and agreed upon by the majority of the committee. It also indicates that the block, and all its ancestors up to the Genesis Block, are considered by the committee to be final. From this point, the block cannot be reverted without a major slashing event concerning at least 1/3rd of the coreChain Validators (by stake weight).
- Attestation stabilization: The finalization process involves multiple QCs, and network peers may fall out of sync on QC sets if they go online and offline at times. To circumvent sync issues, every block other than the direct child of Genesis Block must include the QC of the previous block. After the child block B1 is finalized, the QC that

was included for its parent block B0 is considered to be B0's stable QC, from which point any peers with a different QC for block B0 must switch to the stable QC. Rational leaders are incentivized to include the B0's strongest QC and to create and distribute B1's strongest QC because the strength of QCs influences rewards.

Over time, the coreChain's consensus element will move from a one-step consensus mechanism, to allow for improved performance and to support a greater number of coreChain Validators. Two-step consensus calls for the separation of block production and finalization. In such a scheme, block production involves multiple committees, and blocks are produced with limited safety guarantees but with strong liveness guarantees — blocks can be produced even if fewer than 2/3rds of the coreChain Validators (by stake weight) are online simultaneously. Separately, a single committee runs a finality gadget to finalize the blocks. Further detail regarding the two-step consensus is provided next.

- Block creation: Each epoch, multiple block creation committees are selected at random, and multiple slots are assigned to each committee. In each slot, members of a block production committee will produce a block by running a simplified version of the HotStuff algorithm. This is similar to the process described above under the one-step consensus, but involves fewer rounds and a modified FCR that accounts for multiple committees. Following the production of a block, the block is scheduled for finalization by the coreChain's finality gadget.
- Block finalization: Each epoch, members of a finalization committee are selected iteratively in order to maximize the stake represented by the committee's members in the aggregate. In a given slot, a coreChain Validator may act as a member of both committees simultaneously, so that finalization can occur in parallel to block production. Finalization occurs for each finalization slot, which is equivalent to several block creation slots. For each finalization slot, a continuous span of blocks is selected and finalized. Specifically, if the prior finalized block span was [B0, . . . , Bn], then the subsequent block span for finalization must begin with Bn and be of the form [Bn, . . . , Bmn]. The finalization itself happens by members of the finalization committee executing the HotStuff consensus protocol and agreeing on the block spans that must be finalized. The QC from the prior round indicates true finality and is stored on-chain in subsequent blocks, and all weaker QCs are employed only by finality gadget so that it can advance.

In addition to the introduction of two-step consensus with a dedicated finality gadget, the coreChain's consensus will evolve over time to harness advances in HotStuff, including:

- Chained HotStuff: This is a version of HotStuff that introduces pipeline integration. Instead of having each slot running a single iteration of a full consensus protocol, the different phases of multiple consensus instances can be combined such that the network can start the next consensus instance while still operating the last phases of the previous instance.
- Two-Phase HotStuff: This is a version of HotStuff that requires two phases of communication rather than the three phases employed in Basic HotStuff.
- Pacemaker Improvements: The pacemaker is a HotStuff component that can be made responsible for time settings (e.g., slot length, timeouts setting, synchronization mechanism) and leader selection. We are devising schemes for an advanced Pacemaker

that is optimized for the COOL network's needs, including by employing leader selection based on Single Secret Leader Election schemes.

3.1.3 Execution

The coreChain's execution element employs a virtual machine (VM) with a WebAssembly (WASM) compatible runtime for running smart contracts. WASM is a portable binary-code format and a corresponding text format (.WAT) for executable programs, and a set of software interfaces for facilitating interactions between WASM-based programs (smart contracts) and their host environment (coreChain node).

The coreChain hosts and executes via its WASM VM two types of contracts: System Contracts and User Contracts. Both types of contracts are regular WASM binaries, but have a different set of functions and privileges:

- System Contracts embed logic for system-wide actions that form an integral part of the COOL network protocol, either in service of the network-at-large or of execChains specifically. System Contracts cover functionality relating to staking and delegation, issuance, and data availability, among other key functions discussed in [3.2 coreChain System Contracts] and [4.2 execChain System Contracts (hosted on the coreChain)]. System Contracts can be immutable but are upgradeable by default, and can be run with elevated privileges. By virtue of their design, System Contracts provide a high-degree of flexibility over certain system-wide actions and enable forkless upgrades.
- User Contracts enable third-party developers to run programs on the coreChain. Unlike System Contracts, User Contracts cannot be run with elevated privileges. Although developers will generally prefer to have their apps run over the network's execChains, User Contracts provide a way to run third-party programs for auxiliary functions that serve the network-at-large (e.g., multi-sig treasuries, crowdfunding pools, and on-chain identity services).

The protocol takes a mixed approach to contract execution by combining compilation and interpretation. To bolster performance, System Contracts are always compiled and stored in the node's cache for fast-loading immediately upon node start. All contracts have designated storage which is separated into the following regions:

- The System region is used by the protocol and is not directly accessible from the contract.
- The Metadata region allows for metadata to be associated with the contract (e.g., details of an account holder's offchain identity, such as a website URL).
- The Contract State region stores the state which is used to execute the contract's encoded logic. The region is further divided into two subregions: Primary, which is used to store the state of the contract, and Surrogate, which is used as storage for other (stateless) contracts that lack their own storage.

Contracts on the coreChain can invoke functions exposed by the host environment (coreChain node) to the WASM VM, crossing the WASM-to-host boundary. These functions, called 'imports' in WASM standard, are externally linked pieces of code that the COOL network protocol requires the host to provide during a contract's invocation. Host functions are deterministic and lightweight; their set and interface are fixed and dictated by the protocol. Host functions generally cover use cases that are relatively common, too expensive to implement in smart contracts, or that are inaccessible from within smart contracts (e.g., cryptography-related operations (hashing, RNG), contextual information (account balance and nonce, parent block hash), storage-related operations (read and write), and cross-contract calls).

A gas-metering mechanism provides protection from VM-level DoS attacks. Gas is a measure of the relative time required for primitive VM operations and host functions to execute. Each operation and Host function has its cost expressed in gas units, and every transaction has a gas limit (the maximum number of gas units the transaction initiator is willing to spend) and a gas cost (the COOL-denominated price of a single gas unit). The execution of a transaction proceeds until one of the following stop-criteria is met:

11. The block-level gas limit is reached (i.e., no more gas can be spent in the block);
12. The transaction-level gas limit was reached (i.e., the transaction has no more gas available to fund its execution); or
13. The transaction execution reaches its logical completion, whether successful or not.

Following the execution, the transaction initiator's balance is reduced by the amount of COOL that is equal to the gas cost times the number of gas units the transaction actually consumed. An exception to this flow applies in the case of protocol-triggered executions of System Contracts — although these executions are gas-metered, they do not result in gas being charged.

6 Economics and Incentives

The objective of this section is to introduce the economic and incentive mechanisms that enable the network to develop a vibrant economy that motivates participants while ensuring economic security. We commence by introducing the network's fixed-supply token economics and elaborate on the minting function that determines the token's emissions over time. We then examine how the network's native token motivates participants to stake in exchange for staking rewards. We then elaborate on the transaction fee mechanisms that apply to different tiers of the network, before concluding with some remarks concerning the role the native token plays in on-chain governance.

6.1 Token Economics

The network's native token has a fixed supply of 25,000,000,000 (25B) tokens. Tokens are distributed exclusively to parties that actively contribute to the network by operating infrastructure of one form or another. Beginning at the coreChain's Genesis block, minted tokens are issued as rewards exclusively to node operators according to a predefined rewards function. A minting function defines the rate at which the supply is distributed, in a predictable manner, over a 25-year period – during this period transaction fees will gradually converge with, then surpass, the minting of new tokens as a source of rewards.

6.1.1 Denominations and Notation

The smallest divisible unit of the native token is a nanoCOOL, which is equal to one billionth of a token. For other denominations, we similarly adopt metric prefixes as set forth in the International System of Units (SI) — centiCOOL for 0.01, milliCOOL for 0.001, and microCOOL for 0.000001. In certain contexts, units smaller than a nanoCOOL are required to ensure proper accounting. For this reason, we introduce the notion of atomic units equal to approximately 10^-16 COOL. Atomic units are abstracted away from end-participants through rounding but ensure that the protocol maintains sufficiently precise token balances.

6.1.2 Minting Function

Beginning at the Genesis block, the entirety of the 25B tokens is released in accordance with a predictable token emissions curve. The function according to which the native tokens are minted per-epoch is defined such that it encourages early participation in the network. At the same time, it should have a heavy tail to ensure a prolonged period of non-insignificant reward issuance. Otherwise, most of the tokens would be issued during the early stages of the network, minting only a minuscule amount throughout its maturity, which repels from joining the network at later stages. The right-hand side part of certain bell-shaped functions — such as, for example, Gaussian, Cauchy, or Logistic probability distributions — do satisfy this criterion nicely.

6.2 Purposes

Within the network, the native token serves five primary purposes across all tiers of the network: (1) it is used within the staking and slashing mechanisms that secure the network and incentivize node operators; (2) it is used for transaction fees that deter spam, affect TX ordering, and further incentivize node operators; (3) it is used for network operation – e.g., execChains pay coreChain for finalization and opChain operators get paid for data analysis and network optimization; and (4) it is used to facilitate governance participation.

6.2.1 Staking Rewards

COOL employs proof-of-stake as its Sybil control mechanism and the network's security depends, in part, on the staking that is performed by operators at each tier of the network.

- On the coreChain, coreChain Validators receive rewards based on staking attributes (including delegated stake) and performance, while coreChain Delegators receive rewards as a function of the commission rate set by the Validators being delegated to as well as the selected Validator's performance. coreChain Validators who behave in a malicious manner have their stakes (and the stakes of their Delegators) slashed, and additional mechanisms, as described below, are employed to detect and combat malicious acts that involve collusion.
- On execChains the Sequencers and Provers receive rewards based on staking attributes and performance. Although not mandatory, execChains can elect to support delegation to Sequencers or Provers. For execChains that do support delegation, Delegators receive rewards as a function of the commission rate set by the delegates. Sequencers and Provers who behave in a malicious manner have their rewards (and the rewards of their Delegators, if any) slashed.
- On the opChain, similarly to the coreChain, opChain Validators receive rewards based on staking attributes and performance, while opChain Delegators receive rewards as a function of the commission rate set by opChain Validators. Both the opChain Validators and opChain Delegators are slashed when a slashing event occurs.

6.2.2 Transaction Fees

Transaction fees are levied in the native token for most interactions with the network. Transaction fees apply for interactions with the coreChain, all execChains, and the opChain, though the fee mechanisms employed vary widely depending on the context. In all contexts, transaction fees serve also as a means for mitigating spam and preventing denial of service attacks by imposing a cost on network utilization. Secondarily, as the network matures and becomes less dependent on minting, transaction fees serve as the source of rewards for node operators.

- On the coreChain, transaction fees are levied, inter alia, for staking, delegation, token transfers, and on-chain governance transactions. In addition, execChain Validators (i.e., Sequencers) submitting an attestation of a batch of execChain transactions to the coreChain are required to pay a fee to the coreChain in order for those transactions to be settled, and to cover the corresponding data storage costs.
- On execChains, transaction fees are levied for staking, delegation, token transfers, and all transactions involving Applications. Different execChains can adopt different fee mechanisms which stem in part from execChain's Prover-Sequencer Separation (PSS) scheme. Although by default transaction fees are borne by end-participants interacting with Applications, the protocol supports a sponsored gas fee model which enables Applications to sponsor transactions (within certain bounds) in favor of end-participants

via an account abstraction facility. This notion of sponsored transactions reduces friction and assists Applications in streamlining end-participant acquisition.

- On the opChain, transaction fees are levied for at least the two following, main types of work performed by the Intelligence Tier: core tasks, and proposed jobs. core tasks correspond to infrastructure work and refers to the set of services that are made available to Applications continuously and are performed in the background, irrespective of the network's state. This includes the opChain's continuous data collection, processing, and analysis functions. Proposed jobs refer to the set of actions performed by the opChain in order to drive direct improvements in the network's configuration. These jobs are optional in the sense that the system does not assume that all outstanding jobs will be fulfilled; rather, fulfillment depends on the level of interest of Applications in resolving the issues underlying these actions as well as the capacity of the opChain, as orchestrated by a fee model. (Furthermore, the opChain uses staking as well as pays fees to accommodate for internal operations such as app and operator declarations and bidding.)

7 Governance

The objective of this section is to provide an overview of the algorithmic governance system that enables holders of the native token to direct key decisions in relation to network upgrades and economic parameter changes, among other aspects relating to the network's evolution over time. We commence by discussing the governance principles that inform the design of the governance system. We then describe the institutions that are integral to governance, and elaborate on the supported boundaries and decision types. We conclude by detailing the main governance processes.

7.1 Principles

The principles that inform the design of the governance system include an awareness of limited resources, the need for limitations on governance power, and the trade-offs between governance safety and liveness.

7.1.1 Awareness of Limited Resources

Token-holders have limited resources to devote to their active participation in the decision-making mechanisms. The governance system requires mechanisms for counteracting these limited resources by following the line of thought in the relevant literature. Specific mechanisms include:

- Sampling-based methods: These are methods in which random subsets of community members are assigned to delve into certain issues and decide accordingly. The randomness helps to ensure that the decisions taken by ad-hoc committees align with the wishes of the global population of token holders.
- Delegation-based methods: These are methods in which community members delegate their vote to other members of the community. The cognitive resources required to choose a delegate to make many decisions on one's behalf are far lower than the resources required to delve into all issues by oneself.
- Cryptoeconomic methods: These are methods in which economic incentives are employed to incentivize rational agents to improve the quality of decision making.

Early implementations of the governance system lean heavily on delegation-based and cryptoeconomic methods, while later implementations will employ more techniques that take into account the limited resources towards governance and bolster the attractiveness of governance participation.

7.1.2 Limits to Governance Power

The issue of limits to the power of governance institutions is critical in decentralized networks and in modern democracies more generally because there is an inherent tradeoff between having a governance system that is efficient and powerful while also being spam and attack resistant. This issue is particularly acute in decentralized networks where governance systems are codified in a series of smart contracts — following the concept of 'code is law', any breach in the limits of governance power can be exploited immediately and with a hard remedy (e.g., only by a hard fork). Furthermore, the economic inequality between different token-holders can be drastic (see for an analysis on the Nakamoto index of many DAOs).

7.2 Institutions

In the preliminary set of governance institutions that govern the network, there are two main institutions: the General Assembly and the Council. These institutions are complemented by Technical Committees.

7.2.1 General Assembly

Any holder of the native token is eligible to participate in the General Assembly, which is the **network sovereign**. Participation can take the form of submitting proposals and voting for/against certain proposals. Additionally, members of the General Assembly can delegate voting power to other members of the General Assembly; by doing so, their delegator's voting power is delegated to the chosen delegate on all forthcoming proposals, unless or until the delegator chooses to withdraw the delegation.

The general operation of the General Assembly is built upon the following principles; accordingly, it can be described as a *reality-aware binary-proposal-based liquid democracy:*

- Reality-aware: The General Assembly is built upon the literature on reality-aware social choice. This means that a so-called reality (i.e., status quo) is always present as a possibility to choose from, as a default to stay in; in our context, these are the governance System Contracts. This principle leads to a design in which participants can propose governance proposals that change the status quo if passed.
- Binary-proposal-based: In order to adhere with the principle of governance simplicity and accessibility, the proposals that participants can propose are only binary – so we use a binary-proposal-based system; i.e., they act as a YES/NO question, in which a NO means that the proposal shall not pass and the status quo remains, while a YES means that the proposal shall pass and the status quo changes.
- Liquid democracy: In recognition of limited cognitive resources that are devoted to the active participation in the network's governance, we utilize liquid democracy. We build the infrastructure that allows token holders to delegate their voting power to General Assembly members of their choice. To improve the quality of decision making we also economically incentivize token-holders to actively participate in the decision making process, delegate their vote and serve as public delegates.

7.2.2 Council

The Council is composed of 23 members who are initially appointed by the core developers and then subsequently re-appointed or replaced by the General Assembly. The members of the Council act as civil servants, and their decisions are made public and by name (as well as by a unique identifier, such as their wallet address). The Council serves two purposes:

- It monitors the operation of the General Assembly, in particular the decisions that are about to pass. The Council has affirmative power in that it has the authority to veto any decision it deems to be contrary to the network's best interests, though it can only take such action given a sufficiently-high supermajority – specifically, a 2/3 majority; and, crucially, as the General Assembly is the network sovereign, certain decisions, such as a decision made by the General Assembly to replace Council members are excluded from this vetoing capability.

- It acts as an ad-hoc Emergency Committee in that it can submit and vote on fast-track proposals that may be required in the event of a critical bug or security exploit. Fast-track proposals must be approved by 9/10 of the Council members in order to be passed.

The members of the Council are encouraged to deliberate publicly in public and in community forums. They are required to actively monitor and audit the operation of the General Assembly and, in particular, the decisions that are about to be passed, and to raise corresponding soft flags as well as use their veto power to overrule dangerous decisions that are about to be passed, by mistake, by the General Assembly. The overrules and veto power of the Council on the decisions of the General Assembly is manifested by members of the Council opening corresponding Disputes.

Of the 23 members of the Council, 7 are selected for 2 years, 8 are selected for 1 year, and 8 are selected for 6 months. This mechanism is in place to adhere to the principle of stability and safety (in particular, this reduces the risk of a governance attack to abruptly take full control over the Council). When the end of term reaches for some members of the Council, a general election occurs. In the election to replace the members of the Council, the following principles apply:

- Eligible candidates: Only token holders with sufficient Reputation Points (explained below) are eligible to be selected to the Council.
- Eligible voters: All token holders – i.e., all members of the General Assembly – are eligible to vote, and to do so directly (without delegating their vote as is done for standard decisions of the General Assembly).
- Proportional representation: We use a voting rule that is geared towards proportional representation. We provide the specific details of the election to the Council below.

8 Future Work

In this work we have provided an overview of the COOL network protocol and presented a path towards the construction of the COOL network. This work is not a specification and is not intended to serve as a complete or comprehensive design. Rather, pursuant to further input from the core contributors and the broader community, various mechanisms described herein will be refined and expounded upon, and new mechanisms will be added. Over time we intend to release a full implementable specification of the network protocol that encompasses, where relevant, formal attestations and verification. Several areas of ongoing research include the following:

- Post-quantum cryptography. In recognition of the recent advancements in the field of quantum computing, we intend to perform research into the use of quantum-safe digital signature schemes, including by weighing the pros and cons of different responses to the National Institute of Standards and Technology's (NIST's) call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process, and potentially pursuing our own adaptations or original research in the field.
- Interoperability. Interoperability within the COOL network is subject of research that can be bolstered by recent advances in validity proving systems. We intend to release detailed commentary regarding our interoperability sub-protocols.
- Attestation aggregation and folding schemes. We intend to investigating the latest advances in aggregation and folding schemes for validity attestations to determine how these primitives can be harnessed by execChains, and potentially the coreChain and opChain.
- Advanced consensus mechanisms. We are exploring extensions of the HotStuff consensus protocol that may be applied in the context of future iterations of our consensus engine. Further, we are exploring the notion of hybrid leaderless and leader-based solutions that cycle between the two modes depending on network conditions.
- Finality gadget. We see room to develop a novel finality gadget that does not rely directly on pre-existing protocols. A finality gadget built from the ground up specifically with the COOL network protocol's architecture could drive significant performance improvements when moving to a two-step consensus approach.
- Smart accounts and account abstraction. We intend to undertake additional research regarding account abstraction, with a view to abstracting away the infrastructure from end-participants further. This research will encompass work on mechanisms for enabling wallet creation and transactions without the typical hurdle of creating and storing a seed phrase.
- Optimization. We intend to perform further work on refining the optimization mechanisms that are employed by the opChain when driving continuous improvements to the network's configuration, in the expectation that the set of models employed will expand over time. This includes further data analysis capabilities as well as optimization techniques that take into account the uncertainties in the setting.
- Compliant privacy. We intend to release research relating to a novel zero-knowledge attestations-powered sub-protocol that facilitates cross-chain transactions in a manner that is privacy-preserving while enabling participants to verify cryptographically that the privacy-protected funds are non-illicit.

We hope that this work solicits feedback and questions from interested parties, including community members and prospective core contributors. The exact scope of future work will continue to evolve in response to the needs identified by the COOL core contributors and the broader COOL community.