primal network

...the prime of blockchain with endless possibilities.

Preface

Bitcoin ushered us into a virtual era, an age of digital currency. Safe to say; increasingly digital currency “cryptocurrency” is the new money and evolving norm for global trade and transaction. Prime has taken centre stage at the first digital currency for global community, making available cross-border means of exchange and wealth accumulation with the accessibility of cryptocurrency worldwide.

Primal is the first delegated blockchain that provides multi-chain features with seamless cross-chain DeFi ecosystem which allows instant in-chain (Primal) swap/conversion and variety of possible and probable decentralised lifestyle.

Our Vision

Power the inclusive global market using peer-to-peer system.

Our Goal

Build a blockchain that provides the platform to transact, interact, operate and manage financial and lifestyle affairs.

Our Mission

Provide a cryptocurrency aided and backed by global community.

Make available a blockchain that decentribute (decentralise and distribute) finance and lifestyle

info@primalnetwork.org
admin@primalnetwork.org

Render services and create products relevant to daily financial lifestyle activities

Giving cross-border solutions that bridges continental trade & transaction hassle

Build a marketplace, hub and grounds for global relations, interactions and communications

Our Purpose

Promote unequivocal, safe & secure and swift financial lifestyle environment in the world.

Our Value

Progression

Reliable

Inclusion

Maestro

Advantage

Lifestyle

Primal is a Progressive blockchain technologies corporation that is ardent about the advancement and forward development of borderless/cross-border financial lifestyle with a global community at the core of it. Being Reliable, which assures the community that our service is designed and patterned to Include diversity as one peoples in one global community with a Maestro of creative and novel products and services that adapts to existing norms, sets us ahead for yet-to exist innovations to dictate the standards of information technology. As such, Primal would be the Advantage for our community's financial Lifestyle where safety & security of finance meets dynamic lifestyle solutions.

info@primalnetwork.org
admin@primalnetwork.org

primal network

INTRO

We envisage a borderless global community where Primal is the Bridge to the free flow of activities -transaction, exchange, trade, etc- online and offline. A time, when our identity and data is in our control - secure from central authority.

Primal will make available a decentralised system for daily essentials that will answer and cater humanly possible needs and wants proffering solutions using unrivalled and continuous innovative technology chain that will ensure all that pertains to human affairs for lifestyle are met and decentralised for personal control. Primal will enable effortless connection of features, services, decentralised applications and products from independent blockchains and multi-chains. The network is developed to link several chains and networks -private and public- present and future technologies to be. There will be a seamless trustless flow of transactions and information in our circumchain.

Problems

Some of the current crypto minters have a number of deficiencies:

- **Inadequate Planning**
  Many minting platforms are created with great ideas, but with little to no experience in in community engagement. They often launch their applications with the simplest approach to get the system up and doing without measures in place to manage exponential growth and engagement traffic. Minting systems need to be engineered from the ground up with activities that efficiently engage and interact with the traffic as the community grows, yet maintaining speed, dynamism and security. This allows for convertible growth with convenient and purposeful control of the community, which is critical for long-term success.

- **Unattainable Goal**
  There are some minting network that are struggling in spite of their large community due to their inability to translate their goals into reality. Many of them have been operating the minting platform on the ground of amassing millions of traffic before launching their mainnet, while having no engaging strategy to keep their community alert and active.

- **Valueless Token**

  While many are minting tokens, the creators fail to adequately plan an accurate time to inject value into the tokens minted for the community to use for exchange, transaction, trade and so on. They delay the process of transiting from simulation to real-time asset. This leads to disengagement and loss of the community.

- **Lack of Activities**

  The little to no engagement activities periodically added in the form of updates results to inactivity and loss of interest in the project. The community soon become bored and tired of repeating the process of minting without attractive features to engage and motivate them to remain active and keep spreading the coverage of the project globally.

SOLUTION

This is where Primal comes in: as it will not only distribute, but also deregulate activities and lifestyle by allowing a guide to determine participation and authorities that are valid and/or invalid. This also will eliminate the need for reliance on consensus algorithm. Transaction history will deliberately be given a window of 60 seconds to track after which will be erased beyond trace. Also, there will actually be no centrality, hence, truly achieving decentralisation.

- Scalability: up to 1 billion transactions per second. Confirmations are within 15 seconds
- Multichain: ready features of various blockchains for easy and quick access to any blockchain; with a free-flow interaction without borders or barriers and communicate without limitations and breakdowns.
- Crosschain: instant transactions, swap and conversion of multicurrencies within Primal chain
- Blockchain: a delegated blockchain; first of its kind. A blockchain with no centrality and no dependence on consensus for finality.

ACCESS

Primal will provide all financial and lifestyle services known, and relevant to man in a decentralised platter. Any and every relevant digital and non-digital services and products available will be replicated in the blockchain to provide options and lasting solution to centrality.

- Financial service
- Banking
- Insurance
- Market and commerce
- Trade, swap, exchange
- Sale, listing and so many more

FRAMEWORK

PLATTER: interoperability, consensus, scalabilty, security and various other services will be provided for chains built with Platter. This is the framework the synchrochain is built on and is a readily available platform for use. Independent developers can build their own synchrochain with this framework. Developers can also power blockchains built with Platter without being in Primal Network.

The synchrochain among it functionalities will have a DeFiecosystem chain that serves as liquidity pool for innumerable currencies. The interoperability of Synchrochain cuts into diverse of blockchains like Bitcoin, Ethereum, Binance, Polkadot, Tron and several others. Features found in these various blockchains like ZK-snarcks for Z-cash, UTXO transactions for Bitcoin, Smart-Contracts for Ethereum and more will be contained in the Primal Chain as plug-ins.

primal network

Quick Peak

There are several components that make up the Primal Chain. In fact, the Primal Chain is just a platform for the co-existence of innumerable blockchains and also a foundation for innovative building of novel blockchains as it has an adaptive feature that serve as a “door”. Primal Chain will be used by financial institutions, security firms, entrepreneurs for e-commerce, entertainers for distribution, privacy, and so many more.

- Circumchain: serves as the junction of connection and communication for sychrochains and Primal.
- Sychrochain: is an independent chain that is either built or plugged to run on the circumchain.
- Radius: is like the synchrochain but differs a little as a temporal chain unlike the synchrochain
- Gate: a passageway for connecting Primal to other blockchains like Bitcoin, Ethereum, Binance, Tron and many others.
- Premiers: they fix Primal, confirm proofs from aggregators and partake in consensus
- Delegators: they fix Primal and select premiers
- Aggregators: they collect transactions, create blocks and proofs for premiers
- Observers: they watch and inspect the network and report malicious activity.

PRIMAL

It has one main coin and two tokens native to the Primal chain. It is the most unique cryptocurrency and the first non-fiat backed stable cryptocurrency with little to zero volatility.

- Prime: The code is PRM. It is the main coin and available to the public powered by the Primal Chain.
- Flix: is the fixing token used only for such purpose and not available to the public.

- Raks: is the governance token used for qualification aspremiers and other governance activities.

INITIAL TOKEN CHAIN

The Prime will first be a token on the Binance Smart Chain, being a BEP-20 token. The sales will take place as a BEP-20 token and be listed on exchanges as that before transition into coin in its native blockchain.

FEATURES

- ❖ Multi-chain Support: Bitcoin, Ethereum, Binance, Smart Chain, Polygon, TRON, Polkadot, Cosmos networks and so many more
- ❖ In-chain (within Primal) instant swap and/or conversion of currencies
- ❖ One signature wallet to receive coins from any blockchain and store balance in any coin
- ❖ Exchanges: centralised and decentralised
- ❖ DeFi: complete services which includes NFT, insurance, banking, market/e-commerce, trade and so many more
- ❖ Operating System "OS": mobile and PC devices to power and run applications
- ❖ Software and Application: mobile and PC devices software and apps
- ❖ Digital Services: search engine, e-commerce, social media, messenger and so many more
- ❖ Payment Services: gateway, processor, PoS, ATM and so many more

primal network

USES

- ❖ DeFi projects and/or services
- ❖ PRM: for any kind of transaction
- ❖ Payment
- ❖ Trade
- ❖ Investment
- ❖ Exchange
- ❖ And other financial and daily needs related affairs

TOKENOMICS

There will be no tax on purchase and sale of PRM. You can use PRM for transaction of products or services, including but not limited to:

- Trading
- Investment
- Exchange
- And other financial related affairs

Repurchasing & Recirculating Plan

Every quarter, we will use 20% of our profits to buy back PRM and strategically recirculate them based on market trends, project innovations and price observations; in addition to this, there'll be strategic injection of PRM into the circulating supply. All buy-back transactions will be announced on the circumchain. We will continually buy and periodically recirculate using market data on PRM progress and stand.. This will positively and significantly affect the market price of PRM, thus increasing the value.

Allocation

Amount (PRM)	Participant
10%	Minting and Fix
32%	Sale
20%	Founding Team, Staff
3%	Collaborations, Partnerships and Mergers
35%	Locked and future development

FUNDS ALLOCATION

Funds raised will be allocated in the following:

- Development: 52% will be allocated to development, maintenance, security and infrastructure
- Marketing: 25% will be allocated to marketing and collaboration
- Administration: 7% will be allocated to administration
- Founders: 8% will be allocated to the founding team and investors
- Mergers and Partnerships: 5% will be allocated as benefits for the relationship
- Miscellaneous: 3% will be kept for foreseeable needs

primal network

PRM

MAP

Date	Milestone	Task
2021 Q2	AIRDROP	Community assemblage via minting app and airdrop
2022 Q3	WALLET	Centralised multicurrency wallet services
2023 Q2	CENTRALISED EXCHANGE	Centralised exchange to mark end of phase 1 and signify transition to phase 2
2022 Q3	BLOCKCHAIN	Testnet and live testing
2023 Q2	BLOCKCHAIN	Mainnet and launch
2023 Q2	PRIMAL	Launch of PRM as a native currency
2023 Q3	DECENTRALISED EXCHANGE	Decentralised exchange will be launched to have all functionalities of the centralised exchange in a decentralised model and offer more features
2024 Q1	OTHER DEFI, METAVERSE, NFT AND NOVELTY PROJECTS	innovative projects will be created with dates to be announced
	OS, SOFTWARE & APP	Creation of operating systems, software and applications that meet financial and daily lifestyles. Dates will be announced
	MANY MORE	To be announced

info@primalnetwork.org
admin@primalnetwork.org

primal network

PRIMAL CHAIN

CIRCUMCHAIN

info@primalnetwork.org
admin@primalnetwork.org

primal network

PREFACE

Blockchain is a concept designed and created to be a solution to many problems that have been endemic to our daily lives, but the architectures all suffer from a number of issues not least are means of extensibility and scalability. There's solution to the architecture, which will be expounded in this bluepaper; the *multifarious multi-chain*, which fundamentally sets the two apart.

In compartmentalising these two parts, and by keeping the overall functionality provided to an absolute minimum of *security* and *transport*, we introduce practical means of core extensibility. Scalability is addressed through a divide-and-conquer approach to these two functions, scaling out of its bonded core through the incentivisation of trustless public nodes.

The multifarious nature of this architecture allows many highly diverse types of consensus systems interoperating in a trustless, fully decentralised "federation", allowing open and closed networks to have trust-free access to each other.

We put profer a means of providing backwards compatibility with one or more pre-existing networks such as Bitcoin, Ethereum. We believe that such a system provides a useful base-level component in the overall search for a practically implementable system capable of achieving global-commerce levels of scalability and privacy.

INTRODUCTION

Blockchains have led the way to a whole new dimension of utility over several fields including "Internet of Things" (IoT), finance, governance, identity management, web decentralisation and asset-tracking. However, this hasn't come without it being malnourished on the long run. The hunger to feed more mouths with insufficient resources, rather approach too conservative to keep the flow as adequate and efficient as possible than it inadequacies. The technological fails have been nailed down to five points, taking from Polkadot's observations and highlights of these failures:

- **Scalability:** How much resources are spent globally on processing, bandwidth and storage for the system to process a single transaction and how many transactions can be reasonably processed under peak conditions?
- **Isolatability:** Can the divergent needs of multiple parties and applications be addressed to a nearoptimal degree under the same framework?
- **Developability:** How well do the tools work? Do the APIs address the developers' needs? Are academic materials available? Are the right integrations there?
- **Governance:** Can the network remain flexible to evolve and adapt over time? Can decisions be made with sufficient inclusivity, legitimacy and transparency to provide efficient leadership of a decentralised system?

- **Applicability:** Does the technology actually address a burning need on its own? Is other "middleware" required in order to bridge the gap to actual applications?

Primal will be both a solution and improvement to the deficiencies facing the blockchain technology. Since Bitcoin ushered in the advent of blockchain, there have been implementations such as the Parity Ethereum client that can process in excess of 3,000 transactions per second when running on performant consumer hardware. However, current blockchain networks are in reality still limited to below 50 transactions per second. This is the current state the blockchain is plagued with. This limitation is owed to synchronous consensus mechanisms that require wide timing margins of safety on the expected processing time, which is borne from the conservative approach to support slower implementations by giving room to canonical validations. This consensus is designed as a state transition mechanism by which parties collate and execute transactions. The logic of this mechanism is based on the consensus of parties agreeing upon one of a number of possible, valid, histories.

Further, this concerns proof-of-work (PoW) consensus of both Bitcoin and Ethereum, and proof-of-fix (PoS) consensus of both NXT and Bitshares; all whom are faced with the same limitations. It is a simple strategy that helped make blockchains a success. This strategy is not billed to solve the growing blockchain community and ultimately would lead to a technological collapse; it will be safe to juxtapose the two strategies to make up one protocol, but doing so will mean bringing together under one roof the different difficiencies facing the two distinct protocols,, with their different scalability and privacies. This, however, is more a problem than a solution; thus, Primal will rather than merging the two distinct consensus as one learn and put together the varying differences that works with innovation create a new protocol that solves all the problems faced with the different blockchain protocols. By so doing, there will be a separation of the consensus design from the state-transition which is obviously a common thorn.

Summary

This paper seek to explain the concept of Primal Protocol which will be herein referred as Primal system.

The Primal system is a central open collaborative decentralised network called the circumchain, that communicates with other external chains in a parallel system; the parallel chains are called *synchrochains*. The synchrochains are collaborators of the circumchain, which provides security services to them. The circumchain will mainly be a security chain with no entities running in it, rather all entities are delegated to on-chains or collaborates with external chains which are to be the synchrochains for the various features and functionalities. The circumchain will not be burdened with the dealings and affairs of the synchrochains, rather will be guided and guarded in strict order of the shared communications. The concern of the circumchain with the synchrochains will be to see to their followership or rather adherence of the interface provided; the synchrochains are themselves independent chains which have natural

components of blockchains. The circumchain, also support other non-blockchain systems to run as Primal synchrochains in as much they are adhere to the interface.

Security

Although, the synchrochains being independent blockchains have measures in place to secure transactions, communication with them on the circumchain is regarded to be trustless, hence, interactions are strictly via interface with no involvement in their on-chain affairs. However, Primal provides security ,ensures to validate the exchange in communication with the synchrochains without assumptions of them being secure and trustable. The very instance ofa sniff of perversion, distortion or defilement of the synchrochain, the entirety of the synchrochain is viewed as mischievous and so snuffed out. The circumchain as open decentralised network, thus deals with malevolent activities using measures out in place to obliterate and deter further occurrence or recurrence. Knowing fully that certain nodes are untrusted, while there subset of nodes that are trusted, thus, the protocol ensures that the circumchain is externally as a whole is trustable as much as it is internally.

Nodes and Roles

The Primal circumchain network are composed of nodes and roles. Nodes are network-level entities that maintain and physically execute the Primal software; roles are protocol-level entities that sustain the system by their specific and varying performances.

The circumchain is an open network. Any node may run the software and participate as any of these types of nodes:

1. Modest collaborators - collect certain user-relevant data from the network. The availability of modest collaborators is irrelevant - they don't perform a service for others.
2. Full node - retrieves all types of data, stores it long-term, and propagates it to others. Must be highly available.

   (a) Sentry node - publicly-reachable full nodes that perform trusted proxying services for a private full node, run by the same operator. Also known to be a *full node* of a synchrochain. They participate to a level that they can certify all data passing through non-blockchain.

Beyond distributing data, circumchain nodes may perform certain protocol-level roles:; Some of these roles have restrictions and conditions associated with them:

1. Premier - performs the bulk of the security work. Must be a full node of the circumchain. Interacts with synchrochain aggregators, but need not participate in a synchrochain as a full node.

2. Delegator - fixholder who backs and selects premier candidates. This is done by modest collaborators who need not have any awareness of synchrochains.

Synchrochains may decide their own internal network structure, but are expected to communicate with Primal via the following roles:

1. Aggregator - collects and submits synchrochain data to the circumchain, subject to protocol rules. They are chosen as defined by the synchrochain, and must be full nodes of it.
2. Observer - performs additional security checks on the correct operation of the synchrochain, on behalf of the circumchain who gives reward. This role is assigned by delegator and reward incentivised, and must be a full node of the synchrochain.

Protocol

The circumchain protocol, including communication with synchrochains, works as follows.

1. For each synchrochain:
   (a) Aggregators watch the progress of the block-producing and consensus protocols, steps (2) and (5) respectively below, e.g. by participating in the circumchain as a full node. Based on what they think is the latest circumchain block that will most likely be finalised, they build on top of the latest synchrochain block (or other data) that would be finalised by it.
   (b) Aggregators sign data building on top of said latest synchrochain block, and submit it possibly indirectly, to the premiers assigned to their synchrochain, for inclusion in the circumchain. Ideally they submit a unique one, to help performance.
   (c) The synchrochain aggregators decide which synchrochain block to support, and presents relevant data of it as a synchrochain's next *candidate* to be added to the next circumchain block.
2. A block-producing aggregator collects candidates from all synchrochains, and puts this collection along with any recent circumchain extrinsics into a circumchain head block. For performance, this does not contain the full data from all synchrochains, but only metadata and partial data, including security-related metadata.

   In the negative case, this can result in forks, resolved later in step (5). This subprotocol is designed so that even with forks, participants have an idea of the block most likely to be finalised, similar to Proof-of-Work protocols.
3. A subprotocol is run to ensure that the full data is indeed available, including and distributing it to various other circumchain nodes.

4. Data submitted from a synchrochain might include indications that they are sending messages to another synchrochain, including metadata to facilitate this. This is now included on the circumchain head(s), so recipient synchrochains are aware of which new messages have been sent to them. They now collect the message bodies from the sending synchrochains. Premiers submit their votes on the block and finalise it, resolving any forks to a single head. These votes are added to the circumchain blocks.

Preliminaries

Here are the entities in the running of Primal, as well as security model for these entities.

Roles

Nodes which run Primal Network take up different roles and functions;

Premiers: A premier is the highest in charge and helps seal new blocks on the Primal Network. The Premier's role is dependent upon a sufficiently high bond being deposited and fixed for continual exchange among the committee of premiers. The premiers not in all cases own the entirety of the bond rather the delegators who bond with them. A premier must run a circumchain collaborator implementation with high availability and bandwidth. At each block the node must be set to accept the role of ratifying a new block on some synchrochain, and may be required to double check a few more. This process involves receiving, endorsing and republishing candidate blocks. The synchrochain assignment is random and changes frequently. Since the premier cannot reasonably be expected to maintain a fully-synchronised database of all synchrochains, producing a new synchrochain block will be delegated to a third-party, known as a aggregator. Once all new synchrochain blocks have been properly ratified by their appointed premier subgroups, premiers must then ratify the circumchain block itself. This involves updating the state of the transaction queues (essentially moving data from a synchrochain's output queue to another synchrochain's input queue), processing the transactions of the ratified circumchain transaction set and ratifying the final block, including the final synchrochain changes. A premier provably not fulfilling their role will be penalised i.e. their bond will either be deducted or be taken completely.

Delegators: A delegator is a fix-holding party who contributes to the security bond of a premier. They have no additional role except to place risk capital and as such to signal that they trust a particular premier to act responsibly in their maintenance of the network. They receive a pro-rata increase or reduction in their deposit according to the bond's growth to which they contribute.

Aggregators: A transaction aggregator assist premiers in producing valid synchrochain blocks. They maintain a "full-node" for a particular synchrochain; meaning that they retain all necessary information to be able to author new blocks and execute transactions in much the same way as block producers do on current blockchains. Under normal circumstances, they will aggregate and enact transactions to create an unsealed block, and provide it, together with a proof of validity, to one or more premiers who are responsible for proposing a synchrochain block.

Observers: An observer is not in the business of blockauthoring process, but are independent "bounty hunters" motivated by a large one-off reward. Precisely due to the existence of observers, we expect situations of misbehaviour to seldom happen, which could be of any other kind rather than just malicious acts, may be carelessness with secret key security. The are expected to frequently seek reward by carefully screening and observing rapidly ratified invalid synchrochain blocks. Their rewards are minimal, thus, further rewarded through a timely proof that at least one bonded party acted illegally.

Observers share similarities with "full nodes" in today's blockchain systems that the resources needed are relatively small and the commitment of stable uptime and bandwidth is not necessary. Observers differ in so much as their bond is regarded to be small compared to the high bonds that got premiers elected. This bond just like that of premiers, which consists of fix and exchange fixes prevents sybil attacks from wasting premiers' time and compute resources. Observers are part of the security model of Primal. Hence, their roles are taken seriously; they are chosen by delegators as they are regarded as potential premiers who didn't have enough bond.

The structural elements and different roles of the Primal protocol are, thus, in an example with thirty-two synchrochains with four each adjacently and inversely proportionate to each other, 160 premiers, and 8 aggregators per synchrochain. Note; there are 5 premiers to a synchrochain, but aggregators is subject or determined by the synchrochain. The bridge is a sub-protocol that allows external independent chains to interoperate with Primal.

Deterrence

Roles: We are aware that honesty can be a luxury, thus expect dishonesty among the premiers,, however we know that the good eggs follow the protocol while malicious ones can follow any arbitrary algorithm. We assume that 75% of delegators' fix belong to honest ones. Safe to say, we assume more than two third of premiers chosen by aggregators are honest. To deter misdemeanours and mischievous acts, we demand alot to be fixed to qualify as a delegator which is the first role required to ever be chosen as either a premier and/or an observer. Adding to that, a premier needs to further fix 25% more of the current fix which will be bonded with by the delegators to formulate votes and credibility. The delegators are only required to fix and bond with premier candidates; premiers fix, bond additional 25% of current fix with delegators and another 25% of current fix to be once again bonded with delegators for the purpose of exchange among the committee of premiers; observers likewise premiers fix, bond and have additional bond for the purpose of exchange, but among observers alike. In addition to that, delegators, observers and premiers are rotated hourly with polls taking place every 35 minutes for 10 minutes in expectations and preparations for the next committee (comprising of delegators, observers and premiers).

Synchrochains: We are not oblivious of the chances that there could be breach of security on block production mechanism for synchrochains. On one hand, we assume that the processes of being an aggregator are tedious, thus, significant amount of aggregators are honest; while on the other hand, we provide extra measures to ascertain their credibility, we demand 25% pof the delegators fix from them and attach observers with them. We do not depend the security of Primal on honesty, rather results as proof of good bbehaviours.Where maliciousmembers of aggregators are found, they forfeit their fix with us and are dismissed along with their blocks.

Keys: We assume that malicious parties generate their keys with an arbitrary algorithm while honest ones always generate their keys securely. However, keys undergo endorsement.

Network and Communication: All premiers have their own local clock and their clocks do not rely on any central clock. We assume that premiers and aggregators are in a partially synchronous network. It means that a message sent by a premier or a aggregator arrives at all parties in the network at the same units of time with none receiving later. Therefore, we assume an parallel delivery of messages in Primal. Further, aggregators and observers can connect to the circumchain network to submit their reports.

Components and Sub-Protocols

Primal's premiers are selected by the Delegated Proof-of-Exchange "DPoE" consensus. Delegated Proof-of-Exchange is adapted from Proof-of-Fix "PoS" where an unlimited amount of token holders can participate as delegators, backing with their fix a large but limited set of premiers. This paradigm simultaneously achieves high levels of security and scalability, as well as an unprecedented level of decentralisation by ensuring a property known in voting theory as proportional justified representation. Delegators, who are economically vested in the security of the system, act as legislators over the premiers performance; while observers who lose out in selection for premier act as judicial after being appointed by delegators. Based on the delegators' expressed preferences over candidates, every era the system selects a set of premiers with fix backings that are as high as necessary. Delegators are also economically disincentivised with their constant rotation to strengthen the security of the system and deter harmful behaviours; there can be no same premier. To avoid delegators from concentrating their votes on too few premiers, delegators are mixed, rotated and have their shifts irregular with no same set of delegators at no time or least at the barest minimal possibility. This helps keep the system decentralised. Furthermore, the election mechanism is highly adaptive to sudden changes, such as some premiers being dismissed after a slashing or complete cut of asset, as it automatically redistributes the delegators' backings across the new set of premiers, even when the votes themselves do not change.

Part of the security goal of Primal is to be Byzantine fault tolerant when the participants are rational. It is thought that with the features DPoE gives, the fixholders elect a set of premiers that has a more than 2/3 fraction of honest members.

The elected premiers are responsible for the circumchain activities; each synchrochain's aggregators are responsible for producing synchrochain blocks, the premiers are divided into rotating subsets, one for each synchrochain, and need to attest to the validity of synchrochain blocks before the headers of those blocks are included in the circumchain.

For maximum scalability the number of premiers are restricted to 5 in each of these subsets. The assurance of this is vested in the guarantees of DPoE that every premier is well backed, the availability and validity scheme can ensure that any attack on the validity of Primal is highly expensive in expectation; safe to say, the entirety of Primal's economic security backs every synchrochain. This is in stark contrast to having, say, 100 independent blockchains with an equivalent sum total of fix, where on average each blockchain is backed by 1/100-th of the fix, and thus only benefits from 1/100-th the level of security. We guarantee availability by using erasure coding of each synchrochain block to make the premiers collectively and robustly responsible for the availability of these blocks without breaking scalability.

The functionality depends on the certainty of all synchrochains to be valid, thus reverting the chain till the probability is favourable or confirmed. Therefore, reorganising the chain is a necessity for the chain to be capable of forking. Thus, we use a block production mechanism, Blind Assignment for Blockchain Extension "BABE" for premiers, which has similar properties to proof-of-work chains. Specifically, we can use the longest chain rule as part of our consensus, and the next block producer is not known in advance. On its own BABE would require us to wait a long time from the moment a block is produced to the moment it is finalised, i.e. when we can be confident that with high probability the block will never be reverted. This leads to slow finality; to avoid such we also adopt in accordance to Polkadot's already adapted mechanism GHOST-Based Recursive Ancestor Deriving Prefix Agreement "GRANDPA". GRANDPA is the finality gadget premiers use to finalise blocks. GRANDPA is separated from block production. This separation makes it very adaptive and here allows us to delay finalising blocks until challenges are dealt with, without slowing down block production. GRANDPA gets Byzantine agreement on finalised blocks and will allow us to prove to an entity that keeps track of the premier set which blocks are finalised, which will be important for bridges.

If an account on one synchrochain sends tokens to another synchrochain, then Cross-Chain Message Passing "XCMP" ensures that this message is delivered correctly. It is sent at a speed which is not dependent on how long it takes to finalise blocks, which means that it needs to deal with the possibility of Primal forking. Thus, we optimistically execute based on the thoughts that the synchrochain blocks are valid. If one is not, then we need to revert and for that, it is important that synchrochains only receive messages that were sent by blocks recorded on this new circumchain fork, and not the reverted fork. Thus, we need that the synchrochain and XCMP logic ensure that a fork of the circumchain defines a consistent history of Primal and thus, messages only arrive when they have been sent previously in the history defined by this fork.

If the token transfer is carried out in conjunction with Shared Protected Runtime Execution Enclaves “SPREE” modules then that ensures that so long as the synchrochains execute correctly, tokens can only be created and destroyed in an agreed upon way. In turn the correct execution of the chains code is guaranteed by the availability and validity scheme. SPREE ensures that shared code needed for the token transfer logic is correct as well. Even though chains can change their own code, they will not be able to change the code of SPREE modules. Instead the code of SPREE modules is stored centrally and the execution of that code and its storage will be sandboxed from the rest of the state transition. This ensures that this token transfer message is interpreted correctly and obtains the guarantees about tokens we want.

On the side of economics, we aim to Foster and have a constant yearly deflation rate. As stated before, it is important for the security of the system that all premiers have large amounts of fix backing them. Our adaptive reward schedule for premiers, observers and the aggregators backing them ensures that overall participation in DPoE stays high, and that the premiers’ fix backings are accordingly. On a more granular level, we pay or slash premiers on a perexecuted-action basis, and extend the same rewards or punishment onto delegators proportionally, to ensure that the rational strategy is compatible with honest behaviour.

The circumchain’s logic itself will need updating occasionally. The governance mechanism allows Primal token holders to participate in the decision-making process rather than having any changes to the system be imposed by a central authority – or in the case of some decentralised systems, by a team of developers. Too often, a contentious code change has led existing blockchains to an impasse or a permanent fork. We want a mechanism that balances being able to make uncontentious changes quickly when needed, while also providing the tools to deal with contentious proposals in a decisive and fair manner. The ultimate arbiters of Primal are the Prime token holders and so all important decisions, such as code changes, are made by state-weighted referenda. There is an elected council, responsible for making smaller decisions and partially setting the priority for referenda, in such a way that they cannot block a change that a majority wants.

Delegated Proof-of-Exchange and Delegator Election

Primal will have a native token called Prime. It will use Delegated Proof-of-Exchange (DPoE), our very own version of nominated proof-of-fix (NPoS). Consensus protocols with deterministic finality, such as the one in Primal, require a set of registered premiers of bounded size. Primal will maintain a number large number of delegators, observers and premiers, in the thousands. This number will be uultimately evaluated and decided by governance, and is intended to grow linearly with the number of synchrochains; yet it will be independent of the number of users in the network, thus ensuring scalability. However, DPoE allows for an unlimited number of Prime holders to participate as *delegators*, who help maintain high levels of security by putting more value at fix. As such, DPoE is not only much more efficient than proof-of-work (PoW), but also considerably more secure than conventional forms of PoS such as DPoS and BPoS. Furthermore, we introduce new guarantees on decentralisation hitherto unmatched by any other PoS-based blockchain.

A new set of premiers is elected at the beginning of every *era* – an hourly period, thus, 24 eras daily – to serve for that era, according to the delegators' preferences. More precisely, any Prime holder may choose to become a premier candidate or a delegator. Each candidate indicates the amount of fix he is willing to fix and his desired commission fee for operational costs. In turn, each delegator locks specified fix and publishes a list with any number of candidates that s/he trusts. Then a public protocol takes these lists as input and elects the candidates with the most backing to serve as premiers for the next era.

In addition to the premiers candidacy, a delegator who wishes to be premier must fix a further 25% of current fix value and bond with delegators who wish to elect s/he as premier, such bonded fix will be used for exchange among the committee of elected premiers. Therefore, in the place of slashing or complete loss of asset, the fixed tokens, additional 25% fixed and bonded tokens for election and the further 25% fixed and bonded tokens for exchange will be affected.

Delegators share the rewards, or eventual slashings, with the premiers they delegated on a per-fixed-prime basis. Delegators are thus economically incentivised to act as legislators for the system, and they will base their preferences on parameters such as premiers' fixing levels, commission fees, past performance, and security practices. Our scheme allows for the system to elect premiers with massive amounts of aggregate fix - much higher than any single party's Prime holdings - and thus helps turn the premier election process into a democracy. In fact, at any given moment we expect there to be a considerable fraction of all the Prime supply be fixed in DPoE. This makes it very difficult for an adversarial entity to get premiers elected since it either needs a large amount of Prime or high enough reputation to get the required delegators' backing, as well as being very costly to attack because it is liable to lose all of its fix and its earned reputation.

Primal elects premiers via a decentralised protocol with carefully selected, simple and publicly known rules, taking the delegators' lists of trusted candidates as input. Formally, the protocol solves a multi-winner election problem based on approval ballots, where delegators have voting power proportional to their fix, and where the goals are decentralisation and security.

Decentralisation: Our decentralisation objective translates into the classical notion of proportional representation in voting theory. That is, a committee should represent each minority in the electorate proportional to their aggregate vote strength (in this case, their fix), with no minority being under-represented. We highlight here that delegators – and their lists of trusted candidates – constitute a valuable gauge for the preferences of the general community, and that diverse preferences and factions will naturally arise not only due to economical and security-related reasons, but also political, geographical, etc. Such diversity of points of view is expected and welcome in a decentralised community, and it is important to engage all minorities in decision-making processes to ensure user satisfaction.

Circumchain State Machine

Primal, hence, is a replicated sharded state machine where shards are the synchrochains and the Primal circumchain is part of the protocol ensuring global consensus among all the synchrochains. Therefore, the Primal circumchain protocol, can itself be considered as a replicated state machine on its own. In this sense, this section describes the circumchain protocol by specifying the state machine governing the circumchain. To that end, we describe the circumchain state and the detail of state transition governed by transactions grouped in the circumchain blocks.

State: The state is represented through the use of an *associative array* data structure composed by a collection of (*key,value*) pairs where each key is unique. There is no assumption on the format of the key or the value stored under it besides the fact that they both the key and the value need to be finite byte arrays.

The (*key,value*) pairs which comprise the circumchain state are arranged in a Merkle radix-16 tree. The root of this tree canonically identifies the current state of the circumchain. The Merkle tree also provides an efficient mean to produce the proof of inclusion for an individual pair in the state.

To keep the state size in control, the circumchain state is solely used to facilitate the circumchain operations such as fixing and identifying premiers. The Merkle Radix tree is not supposed to store any information regarding the internal operation of the synchrochains.

State transition: Like any transaction-based transition system, Primal state changes via an executing ordered set of instructions, known as extrinsics. These extrinsics include transactions submitted by the public. They cover any data provided from “outside” of the machine’s state which can affect state transition. Primal circumchain is divided into two major components, namely the “Runtime” and the “Host”. The execution logic of the state-transition function is mainly encapsulated in the Runtime while all other generic operations, commonly shared among modern blockchain-based replicated state machines, are embedded into the Host. In particular, the latter is in charge of network communication, block production and consensus engines.

Runtime functions are compiled into a WebAssembly module and are stored as part of the state. The Host communicates the extrinsics to the Runtime and interacts with it to execute the state transition. In this way, the state transition logic itself can be upgraded as a part of the state transition.

Extrinsics: Extrinsics are the input data supplied to the Primal circumchain state machine to transition to new states. Extrinsics need to be stored into blocks of the circumchain in order to achieve consensus among the state machine replica. Extrinsics are divided into two broad categories namely: transactions and ”inherents” which represent data that is inherent to a circumchain block. The timestamp *t* of a block is an example of inherent extrinsics which must be included in each Primal circumchain block.

Transactions are signed and are shared on the network between nodes. In contrast, inherents are not signed and are not shared individually but rather only when they are included in a block. The inherents in a block are assumed to be valid if a supermajority of premiers assume so. Transactions on the circumchain are mainly concerned with the operation of the circumchain and Primal protocol as a whole, such as set_code, transfer, bond, confirm, delegate, vote.

Circumchain block producers listen to all transaction network messages. Upon receiving a transaction message, the transaction(s) are confirmed by the Runtime. The valid transactions then are arranged in a queue based on their priority and dependency and are considered for inclusion in future blocks accordingly.

Circumchain block format: A typical circumchain block consists of a header and a body. The body simply consists of a list of extrinsics.

The header contains the *hash of parent block*, *block number*, the *root of the state tree*, the *root of the Merkle tree* resulting from arranging the extrinsics in such a tree and the *digest*. The digest stores auxiliary information from the consensus engines which are required to confirm the block and its origin as well as information helping modest collaborators to confirm the block without having access to the state storage.

Circumchain block building: Here is a summary of various steps of circumchain operation which are carried out by its premiers. A priori, each premier privately knows the times during which it is supposed to produce a block.

Meanwhile, transactions ranging from the endorsed synchrochain block hash, transfer, fixing, delegation or slashing or complete cut for protocol violation are submitted to the circumchain premiers. The premiers examine the authenticity of the transactions and store them in their transaction pool. Once the time slot during which the premier is expected to produce the block has arrived, the premier estimates the block which most likely represents the state which is going to be finalised by the finality protocol and set it as the current state of the circumchain. Then the premier selects valid transactions with from the transaction pool, executes them and updates the state accordingly. The premier executes and collates as many transactions as the block capacity allows and attaches a cryptographic digest of the final stage of the circumchain after executing the selected transactions. Finally the premier signs and publishes the built block.

Upon receiving the new block, other premiers examine the producer's adherence to the protocol as well as the validity of included transactions and store the block in the *block tree* which represents all possible candidates for a final state transition of the circumchain.

Simultaneously, the set of premiers votes on various branches of the block tree (see 4.3.2) and prunes branches which conflict with the version agreed upon by the supermajority of the premiers. In that way, they eventually agree on a canonical state of the circumchain.

Consensus

The consensus for transaction finality will be that of Polkadot's hybrid protocol BABE and GRANDPA. BABE is a block production mechanism that will run in the circumchain to provide probabilistic finality and GRANDPA which provides provable and deterministic finality that works independently from BABE. Informally probabilistic finality implies that after certain time passes, a block in the circumchain will be finalised with very high probability and deterministic finality implies a finalised block stays final forever. Furthermore provable finality means that we can prove to parties not actively involved in the consensus that a block is final.

Provable finality is required for simplicity to make bridges to external independent chains; another blockchain, not part of Primal's consensus could then be convinced of when it is safe to act on data in a circumchain or synchrochain block without any danger of it being reverted. The best way of getting that is to have Byzantine agreement among the premiers on the state of Primal and its synchrochains. However, the availability and validity scheme may also require us to revert blocks, which would mean that getting Byzantine agreement on every block, as in Tendermint or Algorand, would not be suitable. However, this should happen rarely as a lot of fix will be slashed when we do this. As a result, we want a scheme that generates blocks and optimistically executes them, but it may take some time to finalise them. Thus, GRANDPA voters need to wait for assurances of availability and validity of a block before voting to finalise that block. Even the speed at which we finalise blocks may vary - if we do not receive reports of invalidity and unavailability then we can finalise fast, but if we do then we may need to delay finality while we execute more involved checks.

Because of the way Primal's messaging protocol works, message passing speed is constrained by block time, but not by finality time. Thus if we delay finality but in the end do not revert, then message passing is still fast.

As a result of these requirements, we have chosen to separate the mechanisms for block production and finalising blocks as much as possible.

Blind Assignment for Blockchain Extension (BABE)

We use BABE protocol to produce circumchain blocks; BABE assigns premiers randomly to block production slots using the randomness generated with blocks. A block production slot is a division of time when a block producer may produce a block. The time, thus, is arbitrary and these assignments are completely private until the assigned premiers produce their blocks, hence the name.

There could be slots without any assignment which are called empty slot. In order to fill the empty slots, we have a secondary block production mechanism based on Aura that assigns premiers to slots publicly. We note that these blocks do not contribute to the security of BABE since the best chain selection and the random number generation algorithms work as if Aura blocks do not exist. Therefore, next we only describe BABE together with its security properties.

BABE has time division called *periods*, where each period consists of a number of sequential block production slots. Premiers are aware if the slots a block is supposed to be produced at the beginning of every period. When the time for its slot comes, the premier produces the block by proving that it is assigned to this slot.

BABE is based on the cryptographic primitive called verifiable random function (VRF). A premier in an period does the following to learn if it is eligible to produce a block:

1. it obtains the randomness in the genesis block if $m = 1$ or $m = 2$. Otherwise, it obtains the randomness generated two periods before.
2. it runs the VRF with its secret key and the input: randomness and the slot number.

If the output of VRF is less than the threshold τ, then the premier is the slot leader meaning that it is eligible to produce a block for this slot. We select τ with respect to security requirements of BABE e.g., bigger τ makes less probable to select only honest premiers for a slot than smaller τ. When a premier produces a block, it adds the output of the VRF and its proof to the block which shows that its VRF output is less than τ in order to convince other premiers that it has a right to produce a block in the corresponding slot. The premiers always generate their blocks on top of the best chain. The best chain selection rule in BABE says that ignore the Aura blocks and select the longest chain that includes the last finalised GRANDPA block.

Relative Time Protocol: The elected premiers for a slot need to know when the right time is to produce a block for the consistency and the security of BABE. For this, premiers use their local computer clock which is not adjusted by any centralised clock adjustment protocols such as the Network Time Protocol. Instead, they keep their clock synchronised with the other premiers with the relative time protocol. The formal security model of local clock synchronisation in blockchains without NTP and further details about the relative time protocol can be found in BABE, we assume that after the genesis block is released, elected premiers of the first period store the arrival time of the genesis block with respect to their local clock. Then, they mark the beginning time of the first slot and increment the slot number every T seconds. After this point, they periodically run the relative algorithm not to lose the synchronisation with others because of their local clock drifts. In addition to this, a premier who joins after the genesis block runs the relative time algorithm to be synchronised with the other premiers.
In every sync-periods (different than periods in BABE), premiers update their clock according to the result of the relative time protocol and use the new clock until the next sync-period. The first sync-period ε_1 starts just after the genesis block is released. Further, each premier stores the arrival time t_j of blocks together with the slot number sl'_j in the block during a sync-period. At the end of a sync-period, the premier retrieves the arrival time of probabilistically finalised blocks generated during the sync-period and computes some candidate start times of the first slot sl of the next sync-period i.e,

given that $a_j = T(sl - sl'_j)$, $C_T = \{t_j + a_j\}$. The times in C_T are considered as candidates. In order to choose one candidate, the premier then sorts the list of candidates C_T and outputs the median of the sorted list as a start time of the *sl*.

Overview of BABE Security:

- *Common Prefix (CP):* It ensures that the blocks which are *k*-blocks before the last block of an honest premier's blockchain cannot be changed. We call all unchangeable blocks *finalised* blocks. BABE satisfies CP property thanks to the honest super majority since malicious premiers are selected for a slot probabilistically much less than the honest premiers. It means that malicious premiers do not have enough to construct another chain which does not include one of the finalised blocks.

- *Chain Quality (CQ):* It ensures a minimum honest block contribution to any best chain owned by an honest party in every certain number of slots. We guarantee even in the worst case where a network delay is maximum that there will be at least one honest block in the best chain during an period so that the randomness cannot be biased.

- *Chain Growth (CG):* It guarantees a minimum growth between slots. Thanks to super majority of honest premiers, malicious premiers cannot prevent the growth of the best chain.

- *Chain Density (CD):* It ensures that in a sufficiently long portion of the best chain more than half of the blocks produced by honest premiers. CQ and CG properties imply this property.

Further details about BABE and its security analysis can be found in.

GRANDPA

As mentioned above, we want a finalisation mechanism that is flexible and separated from block production, which is achieved by GRANDPA. The only modification to BABE required for it to work with GRANDPA is to change the fork-choice rule: instead of building on the longest chain, a premier producing a block should build on the longest chain including all blocks that it sees as finalised. GRANDPA can work with many different block production mechanisms and it will be possible to switch out BABE with another.

Intuitively GRANDPA is a Byzantine agreement protocol that works to agree on a chain, out of many possible forks, by following some simpler fork choice rule, which together with the block production mechanism would give probabilistic finality if GRANDPA itself stopped finalising blocks. We want to be able to agree on many blocks at once, in contrast to single-block Byzantine agreement protocols.

We thought that we can ask the fork choice rule for the best block given a particular block. The basic idea is that we want to reach Byzantine agreement on the prefix of the chain that everyone agrees on. To make this more robust, we try to agree on the prefix of the chain that 2/3 of premiers agree on.

Votes rules are determined by Greedy Heaviest Observed Subtree (GHOST). We use this rule inside what is structured like a more traditional Byzantine agreement protocol, to process votes. The 2/3 GHOST rule works as follows. We have a set of votes, given by block hashes in which honest premiers should not have more than one vote, and we take the head of the chain formed inductively as follows. We start with the genesis block and then include the child of that block that 2/3 of the voters voted for descendants of, as long as there is exactly one such child. The head of this chain is $g(V)$ where V is the set of votes. There are two voting phases in a round of GRANDPA: prevote and precommit. Firstly premiers prevote on a best chain. Then they apply the 2/3-GHOST rule, g, to the set of prevotes V they see and precommit to $g(V)$. Then similarly they take the set of precommits C they see and finalise $g(C)$.

To ensure safety, we ensure that all votes are descendants of any block that could possibly have been finalised in the last round. Nodes maintain an estimate of the last block that could have been finalised in a round, which is calculated from the prevotes and precommits. Before starting a new round, a node waits until it sees enough precommits for it to be sure that no block on a different chain or later on the same chain as this round's estimate can be finalised. Then it ensures that it only prevotes and precommits in the next round to blocks that are descendants of the last round's estimate which it keeps updating by listening to precommits from the last round. This ensures safety.

To ensure liveness, we select one premier in rotation to be the primary. They start the round by broadcasting their estimate for the last round. Then when premiers prevote, if the primary's block passes two checks, that it is at least the premier's estimate and that it got $> 2/3$ prevotes for it and its descendants in the last round, then it prevotes for the best chain including the primary's block. The idea here is that if the primary's block has not been finalised, then progress is made by finalising the block. If the primary's block has not been finalised and all premiers agree on the best chain including the last finalised block, which we should do eventually because BABE gives probabilistic finality on its own, then we now make progress by finalising that chain.

Synchrochains

Block Production

An aggregator produces a synchrochain block, sends it to the synchrochain premiers, who sign its header as valid, and the header with enough signatures is placed on the circumchain. At this point, the synchrochain block is as canonical as the circumchain block its header appeared in. If this circumchain block is in the best chain according to BABE, so is the synchrochain block and when this circumchain block is finalised, so is the synchrochain block; because the synchrochain premiers switch synchrochains frequently, they are stateless collaborators of the synchrochain. Thus we distinguish between the synchrochain

block *B*, which is normally enough for full nodes of the synchrochain such as aggregators to update the synchrochain state, and the *Proof of Certificate(PoC)* block B_{PoC}, which a premier who does not have the synchrochain state can certify.

Any premier should be able to certify B_{PoC} given the circumchain state using the synchrochain's *state transition validation function* (STVF), the Web assembly code for which is stored on the circumchain in a similar way to the circumchain's runtime. The STVF takes as an input the PoC block, the header of the last synchrochain block from this synchrochain and a small amount of data from the circumchain state.

The STVF outputs the validity of the block, the header of this block and its outgoing messages. The PoC block contains any outgoing messages and the synchrochain block *B*. The synchrochain premiers should share the synchrochain block to the synchrochain network, as a back up to the aggregator itself doing so.

The PoC block will be the synchrochain block, its outgoing messages, its header and modest collaborators proof witnesses. These witnesses are Merkle proofs that give all elements of the input and output state that are used or modified by the state transition from the input and output state roots.

To aid in censorship resistance, a synchrochain may want to use proof of work or proof of fix to select aggregators, where the selection strategy is up to the given synchrochain. This can be implemented in the STVF and need not be a part of the Primal protocol. So for proof of work, the STVF would check that the hash of the block is sufficiently small. However, for speed, it would be useful to ensure that most circumchain blocks can include a synchrochain block. For PoW, this would necessitate it being probable that multiple aggregators are allowed to produce a block. As such we will still need a tie-breaker for the synchrochain premiers to coordinate on validating the same synchrochain block first. For proof of fix this may not be necessary.

Optionally, for some synchrochains, the synchrochain block *B* may not be enough for aggregators to update their state. This may happen for chains that use succinct zero-knowledge proofs to update their state, or even for permissioned chains that just give signatures from authorities for validity. Such chains may have another notion of synchrochain block which is actually needed to update their state and must have their own scheme to guarantee the availability of this data.

Validity and Availability

Once a synchrochain block is created it is important that the *synchrochain blob* consisting of the PoC block and set of outgoing messages from the synchrochain is available for a while. The naive solution for this would be broadcasting/share the synchrochain blobs to all circumchain nodes, which is not a feasible option because there are many synchrochains and the PoC blocks may be big. We want to find an efficient solution to ensure PoC blocks from any recently created synchrochain blocks are available.

For a single chain, such as Bitcoin, as long as 51% of hash power is honest, not making block data available ensures that no honest miner builds on it so it will not be in the final chain. However, synchrochain consensus in Primal is determined by circumchain consensus. A synchrochain block is canonical when its header is in the circumchain. We have no guarantees that anyone other than the aggregator and synchrochain premiers have seen the PoC block. If these collude then the rest of the synchrochain network need not have the synchrochain block and then most aggregators cannot build a new block and this block's invalidity may not be discovered. We would like the consensus participants, here the premiers, to collectively guarantee the availability rather than relying on a few nodes.

To this end we designed an availability scheme that uses erasure coding to distribute the PoC block to all premiers. When any misbehaviour, particularly in relation to invalidity, is detected, the blob can be reconstructed from the distributed erasure coded pieces.

If a block is available then full nodes of the synchrochain, and any modest collaborators that has the PoC block, can check its validity. We have three-level of validity checks in Primal. The first validity check of a PoC block is executed by the corresponding synchrochain premiers. If they certify the PoC block then they sign and distribute the erasure codes of the blob, including the PoC block, to each premier. We rely on nodes acting as observers to report the invalidity of a blob as a second level of validity checking. They would need to back any claim with their own fix in Primes. We would assume that some aggregators will be observers, as they have a fix in continued validity of the chain and are already running full nodes, also to protect their fix in Prime. The third level of validity checking is executed by a few randomly and privately assigned premiers. We determine the number of premiers in the third level of validity checking considering the amount of invalidity reports given by observers and unavailability reports given by aggregators. If an invalid synchrochain block is detected, the premiers who signed for its validity are either slashed or complete cut. We wait for enough of these randomly assigned checkers to check the block before voting on it in GRANDPA. We also want to ensure that the block is available before selecting the randomly assigned premiers. This means that the synchrochain premiers have to commit running a high risk of being slashed or completely cut for a small probability of getting an invalid block finalised. This means that the expected cost of getting an invalid block into Primal is higher than the amount of fix backing a single synchrochain.

The security of our availability and validity scheme is based on the security of the GRANDPA finality gadget and the quality of randomness generated in each BABE period.

Security Exchange (SE)

Part of the security endeared in the synchrochain circumchain communication is the continuous exchange. The fix of premiers which is for the purpose of exchange as part of the eligibility requirements from a delegator for premier candidacy and also the additional bond of fix between the premier and delegators for exchange as well are raise in extent of the security of Primal. The two distinct fix for the purpose of exchange are locked in a mobile transition

throughout the course of the premiers terms. Provided they are honest and diligent, then they will benefit rewards for the exchange, but if they are malicious mth3n they suffer the loss of slash or complete cut of all their fixed tokens.

The locked-in fix for exchange will be among the premiers carried by nodes. There'll be five central nodes for the purpose of mobile messaging. The central nodes are mobile and interchange sporadically and periodically. One node will be at the centre of the circumchain, with four others continually exchanging messages with the central one. It's the central nodes that'll be responsible for the mobility of the fix. Example: there are 24 premiers in a circumchain; A to X have equal or irregular amount based on a particular base fixed for exchange, the fix will begin from A who might be at the north-east of the circumchain and move to E who is at south-west of the circumchain through the four nodes based on the delegation of transition by the central node. A node is only central for 1 second before it switches with another of the remaining four nodes, however, can delegate thousands of sporadic messages or exchanges within 1 second based on communication at hand.

The exchange works in this manner: the fix that moves from A when it reaches E, it will be visible in E's spy wallet, but not accessible, then move from E along with the locked-in fix of E to L adding to the A's fix. Also, when it reaches L, it will be visible in L's spy wallet but inaccessible and then moves along with L's locked-in fix to X, and so on continuously to all the premiers and continues the cycle throughout their terms as premiers. In the place of many, some or particular premier's loss, the fix will be split among the observer(s) that identified malicious act and the other honest premiers and their delegators (bonus reward for them).

Mobile Messaging (MM)

This is a security measure put in place to further raise the scalability of the circumchain in communication with the synchrochains. The nodes act as messengers carrying information in a sporadic, periodic and unpredictable manner. This also prevents predictable and unexpected attacks. The nodes interchange periodically. Beyond their interchange they are not permanent, they are shuffled and randomly picked from the pool of observers who are modest collaborators. There can only be five nodes per circumchain per term. Their transition is measured in nanosecond, thus capable of linking and bridging tens of thousands of messages within 1 second.

The messages that are carried here are interactions among aggregators and premiers, observers and premiers, observers and aggregators, and a-three-way communication. Also, when there's a communication from an independent blockchain, which requires a bridge, temporary premiers are involved, thus

having synchrothread. The temporary premiers will also be locked-in in the central communication for transparency, accountability and scalability. The, nodes also certify PoC blocks validity; because, the nodes are run by premiers as well. This becomes a surprise validity measure out of the blues.

Cross Chain Messaging Passing (XCMP)

XCMP is the protocol that synchrochains use to send messages to each other. It aims to guarantee four things: first that messages arrive quickly; second that messages from one synchrochain arrive to another in order; third that arriving messages were indeed sent in the finalised history of the sending chain; and fourth that recipients will receive messages fairly across senders, helping guarantee that senders never wait indefinitely for their messages to be seen.

There are two parts to XCMP. Metadata about outgoing messages for a synchrochain block are included on the circumchain and later this metadata is used to authenticate messages by the receiving synchrochain. The message bodies corresponding to this metadata need to be actually distributed from the senders to the recipients, together with a proof that the message body is actually associated with the relevant metadata. The details of distribution are covered as a networking protocol in Cross-chain messaging; the remainder is covered below.

The way circumchain blocks include headers of synchrochain blocks gives a synchronous notion of time for synchrochain blocks, just by circumchain block numbers. Additionally it allows us to authenticate messages as being sent in the history given by the circumchain i.e. it is impossible that one synchrochain sends a message, then reorganises the chain so that that message was not sent, but has been received. This holds even though the system may not have reached finality over whether the message was sent, because any circumchain provides a consistent history.

Because we require synchrochains to act on every message eventually, non-delivery of a single message can potentially stop a synchrochain from being able to build blocks. Consequently we need enough redundancy in our message delivery system. Any premiers who validate the PoC block should keep any outgoing messages from that block available for a day or so and all full nodes of the sending synchrochain also store the outgoing messages until they know they have been acted on.

To achieve consistency, when a source synchrochain *S* sends messages in a synchrochain block *B* to a destination synchrochain *D*, then we will need to authenticate these using the circumchain state, which is updated based on the synchrochain header *PH* corresponding to the synchrochain block *B* that was included in the circumchain. We need to limit the amount of data in headers like *PH*, in the circumchain state and also to limit what the circumchain needs to do for authentication when processing such synchrochain headers.

To this end, the synchrochain header *PH* contains a message root *M* of outgoing messages, as well as a bitfield indicating which other synchrochains were sent messages in this block. The message root *M* is the root of a Merkle tree of the head hash H_p of a hash chain for each synchrochain *p* that this block sends messages to. The hash chain with head H_D has the hash of all messages sent to *S* from *D*, not just in block *B* but ever sent from *S* to *D* in any block up

to block *B*. This allows many messages from *S* to *D* to be verified at once from *M*. However the messages themselves are passed, they should also be sent with the Merkle proof that allows nodes of the receiving synchrochain to authenticate that they were sent by a *B* whose header *PH* was in a particular circumchain block.

Synchrochains receive incoming messages in order. Internally synchrochains may defer or reorder acting on messages according to their own logic (possibly constrained by SPREE). However they must receive messages in the order determined by the consistent history given by the circumchain. A synchrochain *D* always receives messages sent by synchrochain blocks whose header was in earlier circumchain blocks first. When several such source synchrochains have a header in the circumchain block, the messages from these synchrochains are received in some predetermined order of synchrochains, either sequentially in order of increasing synchrochain id, or some shuffled version of this.

A synchrochain *D* receives all messages sent by one synchrochain *S* in one synchrochain block or none of them. A synchrochain header PH^0 of *D* contains a watermark. This watermark consists of a block number of a circumchain block *R* and a synchrochain id of a source synchrochain *S*. This indicates that *D* has recieved all messages sent by all chains before circumchain block *R*, and has acted on messages sent in block *R* from synchrochains up to and including *S* in the ordering.

The watermark must advance by at least one sending synchrochain in each of *D*'s synchrochain blocks, which means that the watermark's circumchain block number advances or it stays the same and we only advance the synchrochain. To produce a synchrochain block on synchrochain *D* which builds on a particular circumchain block *R*, a aggregator would need to look at which synchrochain headers were built between the circumchain block that the last synchrochain block of this chain built on. In addition, it needs the corresponding message data for each of those that indicated that they sent messages to *D*. Thus it can construct a PoC block so that the STVF can validate that all such messages were acted on. Since a synchrochain must accept all messages that are sent to it, we implement a method for synchrochains to make it illegal for another synchrochain to send it any messages that can be used in the case of spam occurring. When the synchrochain header of a synchrochain block that sends a message is included in a circumchain block, then any nodes connected to both the source and destination synchrochain networks should forward messages, together with their proofs, from sender to receiver. The circumchain should at least act as a back up: the receiving synchrochain premiers of *D* are connected to *D*'s synchrochain network and if they do not receive messages on it, then they can ask for them from the synchrochain premiers of the sending chain *S* at the time the message was sent.

Economics and Incentive Layer

Primal will have a native token called PRIME.

Fixing rewards and inflation

We start with a description of fixing rewards, i.e. payments to *fixers* – premiers and delegators – coming from the minting of new Primes. Unlike some other blockchain protocols, the amount of tokens in Primal will not be bounded by an absolute constant, but there will rather be a controlled yearly deflation rate. Our delegated proof-of-exchange protocol which is based on fix and exchange rewards strictly competitively. This helps to maintain high fixing and exchanging rates and high security levels. In our design, fixing rewards are the only mechanism that mints Primes.

Premiers and delegators fix Primes. They get paid roughly proportional to their fix, but can be slashed up to 100% in case of a misconduct. Even though they are actively engaged for only one era at a time, they can continue to be engaged for an unlimited number of eras. During this period their fix is locked, meaning it cannot be spent, and it remains locked for several weeks after their last active era, to keep fixers liable to slashing even if an offence is detected late.

Fixing rate, interest rate, inflation rate: Let the fixing rate be the total amount of Primes currently fixed by premiers and delegators, divided by the current total Prime supply. The fixers' average interest rate will be a function of the fixing rate: if the fixing rate dips below a certain target value selected by governance, the average interest rate is decreased, thus incentivising more participation in the exchange of the DPoE. For instance, a target fixing rate of 50% could be selected as a middle ground between security and liquidity. If the fixers' average yearly interest rate is then set to 20% at that level, we can expect the deflation rate to fluctuate. Hence, by setting targets for the fixing rate and fixers' interest rate, we also control the inflation rate. Following this principle, every era we adjust our estimate of the fixing rate, and use it to compute the total amount of Primes to be paid to fixers for that era.

Rewards across premier supports: Once the total payout for the current era is computed, we need to establish how it is distributed. The premier election protocol partitions the active fix into *premier supports*, where each premier support is composed of the full fix of one premier plus a fraction of the fix of its backing delegators, and this partition is made so as to make premier supports as high and evenly distributed as possible, hence ensuring security and decentralisation. A further incentive mechanism put in place to ensure decentralisation over time is paying premier supports according to their work rate, regardless of their fix.

In particular, we devise a point system in which premiers accumulate points for each payable action performed, and at the end of each era premier slots are rewarded proportional to their points. This ensures that premiers are always incentivised to maintain high performance and responsiveness. Payable actions in Primals include: a) endorsing a synchrochain block, b) producing a circumchain block in BABE, c) adding to a BABE block a reference to a previously unreferenced uncle block and d) producing an uncle block.

Rewards within a premier slot: As a delegator's fix is typically split among several premier supports, their payout in an era corresponds to the sum of their payouts relative to each of these supports. Within a premier support, the payment is as follows: First, the premier is paid a *commission fee*, which is an amount intended to cover its operational costs. Then, the remainder is shared among all fixers – both premier and delegators – proportional to their fix. Thus, the premier receives two separate rewards: a fee for running a node, and a payout for fixing. We remark that the commission fee is up to each premier to set, and must be publicly announced in advance. A higher fee translates to a higher total payout for the premier, and lower payouts to its delegators, so delegators will generally prefer to back premiers will lower fees, and the market regulates itself in this regard. Premiers who have built a strong reputation of reliability and performance will however be able to charge a higher commission fee, which is fair.

We finalise the section with some observations on the incentives that our payout scheme is expected to cause on fixers. First, as premiers are well remunerated and their number is limited, they have an incentive to ensure high backing levels from delegators to ensure getting elected, and thus they will value their reputation. Over time, we expect elections to be highly competitive and for elected premiers to have strong track records of performance and reliability and large fix backings. Second, even if payouts across different premier supports are independent of their fix, within a premier support each actor is paid proportional to their fix, so there is always an individual incentive to increase one's own fix. Finally, if a premier gains a particularly high level of backing, it can profit from it by either increasing its commission fee, which has the effect of raising its own reward at the risk of losing some nominations, or launching a new node as a premier candidate and splitting its backing among all its nodes. On this last point, we welcome operators with multiple premier nodes, and even aim to make their logistics simpler.

Circumchain block limits and transaction fees

Limits on resource usage: We bound the amount of transactions that a circumchain block can process, in order to a) ensure that each block can be processed efficiently even on less powerful nodes and avoid delays in block production, and b) have guaranteed availability for a certain amount of high-priority, operational transactions such as misconduct reports, even when there is high network traffic. In particular, we set block constraints on the following resources: on-chain byte-length, and time and memory required to process the transactions.

We classify transactions into several types, according to their priority level and resource consumption profile. For each of these types we have run tests based on worst-case scenarios for state, and for different input arguments. From these tests, we establish conservative estimates on resource usage for each transaction, and we use these estimates to ensure that all constraints on resource usage are observed.

We also add an extra constraint on resources: we distinguish between regular and high-priority transactions, and only let regular transactions account for up to 75% of each block resource limit. This is to ensure that each block has a guaranteed space for high-priority transactions of at least 25% of resources.

Transaction fees: We use the model described above to set the fee level of a transaction based on three parameters: its type, its on-chain length, and its expected resource usage. This fee differentiation is used to reflect the different costs that a transaction incurs on the network and on the state, and to encourage the processing of certain types of transactions over others. A fraction of every transaction fee is paid to the block producer, while another fraction goes to finance the Treasury. We highlight that, for a block producer, the rewards coming from transaction fees may constitute only a small fraction of their overall revenue, just enough to incentivise inclusion on the block.

We also run an adaptive transaction fee schedule that reacts to the traffic level, and ensures that blocks are typically far from full, so that peaks of activity can be dealt with effectively and long inclusion times are rare. In particular, the fee of each transaction is multiplied by a parameter that evolves over time depending on the current network traffic.

We make fees evolve slowly enough, so that the fee of any transaction can be predicted accurately within a frame of an hour. In particular, we do not intend for transaction fees to be the main source of income for fixrs.

Governance

Primal uses sophisticated mechanisms for Governance which allows it to evolve gracefully over time at the ultimate behest of its assembled fixholders. A key and unfailing rule is that all changes to the protocol must be agreed upon by fix-weighted referendum – the majority of fix always commands the network.

In order to make any changes to the network, the idea is to bring Prime holders together and administrate a network upgrade decision with the help of the Council. No matter whether the proposal is submitted by a Prime holder or by the Council, it will ultimately have to go through a referendum to let all Prime holders, weighted by fix, make the decision.

Each Prime holder in Primal has the right to: a) submit a proposal, b) endorse a public proposal to prioritise it in the referendum timetable, c) vote on all active referenda, d) become a candidate for a seat in the Council, and e) vote on candidates for the Council. In addition, any Prime holder may become a delegator or a premier candidate to participate in DPoE.

Proposals and Referenda

The core of the Primal logic is stored on-chain in an amorphous state-transition function and defined in a platform-neutral language: WebAssembly. Each proposal takes the form of a privileged function call in the runtime, that is able to modify the runtime code itself, achieving what would otherwise require a ”hard fork”. A proposal is then tabled and voted upon via referendum. Proposals can be started in one of several ways:

- a public proposal, which is submitted by any Prime holder;
- a Council proposal, submitted by the Council;
- a proposal submitted automatically as part of the enactment of a prior referendum, and
- an emergency proposal submitted by the Technical Committee.

Each proposal approved by referendum has an associated enactment delay, i.e. a time interval between the referendum ending and the changes being enacted. For the first two types of proposals above this is a fixed interval, tentatively set to 28 days. For the third type, it can be set as desired. Emergency proposals deal with major problems with the network which need to be fast-tracked, and hence will have a shorter enactment delay. Having an enactment delay ensures a level of stability, as it gives all parties sufficient notice to adapt to the new changes. After this period, the call to the associated privileged function is automatically made.

Any fixholder can submit a *public proposal* by depositing a fixed minimum amount of Primes, which stays locked for a certain period. If someone agrees with the proposal, they may deposit the same amount of tokens to endorse it. Public proposals are stored in a priority queue, and at regular intervals the proposal with the most endorsements gets tabled for a referendum. The locked tokens are released once the proposal is tabled.

Council proposals are submitted by the Council, and are stored in a separate priority queue where the priorities are set at the Council's discretion.

A **referendum** is a simple, inclusive, fixd-weighted voting scheme. It has a fixed voting period, after which votes are tallied. Referenda are always binary: voting options are ”aye”, ”nay”, or abstaining entirely.

Timetables: Every thirty days, a new proposal will be tabled and a referendum will come up for a vote. The proposal to be tabled is the top proposal from either the public-proposal queue or the Council-proposal queue, alternating between the two queues if both are non-empty. If both queues are empty, the slot is skipped in the referendum timetable. Multiple referenda cannot be active simultaneously, except for emergency referenda which follow a parallel timetable.

Vote counting: Voting on referenda is open to all Prime holders with a voting power proportional to their fix, up to a possible vote multiplier which is awarded to some parties depending on their level of commitment to the system, as we explain now. A party must generally lock their tokens used for voting until at least the enactment delay period beyond the end of the referendum. This is in order to ensure that some minimal economic buy-in to the result is

needed and to dissuade vote selling. It is possible to vote without locking at all, but in that case the voting power is a small fraction of a normal vote for the given fix. Conversely, Primal will offer *voluntary extended locking*, that allows any party to increase their voting power by extending the period of time they are willing to lock up their tokens. This ensures that voters committed to the system long term, who are willing to increase their exposure to the decision of a referendum, have a greater say in the matter.

Turnout biasing: It may seem restrictive to force a full fixholder-based process to do something as little as, say, nudging the block time down by 5%. However, without this rule the network would likely be unstable, as placing its control outside of the hands of fixholders would create a misalignment that may lead to inaction or worse. However, by taking advantage of the fact that turnout is rarely 100%, we can effect different outcomes depending on the circumstances, crafting a balance of power between active and passive fixholders. For example, simple voting systems typically introduce a notion of quorum, whereby a minimum amount of turnout must be reached before a change is passed.

For public proposals, we generalise this notion into a ”positive turnout bias”, where additional turnout always makes change more likely, assuming the same yay-to-nay ratio. More specifically, in case of low turnout we favour the nay side, or status quo, by requiring a super-majority approval, and as turnout approaches 100% the requirement dials down to majority-carries. This works on two principles: Firstly that the status quo tends to be safer than any change, and thus should have some bias towards it. Secondly that, like all means of empirical measurement, there is inevitably going to be some degree of inaccuracy and volatility over time, particularly when turnout is low – a result could be 51% − 49% one month and then change to 49% − 51%, and given the costs involved in enacting the changes of a proposal it is advantageous to ensure that a result would not likely flip shortly after enactment.

On the other hand, for proposals submitted by the Council, referenda have no turnout bias and majority-carries is observed. The reasoning here is that proposals pre-approved by the Council are deemed safer and less likely to be reverted, so the previously mentioned issues are alleviated and we can let Prime holders freely decide on the matter.

Finally, in the exceptional case that a Council proposal receives unanimous support by all Council members, it will observe a ”negative turnout bias”. This is the symmetric opposite of the first case, where additional turnout always makes change less likely, we favour the yay side in case of low turnout by requiring a super-majority of nays to reject the proposal, and as turnout approaches 100% the requirement dials down to majority-carries. See Figure 5.

The Council and the Technical Committee

The Council is an entity comprising a number of actors each represented by an on-chain account. Its goals are to represent passive fixholders, submit sensible and important proposals, and cancel uncontroversially dangerous or malicious proposals.

The Council will constantly review candidate proposals to deal with emerging issues in the system. A candidate proposal is officially backed by the Council – and enters the queue of Council proposals – only after it is approved by a strict majority of Council members, with no member exercising a veto. A candidate proposal can be vetoed only once; if, after a cool-down period, it is once more approved by a majority of Council members, it cannot be vetoed a second time.

As mentioned before, in the case that all members vote in favour, a Council proposal is considered uncontroversial and enjoys a special turnout bias that makes it more likely to be approved.

Finally, Council members may vote to cancel any proposal, regardless of who submitted it, but their vote must be unanimous. Since unanimity is a high requirement, it is expected that this measure will only be used when it is an entirely uncontroversial move. This may function as a last resort if there is an issue found late in the day with a referendum's proposal such as a bug in the code of the runtime or a vulnerability that the proposal would institute. If the cancellation is controversial enough that there is at least one dissenter, then it will be left to all Prime holders en masse to determine the fate of the proposal, with the registered Council cancellation votes serving as red flags so that voters pay special attention.

Electing Council members: There will be 37 seats in the Council of Primal. A general election for all seats will take place once per month. All Prime holders are free to register their candidacy for the Council, and free to vote for any number of candidates, with a voting power proportional to their fix. Much like the premier election problem in DPoE, this is a fixweighted, multi-winner election problem based on approval ballots. We can thus solve it using the same algorithm we use for DPoE which in particular offers the property of *proportional justified representation*; see Section 4.1 for more details. This property guarantees that the elected Council will represent as many minorities as possible, thus ensuring that Governance stay decentralised and resistant to capture. Council members can be re-elected indefinitely, provided their approval remains high enough.

The Technical Committee is composed according to a single vote for each team that has successfully and independently implemented or formally specified the protocol in Primal. Teams may be added or removed by a simple majority of the Council.

The Technical Committee is the last line of defence for the system. Its sole purpose is detecting present or imminent issues in the system such as bugs in the code or security vulnerabilities, and proposing and fast-tracking emergency referenda. An emergency proposal needs a simultaneous approval of at least three-quarters of the Council members and at least two-thirds of the Technical Committee members in order to be tabled. Once tabled, it is fast-tracked into a referendum that runs in parallel with the timetable of regular referenda, with a far shorter voting period, and a near-zero enactment period. The approval mechanics in this case are unchanged from what they would be otherwise, i.e. either a simple majority or, in the case of a unanimous Council approval, a turnout-based bias for approval.

We highlight that for practical reasons the Technical Committee is not democratically elected, but in contrast it has an extremely reduced scope of action and no power to act unilaterally, as explained in the lines above. This mechanism is expected to suffice for non-contentious bug fixes and technical upgrades, but given the requirements imposed, may not be effective in the case of emergencies that have a tinge of political sensitivity or strategic importance to them.

Allocation of Synchrochain slots

We use auctions to have a fair, transparent and permissionless synchrochain allocation procedure. Broadly speaking, parties interested in receiving a synchrochain slot participate in an auction with Prime-denominated bids. The party with the highest bid is declared as winner and is allocated a slot for a specific period of time, with its bid becoming a locked deposit that is released at the end of said period. The leasing cost of the slot thus corresponds to the opportunity cost of having this deposit locked. This Prime-denominated deposit also establishes the voting power of the synchrochain in Primal's governance.

Since implementing seal-bid auctions is difficult and in order to avoid bid sniping, we adopt a Torch auction mechanism with a retroactively determined close. Going into detail, we plan to have an auction every few weeks, where in each auction four contiguous four-month slots are offered for lease. A bid can be made for any combination of one, two, three or four contiguous slots, for a total of ten possible time periods lasting 4, 8, 12 or 16 months. Once the auction starts, parties can post bids as transactions for any one of these ten periods, within a fixed window of time lasting several hours. A party is allowed to submit multiple bids, where a bid is registered only if a) it beats the current highest bid for the corresponding period, and b) the party does not become the provisional winner of two or more periods with gaps in between. For example, the winner of period (1,2) – constituted of the first two slots – cannot bid on period (4) – the fourth slot – until someone else overbids the former period.

The stated goals of this design are to incentivise parties to bid early and avoid bid sniping, to give less funded projects a chance of winning a slot hence securing the decentralised nature of Primal, and to discourage griefing attacks by parties who raise the value of the winning bid with no intention of winning themselves.

Treasury

The Treasury raises funds continually. These funds are used to pay for developers that provide software updates, apply any changes decided by referenda, adjust parameters, and generally keep the system running smoothly. Funds may also be used for further goals such as marketing activities, community events and outreach. This is ultimately controlled by all Prime holders via Governance and it will be the community and their collective imagination and judgment which really determines the course of the Treasury.

Funds for Treasury are raised in two ways:

1. by channeling some of the premier rewards that come from minting of new tokens, and
2. by channeling a fraction of transaction fees and of slashings.

The first method allows us to maintain a fixed deflation rate while simultaneously having the premier rewards be dependent of the fixing level: the difference between the scheduled minted tokens and the premier rewards is assigned to Treasury in each era. We also argue that it is convenient to have a fraction of all slashings be redirected to Treasury: following an event that produced heavy fix slashing, the system is likely to need additional funds to develop software updates or new infrastructure that deal with an existing issue, or it might be decided by Governance to reimburse some of the slashed fix. Thus, it makes sense to have the slashed Primes available in Treasury, instead of burning them and having to mint more Primes soon thereafter.

Cryptography

In Primal, we necessarily distinguish among different permissions and functionalities with different keys and key types, respectively. We roughly categorise these into account keys with which users interact and session keys that nodes manage without operator intervention beyond a certification process.

Account keys

Account keys have an associated balance of which portions can be *locked* to play roles in fixing, resource rental, and governance, including waiting out a couple types of unlocking period. We allow several locks of varying duration, both because these roles impose different restrictions, and for multiple unlocking periods running concurrently.

We encourage active participation in all these roles, but they all require occasional signatures from accounts. At the same time, account keys have better physical security when kept in inconvenient locations, like safety deposit boxes, which makes signing arduous. We avoid this friction for users as follows.

Accounts that lock funds for fixing are called *pile accounts*. All pile accounts register a certificate on-chain that delegates all premier operation and nomination powers to some *controller account*, and also designates some *proxy key* for governance votes. In this state, the controller and proxy accounts can sign for the pile account in fixing and governance functions respectively, but not transfer funds.

At present, we support both pg211703 and pc211703 for account keys. These are both Schnorr-like signatures implemented using the PG211703 curve, so both offer extremely similar security. We recommend PG211703 keys for users who require Hardware Security Module (HSM) support or other external key

management solution, while pc211703 provides more blockchain-friendly functionality like Hierarchical Deterministic Key Derivation (HDKD) and multi-signatures.

In particular, pc211703 uses the Ristretto implementation of Mike Hamburg's Decaf [18, §7], which provide the 2-torsion free points of the pg211703 curve as a prime order group. Avoiding the cofactor like this means Ristretto makes implementing more complex protocols significantly safer. We employ Blake2b for most conventional hashing in Primal, but pc211703 itself uses STROBE128, which is based on Keccak-f(1600) and provides a hashing interface well suited to signatures and non-interactive zero-knowledge proofs (NIZKs).

Session keys

Session keys each fill roughly one particular role in consensus or security. As a rule, session keys gain authority only from a session certificate, signed by some controller key, that delegates appropriate fix.

At any time, the controller key can pause or revoke this session certificate and/or issue replacement with new session keys. All new session keys can be registered in advance, and most must be, so premiers can cleanly transition to new hardware by issuing session certificates that only become valid after some future session. We suggest using pause mechanism for emergency maintenance and using revocation if a session key might be compromised.

We prefer if session keys remain tied to one physical machine because doing so minimises the risk of accidental equivocation. We ask premier operators to issue session certificates using an RPC protocol, not to handle the session secret keys themselves.

Almost all early proof-of-stake networks have a negligent public key infrastructure that encourages duplicating session secret keys across machines, and thus reduces security and leads to pointless slashing.

We impose no prior restrictions on the cryptography employed by specific components or their associated session keys types.[1]

In BABE, premiers use pc211703 keys both for regular Peak signatures, as well as for a verifiable random function (VRF) based on NSEC5 [28].

A VRF is the public-key analog of a pseudo-random function (PRF), aka cryptographic hash function with a distinguished key, such as many MACs. We award block production slots when the block producer scores a low enough VRF output $\mathrm{VRF}_{sk}(r_e||\text{slotnumber})$, so anyone with the VRF public keys can certify that blocks were produced in the correct slot, but only the block producers know their slots in advance via their VRF secret key.

[1] We always implement cryptography for Primal in native code, not just because the runtime suffers from WASM's performance penalties, but because all of Primal's consensus protocols are partially implemented outside the runtime in Substrate modules.

As in [15], we provide a source of randomness r_e for the VRF inputs by hashing together all VRF outputs form the previous session, which requires that BABE keys be registered at least two full periods before being used.

We reduce VRF output malleability by hashing the signer's public key alongside the input, which dramatically improves security when used with HDKD. We also hash the VRF input and output together when providing output used elsewhere, which improves composability when used as a random oracle in security proofs. See the 2Hash-DH construction from Theorem 2 on page 32 in appendix C of [15].

In GRANDPA, premiers shall vote using BLS signatures, which supports convenient signature aggregation and select ZCash's BLS12-381 curve for performance. There is a risk that BLS12-381 might drop significantly below 128 bits of security, due to number field sieve advancements. If and when this happens, we expect upgrading GRANDPA to another curve to be straightforward.

We treat libp2p's transport keys roughly like session keys too, but they include the transport keys for sentry nodes, not just for the premier itself. As such, the operator interacts slightly more with these.

Networking

In the preceding sections we talk about nodes sending data to another node or other set of nodes, without being too specific on how this is achieved. We do this to simplify the model and to clearly delineate a separation of concerns between different layers.

Of course, in a real-world decentralised system the networking part also must be decentralised it's no good if all communication passes through a few central servers, even if the high-level protocol running on top of it is decentralised with respect to its entities. As a concrete example: in certain security models, including the traditional Byzantine fault-tolerant setting, nodes are modelled as possibly malicious but no consideration is given to malicious edges. A security requirement like "> 1/3 of nodes are honest" in the model, in fact translates to "> 1/3 of nodes are honest and can all communicate perfectly reliably with each other all the time" in reality. Conversely, if an edge is controlled by a malicious ISP in reality, it is the corresponding node(s) that must be treated as malicious in any analysis under the model. More significantly, if the underlying communications network is centralised, this can give the central parties the ability to "corrupt" > 1/3 of nodes within the model thereby breaking its security assumptions, even if they don't actually have arbitrary execution rights on that many nodes.

In this section we outline and enumerate the communication primitives that we require in Primal, and sketch a high-level design on how we achieve these in a decentralised way, with the specifics to be refined as we move forward with a production system.

Networking overview

As discussed above, Primal consists of a unique circumchain interacting with many different synchrochains and providing them with security services. These require the following network-level functionality, generally for distribution and availability:

1. As with all blockchain-like protocols, the circumchain requires:
   (a) accepting transactions from users and other external data (collectively known as extrinsic data or *extrinsics*), and distributing them
   (b) distributing artefacts of the collation subprotocol 4.2
   (c) distributing artefacts of the finalisation subprotocol 4.3.2
   (d) synchronising previously-finalised state

   As an important special case, synchrochains may choose to implement themselves according to the above structure, perhaps even re-using the same subprotocols. Part of the Primal implementation is architected as a separate library called substrate for them to do just this.
2. For interaction between the circumchain and the synchrochains, we require:
   (a) accepting synchrochain blocks from synchrochain aggregators
   (b) distributing synchrochain block metadata including validity attestations
   (c) distributing synchrochain block data and making these available for a time 4.4.2, for auditing purposes
3. For interaction between synchrochains, we require:
   (a) distributing messages between synchrochains 4.4.3, specifically from the relevant senders to the relevant recipients

For each of the above functionality requirements, we satisfy them with the following:

- 1(b), 1(c), 2(b) - artefacts are broadcast as-is (i.e. without further coding) via share.
- 1(a), 1(d), 2(a) - effectively, a set of nodes provide the same distributed service to clients. For accepting extrinsics or blocks, clients send these directly to the serving node; for synchronisation, clients receive verifiable data directly from the serving node.

- 2(c) - special case, below. Briefly, data is erasure-encoded so that different recipients receive a small part; pieces are sent directly via QUIC.
- 3(a) - special case, below. Briefly, messages are sent directly via QUIC; in the process outboxes are reassembled into inboxes and the latter can be transferred as a batch to recipients.

We go into these in more detail in the next few sections. Finally, we talk about the lower layers underpinning all of these subprotocols, namely Authentication, transport, and discovery.

Sharing

This subprotocol is used for most circumchain artefacts, where everyone needs to see more-or-less the same public information. Part of its structure is also used for when a node goes offline for a long time and needs to synchronise any newer data it hasn't seen before.

The Primal circumchain network forms a share overlay network on top of the physical communications network, as an efficient way to provide a decentralised broadcast medium. The network consists of a known number of trusted nodes (premiers) who have been permissioned via fixing, and an unknown number of untrusted nodes (full nodes that don't perform validation) from the permissionless open internet. (As an aside, recall that some of the untrusted nodes may have other roles as defined earlier, e.g. synchrochain aggregator, observers, etc.)

A simple push-based approach is implemented currently, with hash-based tracker caches to avoid sending duplicates to peers, and a few restrictions to avoid the most common spam attacks:

- Artefacts may only be received in dependency order; peers are not allowed to send them outof-order. Though this decreases network-level efficiency, it is straightforward to implement and provides a healthy level of security.
- To efficiently communicate to sending peers what they are allowed to send in dependency order, periodically peers update each other with their view of the latest heads of the chain.

There are also more specific constraint rules applied to artefacts belonging to the various higherlevel subprotocols using the share protocol, to avoid broadcasting obsolete or otherwise unneeded artefacts. For example, for GRANDPA we only allow two votes being received for each type of vote, round number, and voter; any further votes will be ignored. For block production only valid block producers are allowed to produce one block per round; any further blocks will be ignored.

There is basic support for *sentry nodes*, proxy servers that are essentially the only neighbour of a private server, running more security-critical operations such the premier role.

The network topology is a weak point currently; nodes connect to each other on an ad-hoc basis by performing random lookups in the address book. Further work will proceed along two fronts:

1. Trusted nodes will reserve a portion of their bandwidth and connection resources, to form a structured overlay with a deterministic but unpredictable topology that rotates every era. For nodes running behind sentries, this effectively means that their sentry nodes instead participate in this topology.
2. For the remainder of trusted nodes' resource capacity, and for the whole of untrusted nodes' resource capacity, they will select neighbours via a scheme based on latency measurements, with the details to be decided. Notably, for good security properties we want a scheme that does not simply choose "closest first", but also some far links as well.

In some sense, this can be viewed as the trusted nodes forming a core with the untrusted nodes around it - but note that trusted nodes are expected to use some of their resources to serve untrusted nodes as well. Both topologies are chosen to mitigate eclipse attacks, as well as sybil attacks in the permissionless untrusted case.

Further work will also include some sort of set reconciliation protocol, to further reduce redundancy when many senders attempt to send the same object to the same recipient at once; and potentially look into lifting the dependency-order restriction whilst retaining security.

Distributed service

This subprotocol is used when some part of Primal is providing a service to some external entity, namely 1(a) accepting circumchain transactions, 1(d) synchronising circumchain state, and 2(a) accepting collated blocks in the list above.

In our initial implementation, this simply involves looking up a particular target set in the address book, selecting a few nodes from this set, and connecting to them. For 1(a) and 1(d) the target set is the whole set of premiers, and for 2(a) the target set is the set of synchrochain premiers for the client aggregator's synchrochain. Both of these can be retrieved straightforwardly from the chain state, and indeed for 1(a) this is simply the same process as joining the share network.

Further work will consider the issue of load-balancing the transport-layer connections over the whole target set, as well as ensuring availability. This may require additional sophistication in the address book.

primal network

Storage and availability

This subprotocol addresses the networking used for the Availability and Validity subprotocol.

Recall that for scalability, Primal does not require everyone to store the state of the whole system, namely all of the state pointed to by all of the blocks. Instead, every synchrochain block is split into pieces by erasure-coding, such that there is 1 piece for every premier for a total of *N* pieces, the erasure threshold being *ceil*(*N*/3) blocks for security reasons explained elsewhere. All the pieces are available initially at the relevant synchrochain premiers, having been submitted by some of the aggregators. (In this role, the synchrochain premiers are also called the *preliminary checkers*.) The pieces are then selectively distributed in the following phases:

1. Distribution - every premier initially wants 1 of these pieces, and the synchrochain premiers must distribute them as such.
2. Retrieval - the *approval checkers* need to be convinced that *ceil*(*N*/3) premiers have their pieces, and some of them will attempt to actually retrieve these.
3. Further retrieval - yet later, and optionally, other non-premier parties might also want to perform further checks, e.g. in response to observers alerts, and again will want any *ceil*(*N*/3) of the pieces.

This subprotocol therefore aims to ensure that the threshold is available and can be retrieved from the relevant premiers for some reasonable amount of time, until at least the latter phases are complete. We will follow a bittorrent-like protocol with the following differences:

- With both distribution and retrieval, the set of recipients is known. Therefore, pieces can be pre-emptively pushed from premiers that already have the piece, in addition to bittorrent's pull semantics.
- Premiers behind sentry nodes will use these as proxies, rather than directly sending.
- Instead of a centralised tracker, tracker-like information such as who has what piece, is broadcast via the circumchain share network.

The preliminary checkers are expected to be fully- or mostly-connected; this is a pre-existing requirement for the collation protocol as well. The approval checkers should also be fully- or mostly-connected, to help the retrieval process complete faster.

Beyond this, nodes may communicate to any other nodes as the protocol sees fit, similar to bittorrent. To protect against DoS attacks they should implement resource constraints as in bittorrent, and furthermore nodes should authenticate each other and only communicate with other premiers, including the preliminary and approval checkers. Non-premier parties in the latter optional phase will be supplied with an authentication token for this

purpose. In a separate more detailed document, we propose a scheme for load-balancing, such that nodes do not accidentally overwhelm other nodes in their random choices.

There are various reasons why we do not consider it very suitable, to use a structured overlay topology for this component:

1. Each piece is sent to specific people, rather than everyone.
2. (a) People that want a specific piece of data, know where to get it - i.e. premiers, for their own piece, i.e. the preliminary checkers.
   (b) Other people want non-specific pieces - i.e. approval checkers, want any 1/3 of all pieces to be able to reconstruct.

Overlay topologies are generally more useful for the exact opposite of the above usage requirements:

1. Each data piece is sent to nearly everyone, or
2. People want a specific data piece, but don't know where to get it from.

For example, bittorrent has similar requirements and does not use a structured overlay - the peers there connect to other peers on a by-need basis.

Cross-chain messaging

This subprotocol address the networking scheme used for the XCMP messaging sub-protocol described in Section 4.4.3.

To recap that section, synchrochains can send messages to each other. The contents of outgoing messages are part of the synchrochain PoC blocks submitted by the sending synchrochain aggregators to its premiers. This is distributed to other premiers as part of the availability protocol. Circumchain blocks contain metadata that describes the relevant outgoing messages corresponding to the incoming messages for every synchrochain. The job of XCMP networking therefore, is for each recipient synchrochain to obtain its incoming messages from the outboxes.

Note that without any additional sophistication, this can always be done be retrieving the erasure-coded pieces of the A&V protocol, for example via the share network, and decoding all the outboxes of all potential senders. This of course is very inefficient - both using a broadcast medium for data only the recipient synchrochain is interested in, and in the data being retrieved which includes messages sent to synchrochains other than the recipient. Nevertheless this can serve as an initial naive implementation for the early stages of the network where traffic is expected to be low.

Further work will proceed along the following lines. One view of the problem is how to efficiently convert the outboxes of all senders, into the inboxes of all recipients. Once we have done that, any recipient can simply retrieve the inbox directly, from whomever has done the conversion. We note that the structure of our A&V networking has a very similar communication requirement - there, the pieces of each synchrochain block have to be distributed to

every other premier, and conversely every premier has to receive a piece of every synchrochain block. Therefore, our main efforts will be directed towards extending the A&V networking protocol, to support the conversion of XCMP outboxes into inboxes.

One important difference that we must cover, is that the pieces in A&V have some in-built redundancy, whereas XCMP messages have no in-built redundancy and must all be distributed reliably. Applying erasure coding to these as well, is a straightforward and obvious solution, but we will also explore alternatives.

Sentry nodes

Sometimes, network operators want to arrange aspects of their physical network for operational security reasons. Some of these arrangements are independent and compatible with the design of any decentralised protocol, which typically works in the layer above. However some other arrangements need special treatment by the decentralised protocol, in particular arrangements affecting the reachability of nodes.

For such use-cases, Primal supports running full-nodes as the sentry nodes of another fullnode that is only reachable by these sentry nodes. This works best when one runs several sentry nodes for a single private full-node. Protocol wise, briefly, sentry nodes are regular neighbours of their private node, with some additional metadata to tell others to communicate this private node via its sentry nodes. In direct-sending mode, they act similarly to TURN servers, without any resource bottleneck constraints since every sentry node is serving only one private node. These additions are fairly straightforward and more details are available elsewhere.

It is not required to run sentry nodes, for example if you believe the aforementioned security benefits are not worth the added latency cost.

A brief discussion about the security trade offs of this approach follows. One benefit of a restricted physical topology, is to support load-balancing and DoS protection across a number of gateway proxies. The indirection can also help to protect the private node if there are exploits in the software - although note this does not cover the most severe exploits that give arbitrary execution rights on the sentry node, which can then be used as a launching pad for attacks on the private node. So, while we don't believe that a public address is itself a security liability when the serving code is written well, sentry nodes can help to mitigate these other scenarios.

(An alternative possibility is for the network operator to run lower-level proxies, such as IP or TCP proxies, for their private node. This certainly can be done without any protocol support from Primal. One advantage of sentry nodes compared to this scenario is that the traffic coming from a sentry node has been through some level of verification and sanitisation as part of the Primal protocol, which would be missing for a lower-level proxy. Of course there can be exploits against this, but these are dealt with with high priority since they are amplification vectors usable against the whole network.)

Authentication, transport, and discovery

In secure protocols in general, and likewise with Primal, entities refer to each other by their cryptographic public keys. There is no strong security association with weak references such as IP addresses since those are typically controlled not by the entities themselves but by their communications provider.

Nevertheless in order to communicate we need some sort of association between entities and their addresses. In Primal we use a similar scheme as many other blockchains do, that is using the widely used distributed hash table (DHT), Kademlia . Kademlia is a DHT that uses the XOR distance metric, and is often used for networks with high churn. We use Protocol Labs' libp2p Kademlia implementation with some changes for this purpose. To prevent Eclipse attacks we allow for routing tables that are large enough to contain at least some honest nodes, and are in the process of implementing the S-Kademia approach for multipath routing.

Currently, this *address book* service is also used as the primary discovery mechanism - nodes perform random lookups on the space of keys when they join the network, and connect to whichever set of addresses are returned. Likewise, nodes accept any incoming connections. This makes it easy to support modest collaborators s and other unprivileged users, but also makes it easy to perform DoS attacks.

Further work will decouple the discovery mechanism from the address book, as described in Shareing, resulting in a more security network topology. Part of this will require some fraction of transport-level connections be authenticated against the currently-trusted premier set. However we also require to retain the ability to accept incoming connections from unauthenticated entities, and this needs to be restricted on a resource basis, without starving the authenticated entities.

Further work will also decouple the implementation of the address book from its interface, so that e.g. we can put part of in on-chain. This has different security trade offs from a Kademlia based address book, some of which is outside of the current scope of Primal such as location privacy. By offering different possibilities, we hope to satisfy a diverse set of nodes having different security requirements.

Future Work

For future work we plan to focus on a number of extensions for Primal. We want to add an incentivisation model for observers to make sure they are incentivised to report malicious behaviour. To enable trustless messaging we are working on SPREE A.1. We are also interested to increase scalability of Primal further, for example by investigating the idea of having nested circumchains. Moreover, we are working on incorporating bridging protocols A.2 to

other chains such as Bitcoin, Ethereum, and Zcash. Furthermore, to increase usability we are planning to enable *Parathreads*, which have the same utility as synchrochains but are temporary and have a different fee model.

Acknowledgement

We would like to thank the Polkadot Network for an exemplary multichain that has inspired our unique chain. For also being a chain worth modelling ours around, thus, the extensive referencing and similarities both in text, strategy and design as we could not find any better to model.

Bibliography

[1] Polkadot Network. https://wiki.polkadot.network.

[2] Availability and validity scheme. https://research.web3.foundation/en/latest/ polkadot/Availability_and_Validity/.

[3] Blind assignment for blockchain extension (babe). https://research.web3.foundation/ en/latest/polkadot/BABE/Babe/.

[4] Visa inc. at a glance, Dec 2015. Accessed: 2020-02-25.

[5] Mustafa Al-Bassam, Alberto Sonnino, and Vitalik Buterin. Fraud and data availability proofs: Maximising modest collaborators security and scaling blockchains with dishonest majorities. *arXiv preprint arXiv:1809.09044*, 2018.

[6] Handan Kılınc¸ Alper. Consensus on clock in universally composable timing model. Cryptology ePrint Archive, Report 2019/1348, 2019. https://eprint.iacr.org/2019/1348.

[7] Daniel Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. *Journal of Cryptographic Engineering volume*, 2012(2):77–89, 2012.

[8] Markus Brill, Rupert Freeman, Svante Janson, and Martin Lackner. Phragm´en's voting methods and justified representation. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[9] Ethan Buchman, Jae Kwon, and Zarko Milosevic. The latest share on bft consensus. *arXiv preprint arXiv:1807.04938*, 2018.

[10] Jeffrey Burdges. schnorrkel: Schnorr vrfs and signatures on the ristretto group. https: //github.com/w3f/schnorrkel, 2019.

[11] Vitalik Buterin. Ethereum: A next-generation smart contract and decentralized application platform, 2014. Accessed: 2016-08-22.

[12] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437*, 2017.

[13] Alfonso Cevallos and Alistair Stewart. Premier election in nominated proof-of-fix. *arXiv preprint arXiv:2004.12990*, 2020.

[14] Tarun Chitra. Competitive equilibria between fixing and on-chain lending. *arXiv preprint arXiv:2001.00919*, 2019.

[15] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Sirer, Dawn Song, and Roger Wattenhofer. On scaling decentralized blockchains. volume 9604, pages 106–125, 02 2016.

[16] Bernardo David, Peter Ga˘zi, Aggelos Kiayias, and Alexander Russell. Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-fix blockchain. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 66–98. Springer, 2018. https://eprint.iacr.org/2017/573.

[17] Sascha Fu¨llbrunn and Abdolkarim Sadrieh. Sudden Termination Auctions—An Experimental Study. *Journal of Economics & Management Strategy*, 21(2):519–540, June 2012.

[18] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 281–310. Springer, 2015.

[19] Mike Hamburg. Decaf: Eliminating cofactors through point compression. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, pages 705–723, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. https://eprint.iacr.org/2015/673.

[20] Mike Hamburg. The STROBE protocol framework. IACR ePrint 2017/003, 2017. https: //eprint.iacr.org/2017/003 and https://strobe.sourceforge.io.

[21] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 129–144, Washington, D.C., August 2015. USENIX Association.

[22] Isis Lovecruft and Henry de Valence. Ristretto. https://ristretto.group. Accessed: 2019.

[23] Petar Maymounkov and David Mazi`eres. Kademlia: A peer-to-peer information system based on the xor metric. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 53–65, London, UK, UK, 2002. Springer-Verlag.

[24] Silvio Micali. ALGORAND: the efficient and democratic ledger. *CoRR*, abs/1607.01341, 2016.

[25] Silvio Micali, Michael Rabin, and Salil Vadhan. Verifiable random functions. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 120–130. IEEE, 1999.

[26] David Mills et al. Network time protocol. Technical report, RFC 958, M/A-COM Linkabit, 1985.

[27] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, Dec 2008. Accessed: 2015-07-01.

[28] Ryuya Nakamura, Takayuki Jimba, and Dominik Harz. Refinement and verification of cbc casper. Cryptology ePrint Archive, Report 2019/415, 2019. https://eprint.iacr.org/ 2019/415.

[29] Dimitrios Papadopoulos, Duane Wessels, Shumon Huque, Moni Naor, Jan V˘cel´ak, Leonid Reyzin, and Sharon Goldberg. Making NSEC5 practical for dnssec. IACR ePrint Report 2017/099, 2017. https://eprint.iacr.org/2017/099.

[30] Luis S´anchez-Fern´andez, Edith Elkind, Martin Lackner, Norberto Fern´andez, Jesu´s A Fisteus, Pablo Basanta Val, and Piotr Skowron. Proportional justified representation. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[31] Luis S´anchez-Fern´andez, Norberto Fern´andez, Jesu´s A Fisteus, and Markus Brill. The maximin support method: An extension of the d'hondt method to approval-based multiwinner elections. *arXiv preprint arXiv:1609.05370*, 2016.

[32] Elaine Shi. Analysis of deterministic longest-chain protocols. In *2019 IEEE 32nd Computer Security Foundations Symposium (CSF)*, pages 122–12213. IEEE, 2019.

[33] Alistair Stewart. Byzantine finality gadgets. *Technical Report*, 2018. https://github.com/ w3f/consensus/blob/master/pdf/grandpa.pdf.

[34] Gavin Wood. Polkadot: Vision for a heterogeneous multi-chain framework. *White Paper*, 2016.

[35] Vlad Zamfir. Casper the friendly ghost: A "correct-by-construction" blockchain consensus protocol. 2017.

[36] Alexei Zamyatin, Dominik Harz, Joshua Lind, Panayiotis Panayiotou, Arthur Gervais, and William J. Knottenbelt. XCLAIM: trustless, interoperable, cryptocurrency-backed assets. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 193–210. IEEE, 2019.

primal network

info@primalnetwork.org
admin@primalnetwork.org