# WHITEPAPER

## Lybell Graphchain

### A CROSS PLATFORM BLOCKCHAIN FOR SCALE, SECURITY AND RELIABILITY

docs@lybell.io

Version 0.8

January 2024

Lybell Graphchain (LGC) Core Development Team

The Lybell Graphchain (LGC) is a layer-1 platform designed to make crypto intuitive and effortless for the user, in a scalable, secure, reliable and environmentally conscious way. In short, it tackles the predominant challenges faced by existing blockchain platforms. The Lybell (non-blockchain) design achieves this through combining two novel approaches: a directed acyclic graph (DAG) based consensus method, and a simplified transaction process. The DAG-consensus makes transactions appear quickly, and ensures that the system scales with demand. Figure 1 illustrates a simple Lybell Graphchain DAG, with the highlighted sections in blue, nodes $A, I$ and $R$ representing the convergence points — points at which all previous transactions are referenced at the head of the chain (details in the paper Appendix). The simple transaction process refers the use of transaction scripts at the protocol level, that is used to make sending and receiving a variety of transaction types straightforward and secure.

## Background

The LGC White Paper (this paper) is an introduction to the current Graphchain system. It provides a introduction to the LGC blockchain, its core ideas, an

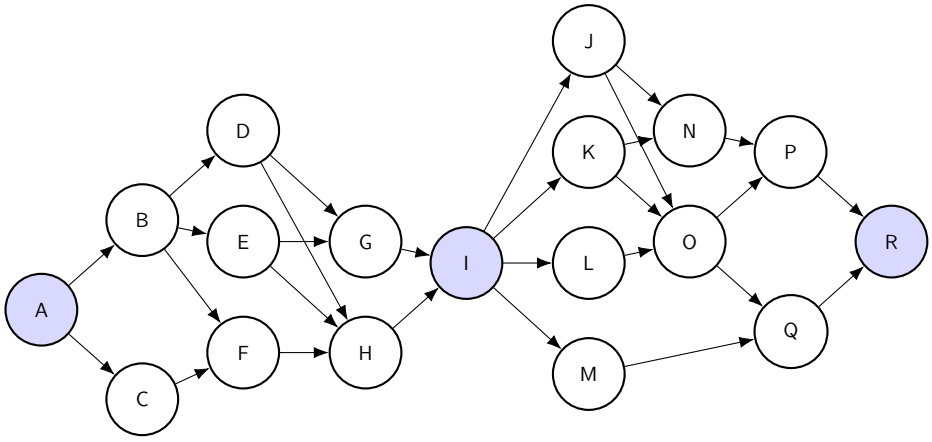


Figure 1: representation of the Lybell Graphchain DAG.

outline of the consensus mechanism and its intended applications. The current document version is 0.8. Lybell Graphchain (LGC) builds on top of the Grapchchain framework presented at the Blockchain, Cryptocurrencies and Contracts workshop at AsiaCCS in 2018 [6]. It emerged from the work started in early 2016 and was distributed on the Cryptology ePrint Archive of the International Association for Cryptologic Research (IACR) [5]. The idea is also featured as a slightly more accessible article in ERCIM news [8]. The prefix *Lybell* represents the specific implementation of the ideas presented in the Graphchain paper, and the name itself is inspired from a variety of dragonfly; libellula.

This white paper is written as a set of principles that we will build into a working system. We recognise that the design decisions presented may change as development progresses, or at a later stage in alpha or beta testing. This document itself may change to reflect decisions made at implementation time.

**At the time of writing, LGC is in alpha-build development.**

# I Design changes

Since the Graphchain paper was first created and published in 2016, it has become clear that the wholly proof of work (PoW) consensus method (as presented) is no longer a realistic option for any system designed with scale or environmen-

tal considerations in mind. This has resulted in a need to modify the consensus mechanism, as we no-longer rely on proof-of-work for consensus (though there remains a function for sybil resistance).

As a transitional measure, we have adopted a drop-in BFT-like consensus mechanism, designed to maintain the integrity and functionality of the network while developing a DAG-based proof-of-stake (PoS) model. This temporary mechanism serves as a way of demonstrating the network functionality during the development period. Our goal is to implement a novel type of PoS consensus mechanism, tailored to the unique DAG-based requirements of our implementation. Details of development and characteristics will be presented in forthcoming updates.

We describe the high-level changes to the consensus method within the following Features section.

# II Features

This section describes the features of the LGC system.

## II.1 Scalability

Lybell GC uses a directed acyclic graph (DAG) mechanism for the addition of new transactions onto the growing chain, which is described in the paper appendix. Unlike other DAG approaches, the system encourages rapid convergence to a new point in the DAG by allowing each new transaction to refer to any number of descendants. This approach ensures that the Lybell blockchain can scale effectively to meet the demands of a wide range of applications and industries.

In addition, we provide a proof-of-stake (PoS) like validation-overlay, to rapidly confirm state-updates and the validity of the network. Trust assumptions can then be defined on an individual/cost basis. We provide a reward mechanism for the work of validators to incentivise their continued network support.

## II.2 Security

The system is built using well-known, and standardised, cryptographic primitives and protocols. However, various competitor blockchain systems exist that are

each performing their own security. The protocol makes use of this distributed consensus effort on various other networks to secure and checkpoint the LGC state at regular intervals, further reducing the possibility of malicious interference and creating frequent roll-back states for a worst case scenario.

## II.3 Sustainability

LGC has been designed with energy efficiency in mind, and so moves on from the original PoW design of the appendix, adapting the consensus mechanism to utilise a validator selection PoS approach, and utilising the work performed on various other chains.

## II.4 Real decentralisation

LGC recognises that decentralisation is achieved through means that allow for not just the operation of the state consensus, but also through governance and ownership. Therefore the native system tokens are used not only for verification, but also for system governance.

### II.4.1 Tokens for utility

The native token is designed primarily as a utility token. The main functions are:

- Validator selection
- Including with state updates to incentivise state update and capture within the network

## II.5 Inflationary measures

Token operation for the first 3to5 years of operation are set at 0%, with an increase after that to be set by network consensus. We expect a small inflationary schedule after this date to encourage validator participation.

## II.6 Scripts and smart contracts

LGC offers a structured approach to smart contracts, whereby smart contracts are standardised and introduced into the system over time, and after testing. This represents a move away from the more liberal approach that allows any smart contract to be created, such as in the Ethereum network. However, it is designed to be more flexible than the conservatism we have seen in some earlier systems.

The initial design will focus on fast token transference, with a limited set of scripts including:

1. Regular fungible transfer,
    - one to one,
    - one to many,
    - many to one, and
    - many to many;
2. Regular non-fungible token creation and transfer
3. Cryptographic data inclusion, supporting (in combination):
    - encrypted data inclusion
    - signed data inclusion
    - plaintext message inclusion with data integrity
4. Token creation and distribution.

We expect to quickly add functionality for the following:

1. Non-transferable token creation and distribution;
2. Micro-payment capabilities with randomisation;
3. Identity-based transactions for
    - pay to email;
    - pay to phone number;

- encryption support.

In addition, at the present time, we intend to explore potential avenues for the development of:

- Privacy coin support;
- Offline transactions;
- Secure communication between other networks.

## II.7 Individuals and business

The LGC chain aims for versatility, accommodating both individual users and business operations, within a decentralised framework. Our goal is to enable direct interaction on the main chain for commercial operations, while upholding the same network properties required for a public chain. We aim to offer business support shortly after the main chain release.

# III A: Graphchain paper

The following appendix pages showcase the Graphchain concept as it was initially introduced in the original eprint, and later updated upon publication in 2018. This presentation also incorporates more recent modifications to the design. As explained in the introductory sections of the whitepaper, some of the original ideas have evolved. We have decided to include this information as an appendix, both for reference and to provide interested readers with a deeper understanding of the motivations and inspirations behind the initial design.

**Authors:** Xavier Boyen, Christopher Carr and Thomas Haines.

## Contents

# 1 Introduction

The Graphchain framework emerged in the wake of proposals hoping to solve some of the issues in the Bitcoin system. The design sought to improve the scalability issue, while at the same time maintaining the decentralised properties.

Various factors account for Bitcoin's success, though there is little question that the decentralised design and perceived immunity from government interference played a significant part in its uptake.

The absence of imposed centralised authority did not prevent the emergence of large syndicates of mining pools, which still hold a great amount of control over blockchain systems. There are a variety of issues with the Bitcoin Blockchain and blockchain systems in general. Unfortunately, some issues cannot be fixed without a complete system redesign [2, 4, 17, 24, 26, 27, 40]. The range of proposals goes from tweaks, or altcoins [43, 42], to holistic design changes, sometimes with in-built centralisation [37] or escrow [9] as solutions to the perceived problems. Nonetheless, two critical issues remain:

1. **Centralisation:** Consolidation of block mining into the hands of a small portion of the total participants.

2. **Scalability:** Long delays until transactions are considered as included in the system.

It can be argued that both problems are inherent to blockchain technology, and that we therefore need a new approach.

### Centralisation

The centralisation issue within blockchain technology is an artefact of the construction of the first blockchains. Blockchain systems, coupled with their proof-of-work mechanisms, make it difficulty to regularly and fairly distribute rewards to the myriad of participants that contribute their computing power to the network. On these blockchains the rewards gained for any individual user's effort are few and far between, making solo mining (mining as an individual, and not part of a mining pool) a high-risk, high-reward venture.

To alleviate these issues, participants coalesced into mining pools to ensure a more frequent income. This comes at the cost of abdicating their duty for

verification oversight and collectively makes the network more susceptible to the attacks. This is not just the *51% attack* [30], but also the "selfish miner" or *33% attack* [16], which in some special cases can become a *25% attack*. To emphasis this issue, note that a single one of these pools has temporarily held absolute majority of the computing power on the network [1]. In essence, this move to a system of mining pools defeats the decentralised principles, and is a widely recognised problem [24, 29, 25].

### Scalability

Blockchained cryptocurrencies use a feed-back-loop mechanism to alter the difficulty of creating a block, to ensure that new blocks are created approximately within a predefined period of time. Unfortunately, periodic block creation is troublesome, especially when it involves a high-variance Poisson process — as each puzzle attempt has probability of succeeding in independent of the previous solution. For miners, this is often just enough time to ensure that they quickly learn of newly created blocks, and perversely encourages the miners to delay the propagation of new blocks other than their own. For users, unpredictable block creation times means that they must wait for transaction confirmation and hinders the adoption of blockchain technology for real-world payments. These scalability issues due to the use of blockchains are a recognised problem [10]. Commercial Bitcoin users now mitigate the blockchain delays by entrusting third parties as payment processors, defeating the decentralised principle. The company Bitpay, `bitpay.com` is one such example.

In short, contention and consolidation are inherent to any blockchain reward structure, making it unsuitable to cater for a large population of miners and users. Despite its importance, the difficulty of realizing a proof-of-work scheme with better characteristics than a blockchain is a central problem in cryptocurrency design. Paraphrasing Narayanan et al. [31]: the holy grail would be to design a consensus protocol which is 'naturally' low-variance by offering smaller rewards for lower-difficulty puzzles.

# 2 Overview

This paper presents a blockchain-free proposal, called the Graphchain. This white paper version is an extended and updated version of the work carried out in [5] and [6]. Instead of relying on a single block structure, the idea of a graphchain shifts the task of affirming transactions onto the transactions themselves. Verification therefore no longer results in a single chain of blocks containing transactions, but in a graph comprised only of transactions validating previous transactions. Figure 2 visually represents the construction. Each square represents a transaction that collects (see Definition 1) one or many previous transactions. The highlighted transaction at the front, or head, of the graph represents a new transaction that includes directly the two transactions that indirectly include all previous transactions in the graph.

## 2.1 A Graphchain

In a graphchain, when a transaction is posted, in addition to the cryptocurrency aspects of signatures and money transfer, it will refer to any unverified previous transactions deemed to be valid. A transaction is said to include, or form a PoW on, some distinct set of transactions if that set is included in the transaction formation. This is akin to including a reference to a previous block in a blockchain. This induces a growing graph of verifications, where each transaction verifies and secures its parent transactions. This can be represented visually like the example in Figure 2.

The growth of the graph is steered using an incentive system that encourages affirming recent transactions while equitably rewarding all transactions, even when they share the same parents — up to some reasonable limit. Figure 2 illustrates how transactions refer to each other. Each arrow points from an ancestor (or parent) node, to a descendent (or child) node. These nodes are called transactions. They can also be thought of as small blocks to use blockchain terminology.

Eventually every valid transaction is verified by all new valid transactions, making the transactions as immutable as in a linear blockchain. This is called *convergence*, and a set of transactions are said to have converged if they all share a common descendant in the graph. In Figure 2 the graph is converged

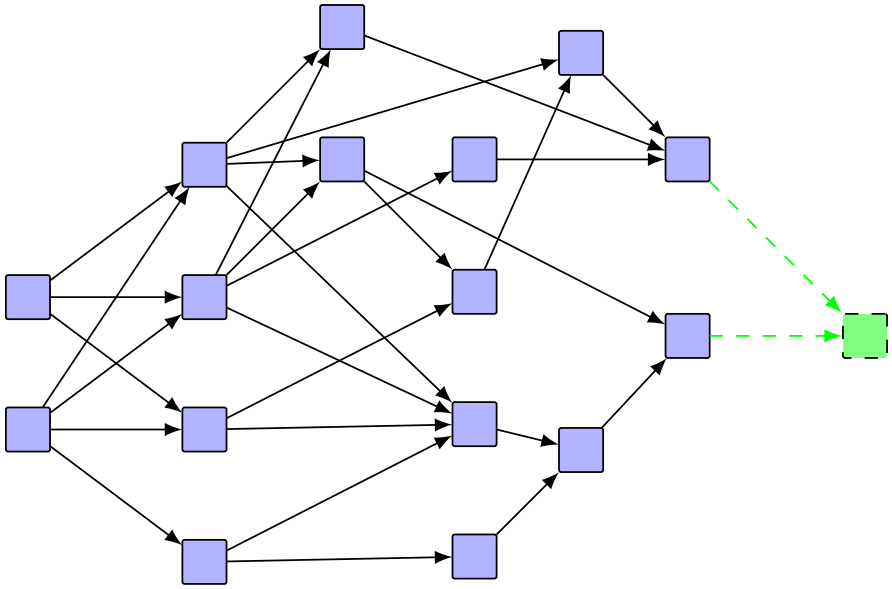

Figure 2: Visual Representation of a Graphchain

on the highlighted transaction, as it is a common descendant of all others. Section 4 proves that convergence does occur, and it is experimentally verified that convergence occurs rapidly, even with network delay, which is essential for scalability. This work, originally published on the IACR eprint archive [5] and since revised, was the first to propose an entirely graph-based cryptographic ledger, a graphchain, which is a robust and suitable stand-in for a blockchain.

## 2.2 Related Ideas

Various problems with Bitcoin's model have previously been studied. Karame et al. [21] evaluate the problem of payment verification speed. Miller et al. [28] look at replacing the PoW in Bitcoin with proofs-of-retrievability, in order to mitigate the waste of computational resources, similar to Park et al. [33] who present a proof-of-space alternative. Gervais et al. [18] critically analyze the claims of decentralization in Bitcoin, while Johnson et al. [20] review the incentives for mining pools to engage in underhanded strategies to achieve a competitive advantage. Pass et al. [34] analyse the Blockchain under an asynchronous setting, where users can join and leave the system as required, and scalability improve-

ments are the focal point of work by Sompolinsky and Zohar [39]. Taking the concern for centralization further, Danezis and Meiklejohn [11] consider the best possible outcomes if a partially-centralised authority is assumed.

Additionally, Miller et al. [29] propose computational puzzles that cannot be outsourced, thus removing the ability for users to work together, such as in mining pools. Lewenberg et al. [24] suggest a scheme where multiple blockchains emerge at once, in order to include more transactions. These two proposals attempt to resolve the two issues focused on here. This chapter, however, takes the different approach of changing the underlying blockchain structure, in order to incentivise solo work, and aims to capture the two desired properties simultaneously. Moreover, in direct contrast to Lewenberg et al. [24], this proposal allows for multiple rewards for securing the same transactions.

Subsequent to the results of this research [5] was Spectre [38] and the work of Bentov et al. [3]. Spectre, like Bitcoin, uses miners to collect transactions into blocks, but allows those blocks to form into directed graph instead of a chain. Sompolinsky et al. [38] show that a directed acyclic graph alleviates a theoretical loss of security when the rate at which blocks are issued significantly slows down. While our solution ensures the order of any two transactions is agreed upon by all non-corrupt nodes, Spectre does not. Spectre however is still based on blocks and dedicated miners who create them, and so can be considered as part way between the original Bitcoin blockchain and a block-free construction. Spectre also has a very specific focus: to avoid certain theorised attacks against the blockchain that may occur in certain circumstances; Spectre does not get rid of transaction rate bottlenecks and miner centralization. The proposal of Bentov et al. [3], more recent than Spectre, achieves many similar properties, but uses a much more complicated protocol to achieve the results, involving an additional Byzantine agreement layer. In contrast, this proposal achieves the same results with a far simpler design.

IOTA [19, 36] also propose a graph-based system. Their approach vastly differs from the work here. The IOTA system assumes an honest majority of users and provides arguments for scalability from a statistical, rather than adversarial, perspective. This work employs incentive structures within the security analysis of the framework in Section 4. One concern when arranging large numbers of transactions into a graph, as opposed to a blockchain, is that the relevant graph

algorithms become a computational burden in the worst case. IOTA appears to do nothing to address or prevent this issue. The Tangle [36] protocol, used in IOTA, may be well behaved under chance events, but not against deliberate action, leaving it at risk of devastating denial-of-service attacks. In contrast, it is one of the core contributions to design a block-free graph-based distributed ledger that guarantees fast and correct consensus through a combination of incentives and enforcement. Another concern is that not much is known about the implemented IOTA source code, which is a worry given that IOTA implemented their own hash function Curl, which was recently broken [14].

Lerner [23] proposes a DAG design, similar in principle to the work presented here. Unfortunately, there is no well defined incentive structure and no analysis of the security in the system. Properties of scalability are also left unclear.

The Hashgraph [45], released in late 2017, also bears some similarities to this framework, in that they too use a graph based structure. A notable difference is that they rely on byzantine agreement protocols, but argue that the hashgraph can be changed to a permissionless system. Unfortunately, while they propose switching to a proof-of-work or proof-of-stake model, they fail to include any analysis of how the incentive structures provide security in this setting.

One emerging solution to transaction throughput issues is payment channels such as the lightning network [35] and Duplex Micropayment Channels [13]. These channels allow on-blockchain transactions to be leveraged to allow for secure off-blockchain transactions, allowing significantly increased throughput, which is a complementary direction of inquiry, and is not exclusive of the work contained here.

Decker et al. [12] published an analysis of the relationships between block size, delay and security in Bitcoin. Since the graphchain scheme rests on different security assumptions, the applicability of their analysis is not clear. As a solution to Bitcoin's delay problems Sompolinsky et al. [39] introduced the GHOST protocol. While GHOST rectifies some of the issues with delays it does so by allowing the inclusion of non-accepted blocks and so does not alleviate the risk to small users of not receiving rewards for their mining effort.

Ethereum [32] currently implements a reduced version of the GHOST protocol, which allows for fast block creation times, and mitigates the problem of orphan blocks — which they call uncles — by allowing them to be referenced

by newly created blocks. It incentivises this behaviour by increasing the reward for the miner of the new block. In Ethereum however, transactions within uncle blocks are not valid, and so transaction throughput is not increased. For Ethereum, the improvements are directed at the user-facing cryptocurrency layer, which are essentially orthogonal to this chapter.

Bitcoin NG [15] and Byzcoin [22] are proposals for separating leader selection from transactions. The innovations of Bitcoin NG and its successors are mostly geared at improving mining fairness within the original paradigm of the blockchain.

The work of Carlsten et al. [7] on the instability of Bitcoin based on fees alone implies a long term problem for Bitcoin and many other currencies. However, the results do not directly apply to the construction here, since fees are not claimed immediately.

## 2.3 Aims of Graphchain

The graphchain mechanism described in this chapter has considerable advantages over the blockchained model:

1. It removes the need for mining-pools.
2. It enables faster transaction confirmations by removing the block aspect.
3. It gives commensurate rewards for differing PoW effort.
4. It allows transaction volume to scale naturally with fluctuating activity.

The framework removes the need for a single blockchain, and allows for a lighter system of transactions arranged into a directed graph. Multiple miners can get rewarded for verifying the same transactions, which simultaneously removes the competition to be the first to create a new block. It allows miners to work at unequal speeds with unequal resources, while still obtaining regular rewards for their efforts, and all the while, contributing to the verification of transactions and system security.

### 2.3.1 Transactions as First-class Objects

Without blocks, transactions have a dual function: *transactional* and *structural*. Accordingly, our transactions consist of:

- A transactional/monetary component, representing the user-facing aspect of the transaction—the payload— for example; cash, currencies, securities, contracts, etc. This framework is oblivious to this component, other than for a minimal ability to carry a value for the fee and reward mechanism.

- A structural/mining component, representing the systemic aspect of the transaction—its metadata. This framework relies on this component to build a globally consistent verification graph using PoW, incentives, and other mechanisms.

### 2.3.2 Fees and Incentives

Each transaction must post a transaction fee: an offering for collection by future transactions that will verify it. Conversely, each transaction must refer to a selection of prior transactions called parents, which must be valid and have fees available for collection. Referencing the parents indicates verification and causes the verifier to collect a certain amount of available fee from them and their ancestors.

Fees are collected from the oldest ancestors with fee remaining first, by walking the directed acyclic graph of those ancestors and selecting any available fee that remains up to a prescribed total amount, which is based on the amount of work performed. The fee and incentive method here differs from the blockchain, and encourages the following properties:

1. Verification is directed toward recent transactions, since direct validations of older transactions will be forbidden once their fees are exhausted.

2. Higher fee transactions can attract parallel verifiers for rapid affirmation.

3. Fees from a transaction are not immediately claimed, but are instead gradually drained as a transaction gains more descendants.

4. A Time-To-Live parameter allows the acceptable network delay to be encoded into the system, which controls the time taken to deplete the fees of a transaction. Provided the network delay is less than this parameter then slow transactions will be unaffected.

5. Any transaction, once collected into the verification graph, will at some point become connected to the ancestor tree of every future transaction in the graph, eventually providing maximum validation regardless of fee — the feature refer to above as convergence.

6. Invalid transactions, for reasons such as double spending, stale fees, misformatting, and so on, will be weeded out by majority vote of the miners, who are incentivised to refuse to extend the transaction graph from the fault.

This approach greatly simplifies the challenge of maintaining global consistency of a partial order by exploiting the economic incentives available within a cryptocurrency. Miners are incentivised to work from the head of the graph (to keep it lean) by requiring that parents still have enough available fee remaining to be eligible parents. Note also that the overhead of embedding the PoW verification process within each transaction is small in comparison to the transactional data.

### 2.3.3 Cryptocurrency Independent

Because our graph-based validation and cooperative PoW framework is independent of the payload of the transaction itself—other than a way to collect fees, pay fees, and create assets (or coins)— the intention is that one can instantiate it with any cryptocurrency functionality of choice. Transforming any existing blockchain-based cryptocurrency into essentially the same cryptocurrency with graphchain-based verification potentially improves scalability and decentralisation by giving everyone an opportunity to profit from their individual participation toward securing the network.

# 3 A Block-Free Ledger

This section describes the rules for a graph-based ledger. It relies on PoW and derived incentives to achieve global consensus, convergence, and timely verification of new transactions.

## 3.1 Transactions

Transactions perform all roles in the system: they create and redistribute tokens, add fees and confirm previous transactions. To create a transaction $x_i$, the following information is provided:

1. Payment information $\text{Pay}_i$ transferring the tokens, as in other cryptocurrencies such as Bitcoin.
2. A set of $n$ previous transactions $T_i = \{x_1, x_2, \ldots, x_n\}$, for some $n$ where $n \geq 1$. This set references the previous transactions that are considered valid transactions by the transaction creator. The transaction being created $x_i$ is not in the set $T$.
3. A transaction fee $f_i$. Every transaction must contain a fee, such that $f_i > 0$, in order to offset the distributed cost of conveying and verifying the transaction.
4. A number of tokens being created and prize being claimed $m_i$.
5. A solution value $s_i$.

Thus a transaction is the tuple:

$$x_i = (\text{Pay}_i, T_i, f_i, m_i, s_i). \tag{1}$$

The parent transactions $T_i$ are mandatory references to at least one prior transaction whose validity the present transaction is vouching for. This indirectly vouches for the validity of the ancestors of the transactions in $T_i$. Provided that the new transaction is itself valid, the PoW attached to the new transaction will add onto the cumulative PoW associated with the parent transactions, and likewise strengthen all of their ancestors.

**Definition 1** (Contains and Collects). *For any transaction defined as above, where $x_i = (\text{Pay}_i, T_i, f_i, m_i, s_i)$, a transaction is said to collect or contain the transactions included in $T_i$ within its proof-of-work.*

For example, the Figure 3 graphically represents transactions where $x_7$ collects transactions $x_6$ and $x_5$, $x_5$ collects transactions $x_1, x_2$ and $x_3$ and transaction $x_6$ is said to collect transactions $x_3$ and $x_4$. Thus the transaction set $T_7$ for $x_7$ would be $\{x_5, x_6\}$.

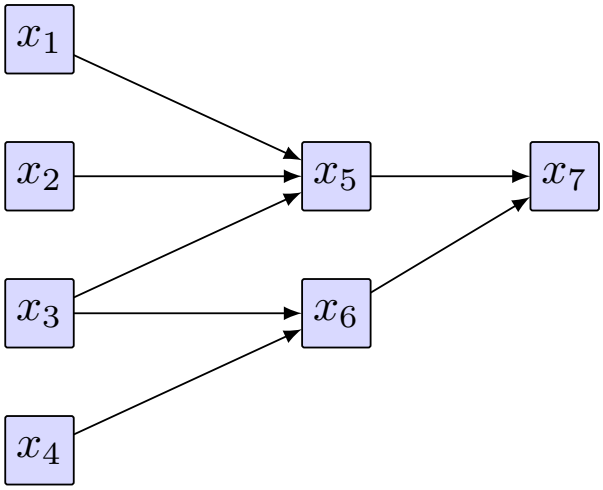


Figure 3: Example Transaction Diagram

An ordering relation on transactions can be expressed.

**Definition 2** (Transaction Ordering). *Let $G$ be a set of transactions. For $x$ and $x'$, two distinct elements in the set, $x \prec x'$ if $x'$ contains $x$, and vice-versa for $x \succ x'$. The set $G$ equipped with this ordering relation $\prec$ is called a Transactional Partially Ordered Set, or T–POSET.*

Returning to the example figure, Figure 3, there is an ordering on the set of transactions in the graph given by $x_3 \prec \{x_5, x_6\} \prec x_7$. This is a partial ordering of transactions (a T–POSET), as it is not possible to strictly order all transactions, such as $x_1$ and $x_6$.

Each transaction $x_i$ within a T–POSET has a solution value $s_i$ to some PoW function. This is purposefully left undefined here to allow for any variety of PoW function. In practice, this could be achieved with something similar to the Bitcoin PoW function, where the challenge is to find a solution such that $H(x_i)$ has a number of leading zeros. Notably, this work allows the transaction creator to decide for themselves how difficult they would like the challenge to be, provided certain conditions are met, as explained later. What matters is that there is some well defined way to calculate how much computational effort went into creating the transaction. This is defined as the function Work.

**Definition 3** (Work). *Let $G$ be a T–POSET of transactions and let $x_i$ be*

*a transaction in $G$, then* Work$(x_i)$ *is a function returning a positive real value representing the expected computational effort required to create the transaction $x_i$.*

For example, if the PoW function is to hash a transaction with a suitable hash function $H$, and a transaction $x_i$ is provided with solution $s_i$ such that $H(x_i)$ has 10 leading zeros, then the expected number of computational steps required to create the transaction, assuming the hash function $H$ behaves like a random oracle, is $2^9$, and so Work$(x_i) = 2^9$.

As the transaction graph is created with each transaction having a well defined work value, it is possible to calculate the cumulative work of all the transactions that are descended from any particular transaction. This is called the weight.

**Definition 4** (Transaction Weight). *Let $G$ be a T–POSET and let $x_i \in G$ be a transaction. Let $Y = \{y_i : y_i \succ x_i\}$ be the set of all the descendants of $x_i$. The* weight *of $x_i$ is defined as the sum of the work contributed by every one of the $y_i$,*

$$\text{Weight}(x_i) = \sum_{i=1}^{|Y|} \text{Work}(y_i). \tag{2}$$

This notion of weight is well defined given any T–POSET, however it is also dynamic in the sense that as the T–POSET grows with new transactions, the weight of an existing transaction will grow as it gains new descendants.

To help describe the weight property, let us extend the example from Figure 3 with Figure 4. Each transaction in the graph is annotated with a work value for each transaction, coloured red. The transaction $x_3$ will have weight Weight$(x_3)$ = Work$(x_5)$ + Work$(x_6)$ + Work$(x_7) = 73$. The transaction $x_2$, which is not an ancestor of $x_6$ will have weight $48$. The transaction $x_5$ and $x_6$ will both have weight 18.

Maintaining consensus across multiple verifiers requires the notion of the *height* of a transaction. For a transaction $x_i$, Height$(x_i)$ is the total work of $x_i$ and all the ancestors of $x_i$.

**Definition 5** (Height). *Let $G$ be a T–POSET and let $x_i \in G$ be a transaction. Let $y_1, \ldots, y_m$ be elements in $G$ and ancestors of $x_i$, then the* height *of $x_i$ is*

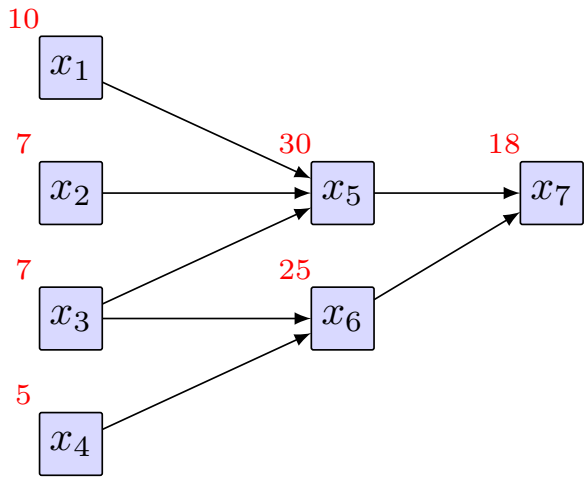


Figure 4: Example Weighted Transaction Diagram

*the sum of the work of every ancestor of* $x_i$*, plus* $x_i$ *itself, given by,*

$$\text{Height}(x_i) = \text{Work}(x_i) + \sum_{i=1}^{m} \text{Work}(y_i). \tag{3}$$

In Figure 4, the height of transaction $x_1$ is its work value $10$. The height of $x_5$ is its work value plus that of all of its ancestors, so $30 + 10 + 7 + 7 = 54$. The height of $x_6$ is then $25 + 7 + 5 = 37$. Now, the height of $x_7$ is the sum of all the weights in the graph so $\text{Height}(x_7) = 92$.

## 3.2 Transaction Validity and Fee Collection

Unlike in other cryptocurrencies, transaction fees are not immediately claimed by the immediate descendant, rather fees increase the total *prize* value of the transaction, see Section 3.3, available for collection by a number of descendants. Every transaction $x_i$, linking to earlier transactions $T_i = \{x_1, ..., x_n\}$, collects a positive and well defined amount of fees, from the union of the transactions in $T_i$ and their ancestors. For the new transaction $x_i$ to be valid, it must meet the following two conditions:

1. The fee that $x_i$ takes is included as part of the token creation $m_i$, and must be available from the prize set of the union of all of $x_i$'s ancestors.

2. For every transaction listed in $T_i$, the prize must be non-zero.

These constraints are important later for ensuring that new transactions are validated in priority, and that all valid transactions quickly converge to sharing a common descendant.

The amount of prize that a new transaction $x_i$ collects is fully determined by $x_i$ and its *local context* (i.e., the smallest subset $G' \subseteq G$ containing $x_i$ and its ancestors). The collected prize increases with the value of work of $x_i$, or even proportionally with an automatic proportion factor $\beta$; see Equation 7 and 8.

### 3.3 Prize

The prize of a transaction $x_i$ with respect to some T–POSET of transactions $G$, is the sum of the fee that is still available from $x_i$ and all of the ancestors of $x_i$. Prize is also a dynamic notion: it is highest when $x_i$ is new and has no descendants, and decreases as the graph $G$ grows and the fees from $x_i$ and its ancestors are claimed by its descendants. The prize of any transaction $x_i$ is a well-defined quantity given the current state of a the graph.

### 3.4 Drain and Prize Depletion

As prize is collected, it is necessary to decide from which ancestors a new transaction will collect its prize in terms of fees. The method employed is to deplete the oldest eligible nodes first, defining age based on their height. This has two desirable purposes: Firstly, verifiers will be further compelled to affirm newer rather than older transactions, otherwise they risk their effort being wasted. Secondly, the sooner that the prize of a node reaches zero, the sooner it and all of its ancestors can be pruned from the dynamic data structure that each verifier must maintain to keep track of the amounts still available for collection.

Defining the depletion ordering unambiguously, $x_i$ *is older than* $x_j$ if Height$(x_i) <$ Height$(x_j)$. This relation induces a total order that is compatible with the partial order of the T–POSET. Ties will be highly unlikely, and can be decided in any fixed deterministic manner, such as comparing the hash of the two transactions and saying that transaction $x_i$ is depleted before transaction $x_j$ if $H(x_i) < H(x_j)$.

Thus, when a new transaction is created and broadcast, the fees that it collects will be taken from the oldest of its ancestors that still have fees available to collect, moving forward towards the top of the graph as those oldest transactions become depleted. Thus, the prize of a transaction $x_i$ is the total amount that can still be collected from $x_i$ and all of its ancestors, and clearly for all such ancestors, their prize strictly less than that of $x_i$.

The definition of prize of a transaction is given in terms of how much a prize a transaction takes, called the *drain*. The drain of a transaction $x_i$, then, is the current portion of the fee of $x_i$ that has already been collected by the descendants of $x_i$. Clearly, $\mathsf{Drain}(x_i) = 0$ for a transaction without descendants, and approaches $\mathsf{Drain}(x_i) = f_i$, the original transaction fee, as $x_i$ acquires descendants that collect its fee. As $\mathsf{Drain}(x_i)$ is also dynamic, it depends on the current state of the T–POSET $G$ in the view of a particular verifier at a given time.

**Definition 6** (Drain). *Let $G$ be a T–POSET, containing $n$ transactions. Let $x_i \in G$ have $m < n$ descendants $\{y_1, y_2, \ldots, y_m\}$, and for each $y_i$, define $\delta_i$ to be the prize taken by $y_i$ from $x_i$. Then for each transaction, the drain of $x_i$ is defined by:*

$$\mathsf{Drain}(x_i) = \sum_{i=1}^{m} \delta_i \tag{4}$$

The prize of a transaction $x_i$ is then the total fees brought by $x_i$ and its ancestors, minus by the total drain from its descendants. Likewise, prize is also a dynamic notion that depends on the current T–POSET $G$.

**Definition 7** (Prize). *Let $G$ be a T–POSET, containing $n$ transactions. Let $x_i \in G$ be a transaction, with fee $f_i$. The prize of a transaction $x_i$ is its fee minus the drain applied to $x_i$:*

$$\mathsf{Prize}(x_i) = \big(f_i - \mathsf{Drain}(x_i)\big). \tag{5}$$

*Further, for any set $G$ the prize of a set of transactions, $T \subseteq G$ is given by,*

$$\mathsf{Prize}(T) = \sum_{x \in T} \mathsf{Prize}(x) \tag{6}$$

Figure 5 provides an illustration of a transaction graph annotated with values representing prize and drain. These values are different from those in Figure 4.

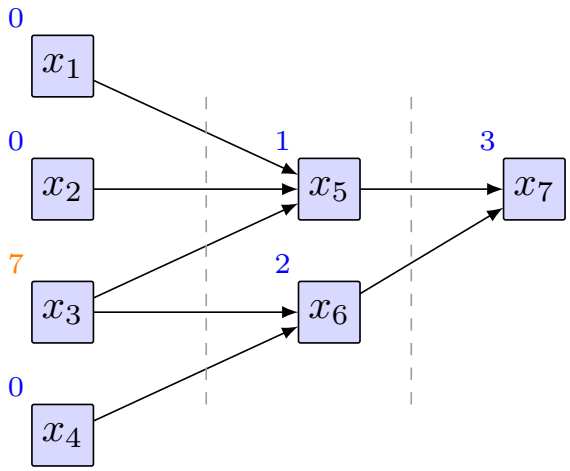


Figure 5: Transaction Diagram of Prize and Drain

The graph is split into three states, by the dashed lines. The first state is where only the first 4 transactions $x_1$ to $x_4$ are available. The second state includes the first state as well as $x_5$ and $x_6$. The third state includes all transactions. This three state diagram is designed to representing the evolving nature of this system. Transaction three posts a fee value 7, coloured orange. Since it has no ancestors, this is also its prize in the first state.

The values coloured in blue represent the drain value taken from $x_3$. Thus, in the first state of the diagram Drain$(x_3) = 0$ and Prize$(x_3) = 7$. In the second state Drain$(x_3) = 3$ contributed by $x_5$ and $x_6$ and Prize$(x_3) = 4$. In the third state Drain$(x_3) = 6$ and Prize$(x_3) = 1$.

## 3.5 Drain Rate Adjustment

Consider the total prize available across all the current transactions in the system $G$, given by Prize$(G)$. Macroscopically, it is possible to control the time it will take for the combined verification effort to deplete this prize completely, this is the purpose of the Time-To-Live parameter. This time is the expected lifetime of a transaction in the T–POSET, before it can no longer be the direct ancestor of a future transaction.

Controlling this lifetime can be done by adjusting the rate at which a trans-

action $x_i$ can drain the fees from its ancestors, in proportion to the difficulty of the PoW posted by $x_i$. Specifically, assuming that the combined computational power of the whole system is constant, then the time to drain a transaction, the Time-To-Live, is a ratio of the total available prize over the total work of all the transactions since the system's inception, i.e.,

$$\frac{\text{Time-To-Live } \ (\text{sec.})}{\text{Age-Of-System } \ (\text{sec.})} = \beta \cdot \frac{\text{Prize}(G)}{\sum_{x_i \in G} \text{Work}(x_i)} \tag{7}$$

Here, $\beta$ is an exchange rate parameter indicating how much fee is to be collected from its ancestors by a transaction that posts a PoW of unit difficulty. Time-To-Live parameter must be chosen as a global system constant, and selected large enough to ensure network delay does not cause new transactions to enter race conditions, yet small enough to encourage creating transactions from the leading transactions on the system.

Unfortunately, $\beta$ determined from the above equation will be technically ill-defined unless all the verifiers share the exact same view of $G$ at the time it is determined. Letting $G' \subset G$ be the uniquely defined ancestor set of a transaction $x_i$, define:

$$\frac{\text{Time-To-Live } \ (\text{sec.})}{\text{Age-Of-}x_i \ \ (\text{sec.})} = \beta \cdot \frac{\text{Prize}(G')}{\sum_{y_i \in G'} \text{Work}(y_i)} \tag{8}$$

This leads to well-defined values for $\beta$ that are easy to verify. Note that Age-Of-$x_i$ needs a clock. Since absolute precision is not paramount to determine $\beta$, it is possible to allow whichever client whose onus it is to recompute $\beta$ to use its own clock, and to require that the verifiers accept it unless the clock skew is very substantial, such as more than one hour. This is the approach used in current blockchain systems where clocks are required.

### 3.6 Verification

The transaction verification process is intentionally similar to blockchain-based cryptocurrencies, with some small differences. Users verify transactions immediately as they are received. Upon receiving notice of a new transaction $x_i$, the client first checks that all previous transactions collected by $x$ are acceptable in terms of formatting.

The next check is to ensure the transaction has the appropriate work attached for the prize being claimed. A final part for verification is to check that the

transaction $x_i$ itself is valid, which requires that it be both *intrinsically correct* or well-formed, and *extrinsically admissible* or valid in the current ledger context. The former condition is a (static) determination whether the transaction could be valid in the smallest possible ledger context that contains it and its ancestors; if that fails it is forever marked as ill-formed. The latter condition is a check for double spending, and the conditions such as availability of fees that the transaction is claiming.

### 3.6.1 Well-formedness

In order to check that a transaction is inherently well-formed, the criterion is simply that it be valid in at least one POSET, namely the smallest possible POSET that contains that transaction together with all the transactions it references and their ancestors. This check is a special case of general conflict resolution where conflicts are checked only between the transaction's ancestors. What makes intrinsic correctness important is that it is a permanent and static notion: once a transaction has been deemed incorrect, it can be placed on a permanent exclusion list, as well as any future transaction that references it.

For a given transaction $t$, let $A(t)$ denote the ancestors of $t$ and Height$(t)$ denote the height of $t$. We say that $t$ has promptly converged ancestors if:

**Definition 8** (Promptly Converged Ancestors)**.** $\forall x, y \in A(t), \text{Height}(x)+f(x,y) > \text{Height}(y) \Rightarrow y \in A(x)$

where $f(x,y)$ is some positive function on $x$ and $y$ for instance $f(x,y) = \beta Prize_{A(y)}(y)$. Informally the definition means, that a transaction's ancestors are all converged promptly. Since the added condition is a static one, it sets the same ground truth for all nodes regardless of their local dynamic view of the system. One just has to to choose the function $f()$ wisely, e.g., to mirror depletion in the normal dynamic view of a long-running node.

**Definition 9** (Well-formed transaction)**.** *A transaction is inherently well-formed if the POSET including only it and its ancestors satisfies the following property.*

1. *The transactions are of the correct form, the mint and prize claimed are correct.*

2. *The payloads taken in the total order are all valid (according to the payload rules).*

3. *The parents all had prize available at the point when the transaction appears in the total order.*

4. *All transactions satisfy the promptly converged ancestors criterion.*

It is also possible to add other well-formedness criteria to change the properties of the system; however, the above are sufficient.

## 3.7 Conflict Resolution

Verifiers may develop slightly differing views of the current state of the system as it evolves, which can be formalised as saying that they will hold different real views $G_1, G_2, \ldots$, of some hypothetical true T–POSET $G$. This is due to transactions taking some time to propagate through the network, and will resolve by itself as long as no conflicts arise.

A conflict arises when two or more transactions $x_1 \in G_1$, $x_2 \in G_2$, etc., are published, such that there can be no single $G$ that contains them all. This would normally require a deliberate attack, such as double spending. The rule for conflict resolution is simply stated as: The highest well-formed transaction prevails (breaking ties deterministically).

## 3.8 Consensus

A verifier can share the network's consensus view of the current T–POSET $G$ of valid transactions by applying the following algorithm:

1. Collect the transactions from all participants, giving sets $G_1, G_2, \ldots, G_n$ and form $G = \bigcup_{i=1}^{n} G_i$ for some $n$.

2. From $G$, remove all ill-formed transactions that are permanently ignorable. Permanently ignorable transactions are those that could *never* become valid since they are not *well-formed*, e.g., because they carry an invalid signature, have two or more ancestors that mutually conflict, or carry an illegal transaction payload. These can be immediately discarded.

3. For the remaining well-formed transactions $G' \subseteq G$ that have neither been deemed valid or invalid:

   (a) Select the maximum-height well-formed transaction not yet classified, and classify it and all of its ancestors as valid.

   (b) Mark as invalid all other transactions that conflict with the validated transactions from Step 1.

This conflict resolution mechanism is a generalisation of the conflict resolution mechanism introduced in the Bitcoin Blockchain.

Maintaining consensus in this way requires incrementally recomputing the height of every transaction. For an $n$-node annotated directed acyclic graph, the trivial algorithm to compute the height of a node runs in $O(n)$ time, requiring $O(n^2)$ time to do the same for every node in the graph. Unfortunately, $O(n^2)$ would become too large for a network with a constantly expanding set of transactions. This is the purpose of the convergence mechanism, described in Theorem 3. Once the graph has converged the height calculation only needs to take into account the transactions after convergence.

### 3.8.1 Consensus in the Graphchain Model

Since the Graphchain model is different in structure from a blockchain, consensus is achieved slightly differently. Consensus refers to the agreement on a set of transactions, and we can say that the system has reached consensus on a certain transaction set. In a blockchain, typically transactions that are considered part of the consensus will be contained within a block that is in the main chain, but is a number of blocks back from the current leading block. Quite how many blocks back from the leading block is acceptable as consensus is subjective, though for Bitcoin 6 is proposed in order to gain maximum assurance. In practice however it is often lower. This means that the transactions are embedded in the system by a considerable amount of proof-of-work, to the degree that the transactions are immutable, and agreed on by all parties. The network has reached consensus.

The model for consensus in the Graphchain works in a very similar way. For any set of transactions, the more transactions that confirm a previous transaction, the more unlikely it is that it will somehow be removed. Thus a set

of transactions are considered to be part of the consensus if there is a current leading transaction that is a descendent of all of these transactions, and the transactions themselves are *sufficiently* encapsulated in proof-of-work, where the sufficiency is again subjective, though it can be quantified.

## 3.9 Mining and Minting

Minting is the process whereby the cryptocurrency token (or coin) supply is created. Coins are minted by the mining process included with a transaction. When creating a transaction, the transaction creator can assign a mint value prescribed by the system parameters. This value is determined from available data before forming the transaction, for a transaction $x_i$, it is calculated as:

$$\text{Mint}(x_i) = f(\text{Work}(x_i) \cdot \text{Height}(x_i)) \tag{9}$$

for some function $f$, for example $f(x) = \alpha \cdot x$ for some publicly known constant system parameter $\alpha = \alpha_1$.

While the intention is to describe the minting function Mint in such a way that it remains as flexible as possible, realistically for the choice of function $f$, within Mint, to be compatible with the design goals of this framework, there must be some feedback adjustment mechanism for $f$, similar to that of the value $\beta$ (Equation 7 and 8).

## 3.10 Monetary and Inflationary Policy

The framework is free from the choice of monetary policy, embodied by the minting function Mint. It can be either inflationary or deflationary, and rigidly fixed or adjustable by consensus or using an external mechanism. It is possible to target an arbitrary inflation schedule, using a decentralised feedback adjustment mechanism for $f$.

An important distinction from Bitcoin here is that miners can define their own reward based on how much work they are able to produce. The intention is that small miners can gain some small reward for their verification effort.

With this in mind, it is important to note the following:

- As long as the reward function $f$ is not super-linear, there is no incentive for miners to form Bitcoin-style mining pools.

- Sub-linear choices for $f$ would disproportionately reward smaller proofs-of-work. In this case, care should be taken in choosing a lower threshold for the amount of work required for a transaction to be included on the graph.
- If the function $f$ is expressed as $f(x) = \alpha \cdot x$ for some $\alpha \in \mathbb{R}$ and $\alpha$ is kept constant throughout the life of the system, then the coin supply will grow as the total verification work grows.
- Thus, for constant $\alpha$:
  - if the total work stays constant, e.g., after reaching an adoption plateau, inflation will be linear;
  - if the total work follows Moore's law, then inflation will follow the same doubling growth.
- It is possible to use a decentralised feedback adjustment mechanism for the choice of $f$. This could be achieved by letting $f(x_i) = \alpha_1 \cdot \text{Work}(x_i) + \alpha_2 \cdot \text{Height}(x_i)$ and adjusting the constant value $\alpha = (\alpha_1, \alpha_2)$ after certain height intervals have been surpassed. From the perspective of this work, this is the recommended method, as it allows for greater control over the inflationary measure which influences the security of the system.

## 3.11 Transaction Application Layer

The framework presented is essentially agnostic as to the mechanism used for creating and spending transactions, other than that there is some way to reward the miner. For example, the transactions can be signatures on messages assigning that assign tokens, as found in many currency cryptocurrencies. Alternatively, one could use Ethereum-like [32] transactions with a richer scripting language. The only requirement is that transactions provide a ledger-based value creation and transfer mechanism, in order to use incentives to drive the system security properties.

### 3.11.1 Rich payloads

When using transaction payloads which are Turing-complete and allow complicated smart contracts care must be taken. Namely, unlike the current prominent

examples, in Graphchain it is possible for transactions to both included in the POSET which do not know about each other at the time of creation. This means that the state of the system is not known when the instruction is given. Consider for example a smart contract which controls the flow of water from a dam; the flow down river is insufficient, so a controller orders the flow increased. However, unknowns to the controller several other controllers gave the same order in parallel and when all orders are received the compounding effect is that the flow is increased so much as to cause flooding downstream.

Fortunately, this issue can be easily rectified by introducing into the payload layer a requirement that the state of the smart-contract when the transaction is received is the same as when it was created and reject all transactions for which this is not the case.

# 4 Security and Properties

This section deals with the security and properties of the proposed graphchain framework. The properties that are important here are that the system can eventually arrive, and ideally quickly, on a consensus of what transactions have taken place and that double spending is prevented.

**Definition 10** (Generalised Proof-of-Work Scheme, Difficulty and Work)**.** *A generalised PoW Scheme is characterised by a function $S$ taking puzzle $a$ and solution $b$ as inputs and returning either true or false. A puzzle $a$ has computational difficulty $d$, if the expected number of computational steps required to find a solution $b$ satisfying $S(a, b)$ is $d$.*

*Furthermore we say that the work of finding a value $b$ such that $S(a, b)$ returns true is equal to $d$, and write* Work$((a, b)) = d$.

In practice, the most common PoW schemes used in modern cryptocurrencies are based on hash functions, where the difficulty is easily adjusted, verification is quick, and inputs can be arbitrary.

We can now analyse security by appealing to the incentive structures within the system. First, we need to make some assumptions on the participants, in keeping with the incentives from decentralised cryptocurrencies. The analysis focuses on incentivising players to cooperate. To that end we introduce rational players, which is a strictly stronger assumption than assuming honest ones.

### 4.0.1 Rational Players

A rational player adheres to the correctness rules of the protocol, but will deviate from them in order to gain greater reward. A rational miner seeks to maximise the value of their reward (from minting and fee collection alike) for any given amount of expended effort. A rational transactor acts rationally if they seek to ensure the acceptance of transactions in which they have the role of payer or payee. Rational players take on both roles.

### 4.0.2 Adversaries

An adversary seeks to disrupt the system, and may act in any way they wish. An adversary in this system can control up to 49% of the computational power within the system, and may create transactions. However, the adversary is not able to forge transactions on behalf of legitimate participants. Moreover, the adversary is not able to prevent rational players from seeing all transactions that are created.

The basis of the analysis assumes that the majority of the participants in the system act rationally.

Importantly, the focus is on the validation and consensus aspect of the security of the system. The security aspect of the transactions comes from the underlying primitives.

## 4.1 Double-Spending Resistance

A key priority is to ensure that a broadcast transaction quickly becomes agreed on by the majority of nodes, and cannot later be removed in some way. Once the transaction has gained enough weight, from the sum of its descendants' work, it can be deemed *impassible*: it will no longer be feasible or economical for an adversary to try to displace it.

For any two transactions $x_i$ and $x_j$, $x_i$ conflicts with $x_j$ if for any honest party $U$ accounting for incoming transactions, $U$ can accept either $x_i$ or $x_j$ but not both. There are a few ways for conflicts to occur, with the most obvious being that two transactions attempt to make payments from the same source, and of a transaction attempting to extract too much value from an insufficient source. Both are generalised as instances of double spending.

Theorem 1 shows that the weight, or total verification work accumulated on a transaction by it and its descendants, directly translates to its level of security.

**Theorem 1** (Double-Spending Resistance). *Let $G$ be a T–POSET, let $x_i$ be a transaction element in $G$, and denote by $c$ the weight of $x_i$ in the context of $G$. Let $\mathcal{A}$ be a probabilistic polynomial time adversary attempting to include a transaction $x_j$ that conflicts with $x_i$, and bounded by a maximum of $k$ computational steps. Assuming a rational majority, the probability that $\mathcal{A}$ can cause $x_j$ to displace $x_i$ in the majority consensus is approximately $k \cdot c^{-1}$.*

*Proof.* From the verification procedure, in order to replace transaction $x_i$, $\mathcal{A}$ needs to form $x_j$ with a greater total weight than $x_i$, as described in Definition 4. From Definition 10 it follows that for $k$ steps, the probability of succeeding is $k \cdot c^{-1}$ as required. □

This is simply stating that to displace a transaction, based on a system of rational players, you have to out-work the previous transaction. While this possibility remains available for a short while after the transaction is first broadcast, once it is picked up by other participants, it quickly becomes difficult to outwork the cumulative effort of the system as the weight will keep on growing. Next it is shown that under the incentive structure presented, transactions consolidate quickly.

## 4.2 Graph Consolidation

The scheme as envisaged provides an incentive to work on the latest transactions—the leading edge—which is important both for fast verification and for convergence.

**Theorem 2** (Leading-Edge Preference). *Let $G$ be a T–POSET, and let $x_1 \in G$ be a legitimate transaction forming a proof-of-work on a set of distinct transactions $X \subset P$. Any optimal strategy for a rational player will include $x_1$ within the next transaction.*

*Proof.* Let $v_1'$ represent the total prize of transaction $x_1$ and let $v_i$ represent the prize of transaction $x_i \in X$. By definition $v_1 \geq \sum^{|X|} v_i$. A rational player will prefer to reference $x_1$ over any transaction in $X$, as it maximises the prize

available for collection, thus offering greater expected reward for the amount of effort. □

## 4.3 Convergence

Now consider the time it would take for transactions published at a given time to all share a common descendent. Convergence, in this sense, means that at some point there will be a future transaction, $x_i$, which will be a common descendant of all the presently published acceptable transactions. With a rational majority of players, all such transactions will at some point share a common descendent. Fist it is necessary to define the properties of convergence.

**Definition 11** (Convergence). *Let $G$ be a T–POSET with $n$ published transactions. Let $x$ be a transaction in $G$ such that for all $g \in G$,* Height$(x) \geq$ Height$(g)$. *Let $G'$ be the set of transactions containing $x$ and all ancestors of $x$, and let $G''$ be the set of transactions that conflict with transactions in $G'$. If for every transaction $g' \in G'$,* Height$(x) \geq$ Height$(g')$, *then we say that $G'$ is converged at $x$. The set of transactions in $G''$ are said to be ignorable and the set of transactions $G''' = G \setminus (G' \cup G'')$ are said to be unconverged.*

From Definition 11, we see that for a graph of transactions there may be many converged subsets.

**Definition 12** (Divergence). *Let $G$ be a T–POSET with $n$ published transactions. We say that $G$ is diverged if it is not converged. Specifically, if there exists some set $G' \subseteq G$ such that for every $x_i \in G'$,* Height$(x_1) =$ Height$(x_2) = \ldots$, *for every $g \in G \setminus G'$,* Height$(x_1) >$ Height$(g)$ *and there exists $x_i$, $x_j$ where $x_i \neq x_j$ and both are conflicting.*

**Theorem 3** (Convergence). *Let $G$ be a T–POSET and let there be $n$ published transactions such that all transactions are valid, altogether non-conflicting in $G$. After some period of time $t$, $G$ plus all additional transactions will become converged.*

*Proof.* For all $n$ transactions in $G$, if there is one transaction such that all other transactions its ancestor, then $G$ is converged as required.

Otherwise, list all transactions in a list $L$ that do not have any descendants. Assume that there are $k$ distinct transactions of this type, and we know $k \leq n$.

None of the $k$ transactions can be the descendant of another such transaction in $L$ otherwise it would not be in $L$. By Theorem 2 we know the optimal selection for a rational participant, when creating a new transaction, is to select the $k$ transactions, as this will offer the greatest prize. The next transaction by any rational participant will collect all $k$ transactions in the list $L$. This transaction will be the common descendent of all transactions within $G$.

□

### 4.4 Strong Convergence

Not only is it the case that any given set of suitable transactions will soon share at least one common descendant. It can be shown that, at some later point, any further transaction will always be a descendant of the entire initial set.

**Theorem 4** (Strong Convergence). *Let $G$ be a T–POSET and let $G' \subseteq G$ be a subset of $n$ transactions in $G$, all valid, non-conflicting, and with no descendants in $G$. Assuming a majority of rational players, for any large enough superset $\tilde{G} \supset G$, then any future transaction that can be added to $\tilde{G}$ must be a descendant of all the transactions in $G'$.*

*Proof.* By Theorem 3, all transactions in $G'$ will eventually have a common descendant. At this point, this descendant transaction which we call $x_G$, along with all other descendant-less transactions will be candidates for inclusion by a following transaction. Let there be $n$ of these transactions. Now as in the strategy for proof from Theorem 3, all of these transactions will eventually converge. As more transactions are added, $x_G$ will eventually have no claimable prize remaining, thus all players will be forced to construct transactions on the descendants of $x_G$, which contains all of $G'$ as required. □

This property shows that, after a while, our T–POSET-based verification graph is as strong as a Bitcoin-style consolidated Blockchain: every new PoW reaffirming every sufficiently old transaction.

## 5 Discussion

This section forms a discussion of the properties of this system, highlighting the differences with Bitcoin (and generally all Blockchain-based cryptocurrencies).

### 5.0.1 Stability

As the system progresses, conflicting subgraphs may have appeared and branched out from the main graph. These subgraphs should eventually be discarded to avoid overloading the memory of the verifiers. However, they must persist for some period of time until it has become clear that the network consensus is that those conflicting transactions should be discarded, rather than the transactions they are conflicting with.

For example, if two conflicting transactions are relayed to different nodes, both nodes may later receive the other conflicting transaction. After some period, additional transactions will be broadcast predominantly confirming either one or the other of the conflicting transactions. Eventually, one transaction will pull ahead of the other in terms of weight, and as the difference increases the verifiers will switch to the graph of greatest height, reaching global consensus.

### 5.0.2 Liveness

An immediate consequence of the incentive structure is that the system will exhibit a liveness property, meaning that no otherwise valid, transaction will stay unverified and thus orphaned for long, before being incorporated into the mesh of converged transactions.

## 5.1 Attacks

One salient difference between the Blockchain and the Graphchain verification approach is the short-term vulnerability to attacks.

### 5.1.1 Casual Attacks

Casual attacks are individually motivated and relatively simple attacks, such as someone trying to form a double spend transaction. Bitcoin transactions are defenceless against casual attacks, until they get picked up onto the Blockchain (taking $\geq 10$ minutes in theory, and hours in reality due to congestion, which is only mitigated by paying a large fee). Even the commercial services that, for a fee, offer to guarantee not-yet-verified Bitcoin transactions, do no such thing; they merely offer an indemnification warranty to the recipient in case a conflicting transactions gets picked up instead.

The proposed framework closes this opportunity for casual attacks very quickly, because unverified transactions adding to the prize act as a magnet for their immediate parallel verification by multiple users.

### 5.1.2 Concerted Attacks

Concerted attacks are focused attempts to dislodge a specific transaction, using substantial computing power. The vulnerability profile of Blockchain transactions against concerted attacks is essentially the same as a casual attack: defenceless for a significant period until consolidated, then sharing the strength of the block that picked it up, which then increases as the chain predictably extends from there.

In a graphchain system, vulnerability decreases right away as verifications accumulate. Partially verified transactions retain temporary exposure to a concerted attack, since a powerful attacker may have the temporary local ability to overpower the honest majority by focusing all of its efforts against one specific target. On the other hand, once a transaction nears or reaches convergence, it will be as strongly affirmed as it would be in a blockchain system of equivalent total verification power.

Note that there is little rational value in using much energy to remove a previous transaction beyond its spend value (e.g., in a double-spending scenario; see Theorem 1), outside of attacks that seek to displace a transaction for other reasons.

### 5.1.3 Disruption and Denial of Service (DoS)

DoS attacks are attacks where attackers seek to disrupt the operation of the system much as possible, for example, by flooding the network with multiple small transactions with mutual conflicts, in an attempt to stall verifiers and clog the network.

Verifiers can employ simple heuristics based on the age and offered prize of a transaction to determine if it holds any value before seeking to determine or revise its validity. This is in fact precisely what the incentive structure recommends. By doing so, the system remains unclogged by flooding, unless the attackers are willing to put in an effort equivalent to a 51 percent attack.

In blockchain systems, by contrast, the bounded size of the blocks combined with a fixed renewal period creates a DoS vulnerability not present here: it is possible to cause the blocks to fill up by sending many small valid transactions, at the only cost of their transaction fee, to clog up legitimate transactions.

### 5.1.4 DoS hardening

In this section we describe the expected behaviour of a node in the network and actions it should take to harden itself against DoS attacks. For simplicity we consider a passive node, one that does not create any transactions.

A node in the network is connected to a relatively small (in the size of the entire network) number of peers. These peers tell the node about new transactions that are appearing in the network; when one of these peers informs the node of a new transaction, the node takes the following steps:

- It checks that the transaction is intrinsically well-formed (see Section 3.6.1)
  - If it is, the node checks that the transaction is compatible with it's existing view of the network,
    - if it is, the node updates its local view accordingly,
    - otherwise,
      - if transaction is higher in height then the current head, it reconstructs its local view form that transaction as described in consensus (see Section 3.8).
      - if the transaction is lower in height then the current head, it stores the transaction (if the transaction is sufficiently low in height and incompatible with the current local view the node will blacklist the sender).
  - if the transaction is not intrinsically well-formed the node discards the transaction and blacklists the sending peer (who may be corrupt).

It seems sensible to have some hesitation about changing the local view of system for insufficient reasons. In Graphchain this would be modeled by requiring the height advantage of the conflicting change to be sufficiently large with respect to the amount of local view with which it conflicts. For instance, a very shallow revision would be switched for any height advantage but one displacing the entire

contents of the time-to-live window would require a significant high advantage. At present, we exclude this from the DoS hardening but if switching seems sufficiently expensive, this decision should be revisited.

### 5.1.5 Splitting the Graph

An attacker attempts to fork the T-POSET by simultaneously broadcasting two conflicting transactions of equal height, then actively working to keep both branches level to avoid convergence.

In the worst case the network will be exactly split in two, with one side selecting one transaction as valid and the other as invalid. It will then be a race for one side to increase the weight of their branch. Once one side moves ahead, the entire network will consider this branch to be the true set. In order to keep the system in a non-ordered state, the adversary has the challenge of attempting to make the other branch higher, faster than the entire network can achieve this.

### 5.1.6 Pre-mined double spending

One attack that is potentially possible on a Graphchain-style system that is not possible on linear system is the pre-mined assisted double spending. In this section we will describe the attack and describe why it does not work. We will use the notation of *ConfirmationTime* to denote the the time a party waits before they are satisfied that the transaction will be included in the history. We will denote *ConfirmationRate* to denote the rate of honest verification work per time unit and *AdversaryRate* to denote the adversary verification work per time unit.

The attack is used by an adversary with less then 50% mining power, the adversary has two conflicting transactions and wants a party to believe one transaction is included but then switch the network state, effectively double spending. The attack begins by the adversary posting one of the conflicting transaction and then waiting out the *ConfirmationTime*. The adversary having now received their benefit posts the conflicting transaction and the additional (previously hidden) transactions they have pre-mined.[1] Against a general DAG based approach this attack will succeed provided that the adversary has pre-mined for time greater then *ConfirmationRate*ConfirmationTime/AdversaryRate*.

[1]For this to work it is crucial that the additional hidden transactions reference the conflicting transaction not the other way around.

*This attack works on DAG based approaches but not chain based because in the chain based approach the adversary's pre-mining cannot be combined with the honest work produced in the same time period; in contrast, in DAG based approaches both the honest work and hidden work can be combined into a coherent history, provided they are non-conflicting.*

Graphchain prevents this attack by careful choices of dynamic incentives and static checks. Recall that a transaction is only well formed in Graphchain if all its ancestors have promptly converged, because of this pre-mining transactions and not forwarding them to the network will create transactions which cannot ever be part of the same POSET as honestly created transactions. Therefore it suffices to let the *ConfirmationTime* be greater then the $(\frac{AdversaryRate}{ConfirmationRate-AdversaryRate}) *$ time-to-live parameter to completely prevent this attack.[2]

## 5.2 Practical Considerations

This section addresses the practical consideration that must be made when creating a graphchain based system. These considerations are outside the scope of the proposal here, as they represent design decisions for implementation. Nonetheless, they can be important for a practical cryptocurrency proposal.

### 5.2.1 Transaction Size and Cost

Any implementation would expect or even require that the cost or fee to be posted by a new transaction, directly reflect the total bit-size of the transaction. Fee has the added benefit of making the transaction more desirable to include quickly, and so higher fees can attract faster verification. Choosing whether such a rule should be hard-coded in the system, or softly expected by the verifiers as a natural consequence of the game theoretic incentives, is left as an implementation decision.

[2]Even without this criterion, one can choose the dynamic requirement that any competing conflict must add additional height greater then that completed by the honest network in the time since the hidden DAG branched off; this, also, would suffice to thwart this attack.

### 5.2.2 Bootstrapping and Early Adoption

In order to instantiate the system, origin transactions must be created, which can come loaded with an initial prize for collection. It will also be necessary to select a minting function, which will determine the rate of inflation and hence the inherent incentives, or disincentives, to early adopters.

### 5.2.3 Scalable Throughput and Responsiveness

Unlike Bitcoin and all other blockchained cryptocurrencies, there is no cap on the number of transactions verifiable in any period of time. Since transactions verify each other, a surge in transactions broadcast will be instantly met with an equal surge in verification response.

We note that Bitcoin is mired in a debate concerning the total size of transactions that can appear in a single block [44], an exclusively blockchain-model problem.

### 5.2.4 Light and Thin Clients

Having to collect and order transactions may seem overly burdensome for thin clients, such as clients on a smartphone. While implementation decisions such as these are not the main focus here, note that in blockchain systems, thin clients only use recent block headers. The same principle can be applied here, in that once the system has reached convergence, a thin client can store this state. It is then possible to verify a transaction is in the state by requesting and checking the other transactions in the tree to any transaction in question.

# 6 Simulation and Experimental Results

This work has developed a simulator to check that a graphchain system can handle the same throughput of transactions as a blockchain system. This was designed to experimentally ascertain whether that the proposed algorithms induce the right system behaviour. The implementation is written in the programming language Rust [41], and relies on assumptions for the incentives of the graphchain system. An interesting observation is that the simulation reaches convergence quickly even if the parents for a new transaction are chosen randomly.

| LEN | NO | RTE | DLY | ATL | ACT | CPUT |
|---|---|---|---|---|---|---|
| 10hr | 5000 | 0.14 per s | 10s | 1000s | 10s | 1s |
| 10hr | 10000 | 0.28 per s | 10s | 1000s | 10s | 3s |
| 10hr | 20000 | 0.56 per s | 10s | 1000s | 10s | 16s |
| 10hr | 30000 | 0.83 per s | 5s | 500s | 5s | 2s |
| 10hr | 40000 | 1.11 per s | 5s | 500s | 5s | 3s |
| 1hr | 10833 | 3 per s | 6.2s | 620s | 6.2s | 31s |
| 2hr | 21666 | 3 per s | 6.2s | 620s | 6.2s | 62s |
| 24hr | 21666 | 3 per s | 6.2s | 620s | 6.2s | 744s |

Table 1: Graphchain Simulation Results

Table 1 provides the data for several simulations, indicating for each: the simulated time length of the execution (LEN), the final number of nodes, or miners (NO), the average transaction arrival rate (RTE), the incurred delay (DLY), the average time to live until a transaction becomes depleted (ATL), the average time until convergence (ACT) and the real CPU time taken by the simulation (CPUT). All times are using the clock of the simulation, except for the CPU time which is in real seconds. Times are displayed in seconds and hours. Of particularly interest are the last three rows (data rows 6, 7 and 8) where the simulation has been tuned to have an equivalent transaction rate as the current Bitcoin network. The reference transaction rate is the peak Bitcoin rate of 260,000 transactions over a day on Monday 7 November 2016, using data from `coindesk.com`. This transaction rate has been adjusted for simulation length in order to achieve the same median network delay, and is calculated from data available at `bitcoinstats.com` on the same day. The upshot is that the simulation, running on a single i7-4770 CPU was able to match the transaction rate of the current Bitcoin network and verify all transactions in almost real time.

# 7 Conclusion

The primary objective of this whitepaper was to establish an alternative way of creating a distributed cryptocurrency that avoids the bottlenecks and cen-

tralisation issues that come packaged with blockchain implementations. This is achieved by redesigning the foundation layer, in favour of a more natural decentralised verification process. This whitepaper has demonstrated that this process still ensures that all new verification effort secures every previous transaction, after a brief period of convergence for each transaction. By creating a framework in this way, the aim is to get closer to the moral of a decentralised cryptocurrency which is still found wanting in current implementations.

This research focuses on removing the block construct from cryptocurrencies in order to improve the decentralised aspects. This is achieved by promoting collaboration over competition and forming transactions into a directed acyclic graph. Further, it is demonstrated under the assumption of a majority of rational actors, that the system will eventually converge, and will adopt similar security properties ot Bitcoin in these converged states.

## References


[1] Dan Godin Ars Technica. Bitcoin security guarantee shattered by anonymous miner with 51 network power. `http://arstechnica.com/security`, 2014. [Accessed: January 18].

[2] Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun. Bitter to better - how to make Bitcoin a better currency. In Angelos D. Keromytis, editor, *FC 2012*, volume 7397 of *LNCS*, pages 399–414, Kralendijk, Bonaire, February 27 – March 2, 2012. Springer, Heidelberg, Germany.

[3] Iddo Bentov, Pavel Hubáček, Tal Moran, and Asaf Nadler. Tortoise and hares consensus: the meshcash framework for incentive-compatible, scalable cryptocurrencies. Cryptology ePrint Archive, Report 2017/300, 2017. `http://eprint.iacr.org/2017/300`.

[4] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. SoK: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy*, pages 104–121, San Jose, CA, USA, May 17–21, 2015. IEEE Computer Society Press.

[5] Xavier Boyen, Christopher Carr, and Thomas Haines. Blockchain-free cryptocurrencies. A rational framework for truly decentralised fast transactions. Cryptology ePrint Archive, Report 2016/871, 2016. `http://eprint.iacr.org/2016/871`.

[6] Xavier Boyen, Christopher Carr, and Thomas Haines. Graphchain: a blockchain-free scalable decentralised ledger. In Satya V. Lokam, Sushmita Ruj, and Kouichi Sakurai, editors, *Proceedings of the 2nd ACM Workshop on Blockchains, Cryptocurrencies, and Contracts, BCC@AsiaCCS 2018, Incheon, Republic of Korea, June 4, 2018*, pages 21–33. ACM, 2018.

[7] Miles Carlsten, Harry A. Kalodner, S. Matthew Weinberg, and Arvind Narayanan. On the instability of bitcoin without the block reward. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 154–167, Vienna, Austria, October 24–28, 2016. ACM Press.

[8] Christopher Carr, Colin Boyd, Xavier Boyen, and Thomas Haines. Bitcoin unchained. *ERCIM News*, 2017(110), 2017.

[9] David Chaum. PrivaTegrity: online communication with strong privacy. Presentation at Real-World Crypto, Stanford University, 2016.

[10] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed E. Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, Dawn Song, and Roger Wattenhofer. On scaling decentralized blockchains - (A position paper). In Jeremy Clark, Sarah Meiklejohn, Peter Y. A. Ryan, Dan S. Wallach, Michael Brenner, and Kurt Rohloff, editors, *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*, volume 9604 of *Lecture Notes in Computer Science*, pages 106–125. Springer, 2016.

[11] George Danezis and Sarah Meiklejohn. Centrally banked cryptocurrencies. In *NDSS 2016*, San Diego, CA, USA, February 21–24, 2016. The Internet Society.

[12] Christian Decker and Roger Wattenhofer. Information propagation in the Bitcoin network. In *13th IEEE International Conference on Peer-to-Peer Computing, IEEE P2P 2013, Trento, Italy, September 9-11, 2013, Proceedings*. IEEE, 2013.

[13] Christian Decker and Roger Wattenhofer. A fast and scalable payment network with Bitcoin duplex micropayment channels. In Andrzej Pelc and Alexander A. Schwarzmann, editors, *Stabilization, Safety, and Security of Distributed Systems - 17th International Symposium, SSS 2015, Edmonton, AB, Canada, August 18-21, 2015, Proceedings*, volume 9212 of *Lecture Notes in Computer Science*. Springer, 2015.

[14] Ethan Heilman and Neha Narula and Thaddeus Dryja and Madars Virza. IOTA vulnerability report: Cryptanalysis of the Curl hash function enabling practical signature forgery attacks on the iota cryptocurrency. `https://github.com/mit-dci/tangled-curl/blob/master/vuln-iota.md`, 2017. [Accessed: January 18].

[15] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert van Renesse. Bitcoin-NG: A scalable blockchain protocol. In Katerina J. Argyraki and Rebecca Isaacs, editors, *13th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2016, Santa Clara, CA, USA, March 16-18, 2016*. USENIX Association, 2016.

[16] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In Nicolas Christin and Reihaneh Safavi-Naini, editors, *FC 2014*, volume 8437 of *LNCS*, pages 436–454, Christ Church, Barbados, March 3–7, 2014. Springer, Heidelberg, Germany.

[17] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 281–310, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[18] Arthur Gervais, Ghassan O. Karame, Vedran Capkun, and Srdjan Capkun. Is Bitcoin a decentralized currency? *IEEE Security & Privacy*, 12(3):54–60, 2014.

[19] IOTA. http://iota.org/, 2016. [Accessed: January 18].

[20] Benjamin Johnson, Aron Laszka, Jens Grossklags, Marie Vasek, and Tyler Moore. Game-theoretic analysis of DDoS attacks against bitcoin mining pools. In Rainer Böhme, Michael Brenner, Tyler Moore, and Matthew Smith, editors, *FC 2014 Workshops*, volume 8438 of *LNCS*, pages 72–86, Christ Church, Barbados, March 7, 2014. Springer, Heidelberg, Germany.

[21] Ghassan Karame, Elli Androulaki, and Srdjan Capkun. Double-spending fast payments in bitcoin. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 2012*, pages 906–917, Raleigh, NC, USA, October 16–18, 2012. ACM Press.

[22] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. Enhancing Bitcoin security and performance with strong consistency via collective signing. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.* USENIX Association, 2016.

[23] Sergio Demian Lerner. DagCoin Draft. https://bitslog.files.wordpress.com/2015/09/dagcoin-v41.pdf, 2015. [Accessed: January 18].

[24] Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. Inclusive block chain protocols. In Rainer Böhme and Tatsuaki Okamoto, editors, *FC 2015*, volume 8975 of *LNCS*, pages 528–547, San Juan, Puerto Rico, January 26–30, 2015. Springer, Heidelberg, Germany.

[25] Sarah Meiklejohn. Top ten obstacles along distributed ledgers path to adoption. *IEEE Security & Privacy*, 16(4):13–19, 2018.

[26] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. A fistful of Bitcoins: characterizing payments among men with no names. *Commun. ACM*, 59(4), 2016.

[27] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed E-cash from Bitcoin. In *2013 IEEE Symposium on*

*Security and Privacy*, pages 397–411, Berkeley, CA, USA, May 19–22, 2013. IEEE Computer Society Press.

[28] Andrew Miller, Ari Juels, Elaine Shi, Bryan Parno, and Jonathan Katz. Permacoin: Repurposing bitcoin work for data preservation. In *2014 IEEE Symposium on Security and Privacy*, pages 475–490, Berkeley, CA, USA, May 18–21, 2014. IEEE Computer Society Press.

[29] Andrew Miller, Ahmed E. Kosba, Jonathan Katz, and Elaine Shi. Nonoutsourceable scratch-off puzzles to discourage bitcoin mining coalitions. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 680–691, Denver, CO, USA, October 12–16, 2015. ACM Press.

[30] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. Available at `https://bitcoin.org/bitcoin.pdf`.

[31] Arvind Narayanan, Joseph Bonneau, Edward W. Felten, Andrew Miller, and Steven Goldfeder. *Bitcoin and Cryptocurrency Technologies - A Comprehensive Introduction*. Princeton University Press, 2016. ISBN: 978-0-691-17169-2.

[32] Ethereum Network. Ethereum: Smart contract and decentralized application platform. `https://github.com/ethereum/wiki/wiki/White-Paper`, 2013. [Accessed: January 18].

[33] Sunoo Park, Krzysztof Pietrzak, Albert Kwon, Joël Alwen, Georg Fuchsbauer, and Peter Gaži. Spacemint: A cryptocurrency based on proofs of space. Cryptology ePrint Archive, Report 2015/528, 2015. `http://eprint.iacr.org/2015/528`.

[34] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 643–673, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany.

[35] Joseph Poon and Thaddeus Dryja. The Bitcoin lightning network: Scalable off-chain instant payments. `https://lightning.network/lightning-network-paper.pdf`, 2016. [Accessed: January 18].

[36] Serguei Popov. IOTA: The tangle. http://iotatoken.com/IOTA_Whitepaper.pdf, 2017. [Accessed: January 18].

[37] David Schwartz, Noah Youngs, and Arthur Britto. The Ripple protocol consensus algorithm, 2014. [Accessed: January 18].

[38] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. SPECTRE: A fast and scalable cryptocurrency protocol. Cryptology ePrint Archive, Report 2016/1159, 2016. http://eprint.iacr.org/2016/1159.

[39] Yonatan Sompolinsky and Aviv Zohar. Accelerating Bitcoin's transaction processing. Fast money grows on trees, not chains. Cryptology ePrint Archive, Report 2013/881, 2013. http://eprint.iacr.org/2013/881.

[40] Florian Tschorsch and Björn Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys and Tutorials*, 18(3):2084–2123, 2016.

[41] Web. Rust. https://www.rust-lang.org/en-US, 2010. [Accessed: June 18].

[42] Web. Litecoin. https://litecoin.org/, 2011. [Accessed: January 18].

[43] Web. Dogecoin. http://dogecoin.com/, 2013. [Accessed: January 18].

[44] Web. BitcoinXT. https://bitcoinxt.software/, 2016. [Accessed: January 18].

[45] Web. Hashgraph: The future of decentralised technology. https://hashgraph.com, 2017. [Accessed: January 18].