2nd International Conference on Blockchain Economics, Security and Protocols

Tokenomics 2020, October 26–27, 2020, Toulouse, France

Edited by

Emmanuelle Anceaume
Christophe Bisière
Matthieu Bouvard
Quentin Bramas
Catherine Casamatta

OASICS

Editors

Emmanuelle Anceaume
CNRS, IRISA, Rennes, France
Emmanuelle.Anceaume@irisa.fr

Christophe Bisière
Toulouse School of Economics, University Toulouse Capitole, TSM-R, France
christophe.bisiere@tse-fr.eu

Matthieu Bouvard
Toulouse School of Economics, University Toulouse Capitole, TSM-R, France
matthieu.bouvard@tse-fr.eu

Quentin Bramas
ICUBE, University of Strasbourg, France
bramas@unistra.fr

Catherine Casamatta
Toulouse School of Economics, University Toulouse Capitole, TSM-R, France
catherine.casamatta@tse-fr.eu

ACM Classification 2012
Computing methodologies → Distributed algorithms; Security and privacy → Distributed systems security; Applied computing → Economics

ISBN 978-3-95977-157-3

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at https://www.dagstuhl.de/dagpub/978-3-95977-157-3.

Publication date
February, 2021

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at https://portal.dnb.de.

Digital Object Identifier: 10.4230/OASIcs.Tokenomics.2020.0
ISBN 978-3-95977-157-3 **ISSN 1868-8969** **https://www.dagstuhl.de/oasics**

OASIcs – OpenAccess Series in Informatics

ISSN 1868-8969

https://www.dagstuhl.de/oasics

Contents

Preface

This volume includes the published papers of Tokenomics 2020, the second edition of the International Conference on Blockchain Economics, Security and Protocols.

Tokenomics is an international forum for theory, design, analysis, implementation and applications of blockchains and smart contracts. The goal of the conference is to bring together economists, computer science researchers and practitioners working on blockchains in a unique program featuring outstanding invited talks and academic presentations.

The conference was initially planned on May 11th and 12th, 2020. Due to the COVID-19 pandemic, it eventually took place on October 26th and 27th of the same year at Toulouse School of Economics (TSE) in an hybrid format with some of the speakers and moderators presenting in front of an audience in TSE new buidling and the other conference participants joining through videoconferencing.

For this second edition, there were 45 papers submitted: 12 papers in computer science (11 as regular papers and 1 as a short paper) and 32 papers in economics. The economics program committee selected 14 papers for presentation at the conference. The computer science program committee selected 6 regular papers for presentation at the conference and publication in this volume. Additionally, 4 submissions were accepted as short papers, for presentation at the conference and publication in this volume.

In addition to accepted papers, we had the pleasure to welcome four distinguished invited keynote speakers:

- Ittai Abraham, senior researcher at vmware research. Abraham discussed the use of game theoretical tools into computer science to model blockchains and cryptocurrencies
- Long Chen, Secretary-General of the Luohan Academy, an open research institute initiated by Alibaba, and former Chief Strategy Officer at Ant Financial. Chen gave an overview of the ongoing changes in financial services driven by progresses in information technologies.
- Jean Tirole, Researcher at TSE and 2014 laureate of the Sveriges Riksbank prize in economic sciences in memory of Alfred Nobel. Tirole discussed the challenges faced by cryptocurrencies using the framework of economic theory.
- Timothy Zakian, Software Engineer, Novi, Facebook. Zakian presented Move, the programming language developed to implement transactions and smart contracts on the Libra blockchain.

Together with the computer science contributions gathered in these proceedings, the papers presented in the economics track tackled a wide range of issues reflecting the vitality of the research on blockchains and cryptocurrencies in economics. This growing interest reflects the current and potential impact of blockchain-based applications for consumers, businesses and governments. It also captures a fundamental feature of blockchains: implementing a distributed consensus is as much an incentive problem as it is a technological challenge.

A first subset of these papers focuses on the functioning of the blockchain itself. In keeping with an earlier stream of papers in computer sciences and economics, Ebrahimi, Routledge and Zetlin-Jones ("Getting Blockchain Incentives Right") use game theory to analyze miners' equilibrium strategies under proof of work and the possibility they may fail to ensure consensus. Amoussou-Guenou, Biais, Potop-Butucaru and Tucci-Piergiovanni ("Rational vs Byzantine Players in Consensus-based Blockchains") use a similar game-theoretic toolbox to analyze the strategies of committee members in a Byzantine Fault Tolerant blockchain. Garatt and van Oordt ("Why Fixed Costs Matter for Proof-of-Work Based Cryptocurrencies") show how miners' cost structure, notably the existence of sunk equipment costs, affect their incentives to deploy hashpower in response to the cryptocurrency price movements. Finally, Hinzen, John and Saleh ("Bitcoin's Fatal Flaw: The Limited Adoption Problem") model how the information time lags inherent to fully distributed consensus create a hard technical constraint on the throughput of permissionless blockchains.

A second set of papers adopts an industrial organization approach to understand the distinctive features of blockchain-powered businesses. Lyandres ("Product Market Competition with Crypto Tokens and Smart Contracts") shows how native tokens and smart contracts alter the nature of the competition between an incumbent firm and a potential entrant. Cong, He and Wang ("Token-Based Platform Finance") combine the industrial organization angle with corporate finance implications: they evaluate the dual role of tokens as influencing users' adoption of a platform, and as providing entrepreneurs with a source of funding. Finally, Bakos and Halaburda ("When Do Smart Contracts and IoT Improve Efficiency? Automated Execution vs. Increased Information") draw on contract theory to clarify the capabilities of smart contracts. In particular, they distinguish between two features of smart contracts, the expansion of the contracting space thanks to IoT sensors and the automatization of the contract execution.

Last, a third set of papers approaches tokens from an asset pricing side. Pratt, Danos and Marcassa ("Reversible and Composable Financial Contracts") show how the value of utility tokens can be derived from users' benefits from immediately accessing the services of a platform. Dai, Jiang, Kou and Qin ("From Hotelling to Nakamoto: The Economic Meaning of Bitcoin Mining") propose a model that relates Bitcoin prices to miners' decisions to warehouse or sell the bitcoins they earn by confirming blocks. This model delivers quantitative predictions and is calibrated to the data. Shams ("The Structure of Cryptocurrency Returns") studies the comovement of multiple cryptocurrency prices and empirically connects high correlations to common demand factors. Finally, Benigno, Schilling and Uhlig ("Cryptocurrencies, Currency Competition, and the Impossible Trinity") study the implications of the adoption of a global cryptocurrency for monetary policies and exchange rates between fiat moneys.

Overall, the breadth of the topics explored by the participants to this conference illustrates the fruitful interaction between computer science and economics for understanding the implications of blockchain-based solutions. It also suggests much more ground to cover and we hope this conference will further stimulate research in this area.

We thank the authors for submitting their work at the conference and the program committee who worked hard in reviewing papers and giving feedback to the authors.

Catherine, Christophe, Emmanuelle, Matthieu and Quentin

Tokenomics 2020 Organization

General Chairs

Emmanuelle Anceaume, CNRS, Irisa (France)
Christophe Bisière, University Toulouse Capitole, TSE and TSM-R (France)
Matthieu Bouvard, University Toulouse Capitole, TSE and TSM-R (France)
Quentin Bramas, ICUBE, University of Strasbourg (France)
Catherine Casamatta, University Toulouse Capitole, TSE and TSM-R (France)

Program Committee

Computer Science

Emmanuelle Anceaume, CNRS, Irisa (France)
Daniel Augot, INRIA, Ecole Polytechnique (France)
Quentin Bramas, ICUBE, University of Strasbourg (France)
Vincent Danos, CNRS, Ecole Normale Supérieure (France)
Giuseppe Antonio Di Luna, Sapienza University of Rome (Italy)
Antonio Fernández Anta, IMDEA Networks (Spain)
Fabrice Le Fessant, OCaml PRO (France)
Juan A. Garay, Texas A&M University (USA)
Chryssis Georgiou, University of Cyprus (Cyprus)
Vincent Gramoli, The University of Sydney (Australia)
Braham Hamid, IRIT (France)
Maurice Herlihy, Brown University (USA)
Pascal Lafourcade, Université Clermont Auvergne (France)
Mario Larangeira, IOHK, Tokyo Institute of Technology (Japan)
Romaric Ludinard, IMT Atlantique (France)
Maria Potop-Butucaru, Sorbonne Université (France)
Leonardo Querzoni, Sapienza University of Rome (Italy)
François Taiani, Université Rennes 1, Irisa (France)
Sara Tucci-Piergiovanni, CEA LIST (France)
Marko Vukolic, IBM Research - Zurich (Switzerland)
Josef Widder, Interchain Foundation & TU Wien (Austria)

Economics

Bruno Biais, HEC Paris (France)
Christophe Bisière, University Toulouse Capitole, TSE and TSM-R (France)
Matthieu Bouvard, University Toulouse Capitole, TSE and TSM-R (France)
Catherine Casamatta, University Toulouse Capitole, TSE and TSM-R (France)
Jonathan Chiu, Bank of Canada (Canada)
Will Cong, Cornell University, Johnson Graduate School of Management (USA)
Guillaume Haeringer, Baruch College, Zicklin School of Business (USA)
Hanna Halaburda, New York University and Bank of Canada (USA & Canada)
Zhiguo He, University of Chicago, Booth School of Business (USA)

Some Economics of Fintech

Jean Tirole
Toulouse School of Economics, France
https://www.tse-fr.eu/fr/people/jean-tirole

Abstract

The contours of digital payments are still in the making. Recent years have seen the emergence of new instruments best exemplified by public cryptocurrencies like Bitcoin or Big Tech payment systems like Alipay. These developments in the private sector have in turn fueled discussions and projects around the creation of central bank digital currencies. Digital currencies have a lot to offer. They can provide consumers with user-friendly low-cost means of payment and facilitate the integration of payment systems across borders. They may also offer alternatives in countries with dysfunctional national monetary systems. On the supply side, private digital currencies can be a source of funding (e.g., initial coin offerings) and allow businesses to retain consumers and to collect information. Which form of digital currency will eventually prevail has yet to be seen. Popular permissionless cryptocurrencies lack in their current form the price stability necessary to serve as a store of value: accepting a payment in Bitcoin exposes a merchant to costly financial risk. Stable coins pegged to a central-bank currency and backed by safe collateral are an attempt to dim excess volatility (e.g., Tether or Libra). But this guarantee creates new challenges: collateral must be segregated and prudentially supervised to ensure consumer protection. It is unclear which authority would have the capacity and incentives to provide that supervision for a global digital currency. More generally, a private global digital currency would raise a range of public policy issues ranging from tax fraud and money laundering control, to loss of seignorage revenue, impediments to monetary policy and potential threat to financial stability. In that context, Central Bank Digital Currencies (CBDC) may provide a solution that combines the convenience of private digital money with the institutional support of a state. But the scope of a CBDC's deployment needs to be carefully calibrated: a CBDC directly held by wholesale or retail depositors would compete with bank deposits, possibly limiting banks' ability to engage in their essential function of maturity transformation through long-term credit. Overall, the deployment of new technologies for payments has the potential to create meaningful value for consumers. However, technological disruption does not upend the fundamental economic principles that have shaped our financial systems and its regulatory framework. Applying these principles may be our best chance to understand the ongoing Fintech revolution.

2012 ACM Subject Classification Social and professional topics; Social and professional topics → Economic impact

Keywords and phrases Cryptocurrency, Stable Coin, Central Bank Digital Currency, Fintech, Financial Regulation

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2020.1

Category Invited Talk

When Nakamoto Meets Nash: Blockchain Breakthrough Through the Lens of Game Theory

Ittai Abraham
VMware Research, Herzliya, Israel
https://research.vmware.com/researchers/ittai-abraham
iabraham@vmware.com

Abstract

We discuss the deep connections between Blockchain Technology, Computer Science and Economics. The talk surveys the ways the Blockchain disruption raises fundamental challenges that have a deep game theoretic nature. We focus on four major open questions:

1. The need for a game theoretic *endogenous* theory of the utility of Money Systems that can model friction, fairness, and trust.
2. The need to incentivize trust in both *consensus* and *execution.* A need for a game theoretic theory of Consensus and analogue to Byzantine Fault Tolerance. A need for a game theoretic framework for *scalable validation.*
3. The challenge of incentivizing *fairness* and *chain quality.* Can we use notions of robust equilibrium to provide better notions of fairness?
4. The open question of how Blockchains can incentivise *welfare.* The need for a theory of Blockchains as *public goods.*

2012 ACM Subject Classification Theory of computation → Algorithmic game theory; Theory of computation → Distributed algorithms

Keywords and phrases Game theory, Consensus, Blockchain

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2020.2

Category Invited Talk

Digital Currencies as Types

Timothy A. K. Zakian
Novi
https://tzakian.github.io/
tzakian@fb.com

Abstract

Linear types have been well studied since their inception by Girard; a linear value can be moved from one place to another, but can never be copied or forgotten. From its inception Move – a new programming language developed to implement custom transactions and smart contracts on the Libra Blockchain – has had values–or resources–that behave in this linear manner as a central part of its semantics. On the Libra Blockchain, Move enables significant parts of the Libra protocol, including the Libra Coins, transaction processing, and validator management. In this talk, we will look at how different digital assets are represented with Move on the Libra Blockchain.

In the process of exploring the representation of digital assets on-chain in Move, we will revisit one of the first examples used in the paper that introduced linear logic; that of payments, and encounter other ideas from programming languages along the way, such as type-indexed data types and code modularity. We will see how we can leverage these ideas to provide strong guarantees of key asset properties such as losslessness, value conservation, and explicit representation of an asset, its currency, and its value.

As we explore the implementation of a digital asset in Move, we will see how, in Move, code is organized into a number of different modules, with each module consisting of resources and functions that can be used with the resources defined in that module. This gives rise to a type of strong encapsulation around the resources defined within a Move module: only functions within the module that define the resource can create, destroy, or access the fields of that resource.

We will see how representing a digital asset as a resource, coupled with this strong encapsulation, and privileging the creation and destruction operations within the module means that we can build a digital asset representation on-chain that is lossless by design: wherever it may go on-chain, such a digital asset cannot ever be "lost" or accidentally forgotten, and, no new digital assets can be created on-chain without the correct privileges.

We can then index this digital asset resource that we've built in Move by a type-level representation of each currency in the system to arrive at an explicit static representation of the currency of a digital asset. This representation statically disallows entire classes of possible issues, such as trying to combine two assets in different currencies, while still preserving all of the properties that we previously had, such as losslessness.

With this representation of a digital asset that we have built in Move, we can also test and verify that the value of the digital assets on-chain are preserved outside of creation and destruction operations; since the only functions that can change the value of an asset must be defined within the same module we can heavily test, and in fact verify, that these functions preserve the value of any digital assets that they may interact with. At the end of this process we arrive at a testable, verifiable, and explicit representation of a digital asset in Move that is lossless, conserves value, and represents its currency and value explicitly.

2012 ACM Subject Classification Theory of computation → Semantics and reasoning; Theory of computation → Linear logic; Theory of computation → Type theory

Keywords and phrases Digital Currencies, Linear Types, Move, Blockchains

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2020.3

Category Invited Talk

On Fairness in Committee-Based Blockchains

Yackolley Amoussou-Guenou
Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France
Sorbonne Université, CNRS, LIP6, F-75005 Paris, France

Antonella Del Pozzo
Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France

Maria Potop-Butucaru
Sorbonne Université, CNRS, LIP6, F-75005 Paris, France

Sara Tucci-Piergiovanni
Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France

Abstract

Committee-based blockchains are among the most popular alternatives of *proof-of-work* based blockchains, such as Bitcoin. They provide strong consistency (no fork) under classical assumptions, and avoid using energy-consuming mechanisms to add new blocks in the blockchain. For each block, these blockchains use a committee that executes Byzantine-fault tolerant distributed consensus to decide the next block they will add in the blockchain. Unlike Bitcoin, where there is only one creator per block, in committee-based blockchain any block is cooperatively created. In order to incentivize committee members to participate in the creation of new blocks, rewarding schemes have to be designed. In this paper, we study the fairness of rewarding in committee-based blockchains and we provide necessary and sufficient conditions on the system communication under which it is possible to have a fair reward mechanism.

2012 ACM Subject Classification Computer systems organization → Dependable and fault-tolerant systems and networks

Keywords and phrases Blockchain, Consensus, Committee, Fairness, Proof-of-Stake, Reward, Selection

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2020.4

Acknowledgements The authors thank Ludovic Desmeuzes for his work on the numerical examples.

1 Introduction

The blockchain technology is one of the most appealing technology since its introduction in the Bitcoin White Paper [31] in 2008. A blockchain is a distributed ledger, where information are stocked in blocks, and hashes link blocks in order to have a chain structure. Blockchain systems mostly use proof-of-work, where the first process that solves a crypto-puzzle can add a new block to the blockchain. First, this technique is highly energy consuming, and second, it does not ensure consistency, *i.e.* conserving the chain structure. Forks may happen and they lead to a tree structure. Some alternatives arisen to avoid at least one of these issues. For example, proof-of-stake based blockchains, where the probability of being able to add a block depends on the stake of a process; this alternative solves the energy consumption issue, but not the consistency one; they are also subject to the *nothing-at-stake* problem, where processes try to produce blocks on all the forks to ensure some rewards, and by doing so, do not resolve the forks. However, in [35], Saleh shows that the nothing-at-stake problem is mitigated. Other alternatives tackle both issues, for example, committee-based blockchains. In committee-based blockchains, for each height/block, a committee is selected and that committee uses a consensus algorithm to decide on the next block to append in the blockchain. By construction, committee-based blockchains are not subject to nothing-at-stake, since they ensure consistency (no fork).

To motivate processes to add and maintain the blockchain, rewarding mechanisms are in place. Because committee members can be faulty, rewarding mechanisms are inherently more complex to handle and their properties must be studied. Ad minimum, the rewarding mechanism must be fair, *i.e.* distributing the rewards in proportion to the merit of participants, where *merit* abstracts the notion of effort processes take for the construction of the blockchain [4], for instance it models the hashing power in Bitcoin.

Informally, we say that a blockchain protocol is fair if any *correct process* (a process that followed the protocol throughout the whole execution) that has α fraction of the total merit in the system will get at least α fraction of the total reward that is given in the system. Our fairness analysis, in line with Francez's definition of fairness [16], generally defines the fairness of protocols based on voting committees (e.g. Byzcoin [27], Hyperledger Fabric [11], PeerCensus [8], RedBelly [7], SBFT [18], Tendermint [2, 3, 5], etc.), actually studies fairness by separating the fairness of their *selection mechanism* and the fairness of their *reward mechanism.*

The selection mechanism is in charge of selecting the subset of processes that will participate to the agreement on the next block to append in the blockchain, while the reward mechanism defines the way rewards are distributed among processes that participate to the agreement. We propose a formal definition of fairness of selection mechanisms, and then we study the fairness of some selection mechanisms. The analysis of the reward mechanism allowed establishing the following fundamental result and necessary conditions with respect to the fairness of committee-based blockchains as follows:

There exists a(n) (eventual) fair reward mechanism for committee-based blockchains if and only if the system is (eventual) synchronous and faulty processes are detectable. (Theorems 11 and 13).

The rest of the paper is organized as follows. In Section 2, we compare the existing studies of fairness in blockchain systems; in Section 3, we define the system model; in Section 4, we give the basics of committee-based blockchain systems; in Section 5, we study the fairness in committee-based blockchain systems; in Section 6, we analyze some behaviors and the impact of the communication model on rewards; and in Section 7, we conclude.

2 Related Work

The closest work in blockchain systems to our fairness study (however very different in its scope) is the study of the *chain-quality*. In [17], Garay *et al.* define the notion of *chain-quality* as the proportion of blocks mined by honest miners in any given window; Garay *et al.* study the conditions under which during a given window of time, there is a bounded ratio of blocks in the chain that malicious players produced, over the total blocks in the blockchain. Kiayias *et al.* in [25] propose Ourobouros [25] and analyze its chain-quality property. In [33], Pass and Shi propose a notion of *fairness* which is an extension of the chain-quality property, they address one of the vulnerabilities of Bitcoin studied formally in [12, 13]. In [12, 13], Eyal and Sirer prove that if the adversary controls a coalition of miners holding even a minority fraction of the total computational power, this coalition can gain twice its share. Fruitchain [33] overcomes this problem by ensuring that no coalition controlling less than a majority of the computing power can gain more than a factor $1 + 3\delta$ by not respecting the protocol, where δ is a parameter of the protocol. We note that in their model, only one process creates a block in the blockchain, and that process has a reward for the created block. In [20], Guerraoui and Wang study the effect of the delays of message propagation in Bitcoin, and they show that in a system of two miners, one can take advantage of the delays and be rewarded exponentially more than its share. We extend the definition of [33] for systems where each block is produced by a subset of processes. This is the case of Tendermint [5] or Hyperledger Fabric [11] for example, where for each block there is a subset of processes, *a committee* that produces that block.

In [22, 21], Gürcan *et al.* study the fairness from the point of view of the processes that do not participate to the construction of the blockchain. Herlihy and Moir do a similar work in [23] where the authors study users' fairness and consider as an example Tendermint. Herlihy and Moir discussed how processes with malicious behavior could violate fairness by choosing transactions, and then they propose modifications to the original Tendermint to make some violations detectable and accountable. In [29], Lev-Ari *et al.* study fairness on transactions in committee-based blockchains with synchronous assumptions by using a detectable communication abstraction allowing them to identify malicious participants. Our work does not study fairness in the point of view of users.

Recent works consider the distribution of rewards in proof-of-stake based blockchains. In [28], Lagaillardie *et al.* show that even if Tendermint is unfair, the presence of delegators helps in the growth of the system. In [14], Fanti *et al.* define equitability, which represents the evolution of the fraction of total stakes of nodes, in particular, they compute the effect of so-called *compounding* where rewards are directly re-invest in stakes. Karakostas *et al.* define in [24] egalitarianism. Egalitarianism means that each node, no matter its stake fraction, wins the election process to append a new block the same amount as everyone. In this work, we focus on fairness, where nodes with higher merit should get more rewards. Our notion of fairness is different that egalitarianism, since the goal of egalitarianism is to have all nodes rewarded the same, no matter their merit.

3 System Model

The system is composed of an infinite set $\Pi = \{p_1, p_2, \dots p_i, \dots\}$ of sequential processes; i is the *index* of p_i. *Sequential* means that a process executes one step at a time. This does not prevent it from executing several threads with an appropriate multiplexing. As local processing time are negligible with respect to message transfer delays, we consider it as being equal to zero.

Arrival model. We assume a *finite arrival model* [1], *i.e.* the system has infinitely many processes but each run has only finitely many. The size of the set $\Pi_\rho \subset \Pi$ of processes that participate in each system run is not a priori-known. We also consider a finite subset $V \subseteq \Pi_\rho$ of committee members. The set V may change during any system run and its size $|V| = n$ is a priori known. A process is promoted in V by a selection function. Such selection function can be based for instance on stakes in proof-of-stake blockchains, or computing power in proof-of-work blockchains.

Time assumptions on communication. The processes communicate by exchanging messages through an eventually synchronous network [10]. *Eventually Synchronous* means that after a finite unknown time τ there is an a priori unknown bound δ on the message transfer delay. When $\tau = 0$ and δ is known the network is *synchronous.*

Failure model. Some processes can exhibit a Byzantine behavior [34] in the system. A Byzantine process is a process that deviates arbitrarily from the given protocol. We do not assume any bound on their number in the system, but up to f committee members can exhibit a Byzantine behavior at each point of the execution. A process that exhibits a Byzantine behavior is called a Byzantine or a *faulty* process. A process that follows the given protocol is called *correct*.

Communication primitives. In the following, we assume the presence of a broadcast primitive. The primitive broadcast() is a best effort broadcast, which means that when a correct process broadcasts a value, eventually all the correct processes deliver it. A process p_i receives a broadcast of a message by executing the primitive delivery(). Messages are created with a digital signature, and we assume that digital signatures are unforgeable, so when a process p_i delivers a message, it knows the process p_j that created the message.

4 Committee-based Blockchains

Any committee-based blockchain uses instances of consensus solving a form of repeated consensus. This way, each committee agrees on a single value to avoid forks. Anceaume *et al.* [4] proved that classical distributed consensus is required to avoid forks.

Each correct process outputs an infinite sequence of decisions called the *output* of the process. More formally, as described by Delporte-Gallet *et al.* [9], and generalized to Byzantine failures in [2], an algorithm implements a repeated consensus if and only if it satisfies the following properties: (i) *Termination:* Every correct process has an infinite output. (ii) *Agreement:* If the i^{th} value of the output of a correct process is B, and the i^{th} value of the output of another correct process is B', then $B = B'$. (iii) *Validity:* Each value in the output of any correct process is valid with respect to a predefined predicate.

Detailed Description of the Algorithm

We denote by $\mathbb{B}$ the set of all blocks. A block contains, among other things, a header and a list of transactions. Let $bc \in \mathbb{B}^*$, be a finite sequence of blocks. $|bc|$ is the length (the number of blocks) of bc. We say that bc is a *blockchain* if $\forall k \in \mathbb{N} : 0 < k \leq |bc|$, in the header of the block at position k in bc, there is the hash of the block at position $k - 1$. If additionally, the list of transactions in each blocks in the blockchain is valid with respect to the given application, we say that bc is a *valid* blockchain. The block at position 0 is the *genesis block*. Each process has a non-negative *stake*, which is the total amount of token it has. $\forall h > 0$, let V_h be the set of committee members for the height h. $\forall h > 0$, we assume that $|V_h| = n$, the size of committee member is fixed and equal to n.

The genesis block initializes the blockchain, selects the committee that will produce the block at position 1, describes how rewards will be distributed among committee members (which we call the *reward mechanism*), gives the initial distribution of stakes, and describes how processes will be selected for being part of committees with respect to the state of the blockchain (the *selection mechanism*). These information should be public, and known by all processes such that with the history of the blockchain, all processes can always compute deterministically the sets of committee members. This preclude in this work to consider proof-of-work blockchains (as in Bitcoin) since the computing power of the processes is not known by all processes nor verifiable.

For a height, processes not in the corresponding committee just wait for the decision from the committee members. The committee members for that height execute the consensus algorithm to decide on the block for that height. Once a process decides on a block, it sends the decided block to the whole network, and moves to the next height. Non-committee members wait to collect enough times the same decided block from the committee members, in order to be tolerant to failures, and then move to the next height. When moving to the next height, processes wait a certain amount of time to collect more messages from committee members. These messages are the ones used to reward the previous committees. Intuitively, if a process receives a decision message for the decided block by a committee member, then probably that committee member followed the protocol during that height[1]. This allows implementing the repeated consensus. In [2], the authors formalized and proved correct the Tendermint repeated consensus algorithm, an example of committee-based blockchain protocol.

In this paper, we analyze the fairness of committee-based blockchains as described in the following section.

5 Fairness of Committee-based Blockchains

Chain Quality and Committee-based Blockchains

In the blockchain literature, chain quality has been defined by [17], and extended by [33], to study fairness in Bitcoin-like systems. A blockchain system has the property of chain quality if the proportion of blocks produced by honest processes in any given window, is proportional to their relative mining power. Intuitively, chain quality ensures that malicious processes do not produce more blocks than their proportion of mining power. One of the main differences between Bitcoin-like system and the committee-based blockchains is that in the former, one process produces a block, whereas in committee-based blockchain, a committee (a set) of processes produces a block. A committee is not necessarily composed only of correct processes, but can contain Byzantine or correct processes (with a correctness hypothesis of having some majority of the members following the protocol). We cannot apply the definition of chain quality to committee-based blockchain. Instead of defining the fairness with the blocks, we will define it relative to the proportion of total rewards a process gets.

Informally, we say that a blockchain protocol is fair if any *correct process* (a process that followed the protocol) that has a fraction α of the total merit in the system will get at least α fraction of the total reward that is given in the system. In order to tackle the fairness of a committee-based blockchain protocol such as HotStuff [36], Hyperledger Fabric [11], Redbelly [7], SBFT [18] or Tendermint [2, 3, 5], we split the mechanism in two: the *selection mechanism* and the *reward mechanism*. We say that each process has a given merit, which represents the effort the process is putting to maintain the blockchain, for instance it can represent the mining power of a process in proof-of-work blockchains, or the stakes in proof-of-stake blockchains, etc. The *selection mechanism* selects for each new height the committee members (the processes that will run the consensus instance) for that height. The *reward mechanism* is in charge to distribute rewards to committee members that produce a new block. Informally, if the selection mechanism is fair, then each process will become committee member proportionally to its merit, and if the reward mechanism is fair then for each height, only the correct committee members get a reward. By combining the two mechanisms, a correct process is rewarded at least a number proportional to its merit parameter over the infinite execution of the system.

[1] That is not true in general, since Byzantine processes, for instance, can send the decided value at the end without doing anything during the protocol execution.

5.1 Selection Mechanism

5.1.1 Definition and Fairness of Selection Mechanisms

In a system where the size of the committee is strictly lower than the number of processes in the system, there should be a way to select the members of the committees. Always selecting the same processes is a way to centralize the system. That set of processes can exercise a power of oligarchy, and add in the blockchain only transactions they want.

Formally, a selection mechanism is the function $\mathsf{selection} : \mathbb{B}^+ \times \mathbb{N} \to \Pi^n \cup \emptyset$, where n is the size of committees, Π the set of processes, and $\mathbb{B}^+$ represents the set of non-empty blockchains such that if bc is a non-empty blockchain, then:

$$\mathsf{selection}(bc, h) = \begin{cases} V_h, & \text{if } |bc| \geq h - 1 \\ \emptyset, & \text{otherwise} \end{cases},$$

where we recall that $|bc|$ means the length of the blockchain bc, and V_h is the set of committee members for height h.

Some information can be computed and/or stored in the blockchain, for instance the number of time each process has been committee member, or the stakes of each process that represents their wealth. The selection mechanism can select, based on these information, for instance, processes with the highest stakes, or processes that were committee members less often, or always the same set of processes, etc.

To abstract the notion of effort of a process, we denote by $\alpha_i(t) \in [0, 1]$ the merit parameter of p_i at time t proportionally to the total merit at time t, such that $\forall t, \sum_{p_i \in \Pi_\rho} \alpha_i(t) = 1$.

If $\forall t \in \mathbb{N}, \forall i, \alpha_i(t) = \alpha_i(0)$, we denote by α_i the merit of the process p_i. That means that the merits do not depend on the evolution of the blockchain, nor on its contents. Let v_i be the number of times p_i becomes committee member, proportionally to the number of blocks, so $v_i \in [0, 1]$. We propose the following definition of fairness of selection mechanisms where merits are fixed and do not change with time. The definition allows all processes with positive merit to be member of committees infinitely often, and with respect to their merit.

▶ **Definition 1** (Fairness of Selection Mechanism)**.** *Assume that the blockchain is built infinitely, so $\forall h \geq 0$, there is a block at position h. We say that a selection mechanism is fair if it respects the following properties:*

1. *If $\alpha_i \neq 0$ then $v_i \neq 0$; or equivalently, $\alpha_i \neq 0 \implies \forall h \geq 0, \exists h' \geq h : p_i \in V_{h'}$*
2. *If $\alpha_i \geq \alpha_j$ then $v_i \geq v_j$.*

Informally, Condition 1 means that each process with a positive merit parameter should become a committee member infinitely often. Condition 2 means that a process with a low merit cannot be selected more than a process with a higher merit. Note that this definition depends only on the merit and not on the behavior of the processes (correct or Byzantine).

A definition of fairness of a generic selection mechanism (that does change over time) is still an open question, but such definition should encapsulate the Definition 1 as special case.

▶ Remark 2. If the total number of processes in the system is equal to the size of committees, then all processes are always selected, so the selection mechanism in that case is trivially fair, although asking a huge set of processes to run the consensus is not scalable.

5.1.2 Examples of Selection Mechanisms

In this section, we briefly study different possible selection mechanisms. Let us assume that there are $N > n$ processes during the whole execution of the system and processes cannot enter nor exit. Let us also assume that merits do not change over time, $\forall t \in \mathbb{N}, \forall i, \alpha_i(t) = \alpha_i(0)$ and that all processes have the same merit, $\forall i, j \in \Pi_\rho, \alpha_j = \alpha_j > 0$.

In the examples below, we consider selection mechanisms that depends on the stakes of processes, while the merit is fixed and does not depends on the (evolution of) stakes. All processes are correct, and a committee member is rewarded when the committee it is part of produces a block. The reward increases the stake of the process. For our analysis, we further consider that processes are ordered by their stake and their id (public address for instance). Let us assume that at the beginning of the execution, all processes have the same amount of stake. Without loss of generality, and up to renaming, consider also that during the execution, if $\exists i, j : i < j$ such that p_i and p_j have the same amount of stakes, then p_i is selected before p_j.

Select the processes with the highest stake

This selection mechanism works as follows: for any height h, with respect to the blockchain up to height $h - 1$, the n processes having the biggest amount of stakes are selected to be part of the committee.

This mechanism lead to the situation where only the n processes selected first will always be selected, and the other processes will never be. This mechanism is not fair, since Condition 1 is not satisfied. The processes not selected have, by assumption, a positive stake, and so they should be selected infinitely often.

Note that in the special case where there are at most n processes with a positive stake, and they are all selected, then fairness of selection is satisfied.

Select the processes with lowest stake

This selection mechanism works as follows: for any height h, with respect to the blockchain up to height $h - 1$, the n processes having the lowest amount of stakes are selected to be part of the committee.

If all N processes are part of the system since the beginning, the number of times each process has been selected after l blocks is on average $l * n/N$ selections. This mechanism is fair according to the Definition 1.

Let us remark that this mechanism is fair with the model considered in this example, since processes cannot exit nor enter during the execution. If that assumption is removed, the following can happen. If processes could enter or leave, once a process is selected and rewarded, it knows that it would not be selected before a long period of time, on average after n/N blocks, one incentive could be to create new sub addresses (processes) such that they will have low stakes and it will be selected more often. Another similar scenario is that if a process has a big amount of stakes, it might not be allowed to participate in committees for a long time until all the process with small stakes caught up. The process might want to split into lot of stakeholders with small stakes. In such a way, the new stakeholders will always have the smallest stakes and be selected, if it continues to do so, it might block other processes to be committee members. Therefore, selecting the processes with the lowest stakes is not stable in an open setting.

▶ Remark 3. An unfair selection mechanism can lead to a centralization of the system, by always letting the same processes decide on the next block. Although the assumption on the bound of Byzantine processes does not depend on the selection mechanism, we note that when a selection mechanism selected the processes with the lowest amount of stakes, and only correct processes in committees are rewarded, at some point processes that were non-correct will have the least stake, and thus be selected.

5.2 Reward Mechanism

5.2.1 Definition of Reward Mechanism

In blockchain systems, processes that produce and add blocks to the blockchain are rewarded. In a committee-based blockchain, a committee is the producer of the block. Within that committee, some processes may not behave as prescribed. There may be different ways of rewarding members of committees. To do so, we define the *reward mechanism.* A reward mechanism consists of the reward function defined as follows: $\mathsf{reward} : \mathbb{B}^+ \times \mathcal{P}(\mathit{Information}) \times \mathbb{N} \to \mathbb{R}^{|\Pi|} \cup (\bot, \dots, \bot)$, where $\mathcal{P}(\mathit{Information})$ is the power set of all messages, and we recall that Π is the set of processes, and $\mathbb{B}^+$ represents the set of non-empty blockchains.

If bc is a blockchain and $|bc| < h$, $\mathsf{reward}(bc, M, h) = (\bot, \dots, \bot)$. Otherwise, it assigns to each process a given reward. In $\mathsf{reward}(bc, M, h)$, M represents the set of messages received, h the height of the blockchain bc where the reward of committee member is computed.

A reward is considered allocated if it is written in the blockchain. The second part of the reward mechanism is to choose when to allocate the rewards corresponding for a given height. If a reward has been allocated at a height h, the process can use it after a certain number of blocks defined in the genesis block (e.g. [17, 13]). We consider that for each production of block its rewards are allocated in exactly one block and not over different blocks, such that after the allocation of rewards a process knows if it has been rewarded or not. Note that once rewards are allocated, they cannot change anymore. Some blockchain systems add punishment mechanism, called *slashing*, to afflict, afterwards, some costs to a process if there is a proof of some misbehavior, as described in [32].

5.2.2 Fairness of Reward Mechanism

Let p_i be a process, and let T be a fragment of p_i's execution. If at the beginning of the fragment T, the internal state of p_i is correct and p_i follows the given protocol during the fragment T, then we say that p_i is T-correct. A correct internal state is a state of the process that can be the result of p_i following the protocol. A correct process is T-correct $\forall T$. For example, p_i is considered h-correct if during its execution of height h it followed the protocol.

We define the following properties for characterizing the fairness of a reward mechanism. Let h be a height. Each committee member has a boolean variable r_i^h, which we call *reward parameter* defined as follows:

1. If p_i is a not a committee members for h, then $r_i^h = 0$,
2. **h-completeness.** If p_i is a committee members for h and p_i is h-correct, then $r_i^h = 1$,
3. **h-accuracy.** If p_i is a committee members for h and p_i is not h-correct, then $r_i^h = 0$.

If $r_i^h = 0$, it means that p_i is not rewarded for height h, and if $r_i^h = 1$, p_i has been rewarded for h. The properties are inspired by classical properties of failure detectors [6].

▶ Remark 4. Note that we do not reward non-committee members. In this article, we do not consider delegations. When a process delegates to a committee member, once the committee member is rewarded, all of its delegates are rewarded proportionally to what they delegated. In future works, we may consider the case of committee-based blockchains with delegation. To do so, r_i^h must contain more information and not just be a boolean variable.

▶ **Definition 5** (Complete Fairness of a Reward Mechanism). *Let $\mathcal{R}$ be a reward mechanism. If $\forall h > 0$, $\mathcal{R}$ satisfies Conditions 1 and h-completeness (Condition 2), we say that $\mathcal{R}$ satisfies complete fairness.*

If a reward mechanism satisfies complete fairness, it means that for all height $h > 0$, all h-correct committee members are rewarded, and non-committee members are not.

▶ **Proposition 6.** *There exists at least one reward mechanism satisfying complete fairness.*

Once a block is in the chain, rewarding all committee members, in the next block, for that block and only them, satisfy Conditions 1 and 2. Condition 1 is satisfied since for all height, non-committee members are not rewarded. Condition 2 also holds, for any given height h, all committee members of h are rewarded, in particular all h-correct committee members.

▶ **Definition 7** (Accurate Fairness of a Reward Mechanism)**.** *Let $\mathcal{R}$ be a reward mechanism. If $\forall h > 0$, $\mathcal{R}$ satisfies conditions 1 and h-accuracy (Condition 3), we say that $\mathcal{R}$ satisfies accurate fairness.*

If a reward mechanism satisfies accurate fairness, it means that for all height $h > 0$, all non h-correct committee members are not rewarded.

▶ **Proposition 8.** *There exists at least one reward mechanism satisfying accurate fairness.*

Never allocating rewards satisfies Conditions 1 and 3. Condition 1 is satisfied since non-committee members are not rewarded. Condition 3 holds, since no process is rewarded. In particular for any given height $h > 0$, no non h-correct committee members is rewarded.

Although simple and trivial to satisfy either complete fairness or accurate fairness, satisfying both at the same time is more complex and not always possible.

▶ **Definition 9** (Fairness of a Reward Mechanism)**.** *Let $\mathcal{R}$ be a reward mechanism. If $\forall h > 0$, $\mathcal{R}$ satisfies Conditions 1, h-completeness (Condition 2) and h-accuracy (Condition 3), we say that $\mathcal{R}$ is fair.*

We say that a reward mechanism is fair when at each height h, all and only h-correct committee members are rewarded.

▶ Remark 10. $\forall h > 0$, if $|V_h| > 1$, then for a reward mechanism to be (eventually) fair, rewards cannot be allocated directly in the block. For any height $h > 0$, the set of h-correct committee members cannot be known in advance. If $\forall h > 0, |V_h| = 1$, the reward can be directly given to the only committee member, so in the block at height h.

▶ **Theorem 11.** *There exists a fair reward mechanism in a committee-based blockchain protocol iff the system is synchronous.*

Proof. We prove this theorem by double implication.

- If there exists a fair reward mechanism, then the system is synchronous.
  Let $\mathcal{R}$ be a reward mechanism. By contradiction, we assume that $\mathcal{R}$ is fair and that the system is not synchronous.
  V_h is the set of committee members for the height h. Let $k > 0$ be the number of blocks to wait before distributing the rewards for V_h. The reward is allocated by the committee $V_{h'}$, where $h' = h + k$. Since the system is not synchronous, the committee members of height h', $V_{h'}$, may not receive all messages from V_h before allocating the rewards.
  By Conditions 1, and 2, since the reward mechanism is fair, by Conditions 1 - 3, all and only the h-correct committee members of the height h have a reward parameter equal to 1. That means that the h'-correct committee members of $V_{h'}$ know exactly who were the h-correct committee members in V_h, so they got all the messages before giving the reward. Contradiction, so the system is synchronous.

- If the system is synchronous, then there exists a fair reward mechanism.
  We assume that the system is synchronous and $\forall h > 1$, all messages sent by h-correct processes at height h are delivered before the block at height $h+1$. Let $\mathcal{R}$ be the following reward mechanism: let h be a height. Rewards for a block at height h are allocated at height $h+1$ by the committee V_{h+1}.
  - If a process is not a committee member for height h, set its reward parameter to 0, this is known since processes are already at height $h+1$.
  - By combining the messages from committee member of h processes, since the communication system is synchronous, it is possible to differentiate between h-correct and non h-correct committee members, and so set the reward parameter of h-correct committee members of h to 1; and set the reward parameter of non h-correct committee members of h to 0.

  By construction, the committee members in $h+1$ allocates rewards to all and only h-correct committee members, so $\mathcal{R}$ is fair, it satisfies all fairness Conditions 1 - 3.

$\square_{Theorem\ 11}$

If there is no synchrony, there cannot be a fair reward mechanism for committee-based blockchains. Our definition of fairness states that for any height Conditions 1-3 are satisfied. Processes should always receive all rewards they deserve. This definition can be weakened.

▶ **Definition 12** (Eventual Fairness of a Reward Mechanism). *Let $\mathcal{R}$ be a reward mechanism. If $\exists h_0 > 0 : \forall h \geq h_0$, $\mathcal{R}$ satisfies Conditions 1, h-completeness (Condition 2) and h-accuracy (Condition 3), we say that $\mathcal{R}$ is eventually fair.*

A reward mechanism is eventually fair if after an unknown time, the rewards are allocated to and only to correct committee members.

We note that if a reward mechanism is fair, then it is eventually fair but the reverse (reciprocal) is not necessarily true.

Detectable Byzantine Processes

In synchronous systems, it is always possible to detect Byzantine processes, for example using the broadcast abstraction detectable all-to-all (DA2A) defined in [29]. Detecting Byzantine processes allows to not reward them, and then to satisfy Condition 3. On the other hand, in eventual synchronous systems, the problem is more difficult. As in this work, Kihlstrom et al., in [26] distinguish between detectable and non-detectable Byzantine. The detectable Byzantine are the processes whose behavior can be detected, for instance by doing omission or commissions failures. Non-detectable Byzantine are Byzantine processes whose fault cannot be detected, for example processes that alter their internal state. In [19], Greve et al. extend that approach and propose a failure detector for detectable Byzantine in dynamic networks.

Note that although Kihlstrom *et al.* proposed in [26] a failure detector for solving consensus, our problem is not the same, and we cannot apply their failure detector as it is. In [26], once a process has a detectable Byzantine behavior, it should be suspected forever. In our model, we do not want to punish indefinitely Byzantine processes. In fact, we want for any height h to not reward only processes that were not h-correct. For example, let p_i be a process such that it is part of committees h and h', such that $h' > h$. Suppose also that during height h, p_i sends contradictory messages (and so is Byzantine), but then recovered before the beginning of height h' and follows the protocol during all h'. Even if p_i is not h-correct, it recovered before h' and is h'-correct. If p_i has been detected and not rewarded for height h, that should be taken into consideration of its work during height h', and since it follows the protocol, it should be rewarded for height h'. The failure detector proposed by Kihlstrom *et al.*, once a process has been suspected, marks such process as Byzantine forever.

▶ **Theorem 13.** *There exists an eventual fair reward mechanism in a committee-based blockchain protocol iff the system is (eventually) synchronous and Byzantine processes are detectable.*

Proof. We prove this theorem by double implication.

- If there exists an eventual fair reward mechanism, then the system is eventually synchronous or synchronous and Byzantine processes are detectable.
  Let $\mathcal{R}$ be a reward mechanism. We assume that $\mathcal{R}$ is eventually fair.
  If $\mathcal{R}$ is fair, by Theorem 11, the communication is synchronous, which ends the proof. Otherwise, since $\mathcal{R}$ is eventually fair, that means that there is a point in time h from which all the rewards are correctly allocated, so for any height $h' \geq h$, h'-correct committee members of committees at height h' are able to distinguish between non-correct processes during the height they are distributing the rewards, the Byzantine are then detectable. If we consider h as the beginning of the execution, then we have that $\mathcal{R}$ is fair, and by Theorem 11, the message delay is upper bounded. We have that after h, the message delay is upper bounded, so the communication is eventually synchronous. Therefore, the Byzantine are then detectable, and the communication is synchronous or eventually synchronous.
- If the system is eventually synchronous or synchronous, and Byzantine processes are detectable, then there exists an eventual fair reward mechanism.
  If the system is synchronous, the proof follows directly from Theorem 11. Consider that the system is eventually synchronous, but not synchronous. Let $\mathcal{R}$ be the following mechanism: Let h be a height. Rewards for a block at height h are allocated at height $h+1$ by the committee V_{h+1}.
  - If a process is not a committee member for height h, set its reward parameter to 0, this is known since processes are already at height $h+1$.
  - By combining the messages from committee member of h processes, if there is not sufficient information to detect the behavior of processes, reward only those detected as h-correct and in V_h, and the process proposing the distribution of reward increases the duration to wait before starting the next height. If there is enough information to detect the behavior of all processes in V_h, then reward the h-correct processes in V_h and do not reward non h-correct processes in V_h.

  $\mathcal{R}$ is eventually fair.

$\square_{Theorem\ 13}$

▶ **Corollary 14.** *In an asynchronous system, there is no (eventual) fair reward mechanism in a committee-based blockchain tolerating Byzantine processes.*

Proof. Assume that the system is asynchronous, where there are good periods such that consensus can be reached. By contradiction, let $\mathcal{R}$ be an eventual fair reward mechanism.

- If there are non-detectable Byzantine processes in the system, $\mathcal{R}$ is not fair (Theorem 13);
- If all Byzantine processes are detectable, then by Theorem 13, the system must be synchronous, or eventually synchronous.

We have a contradiction, since the system is asynchronous. It is not possible to have an (eventual) fair reward mechanism in an asynchronous system. $\square_{Corollary\ 14}$

Note that this result holds even if all the processes are correct but not known in advance, and the protocol tolerates Byzantine faults. In fact, it is different from the FLP impossibility result of consensus in an asynchronous system with one faulty process [15].

6 Numerical Examples

In this section, we examine the impact of different communication models on the fairness of reward mechanisms through several numerical examples that confirm the results on the fairness of reward mechanisms from Section 5.2.

Execution. In our analyses, processes run a committee-based blockchain protocol as described in Section 4, and rewards for a block produced at a height h are allocated in the block at height $h+1$. Note that the consensus module is Byzantine fault tolerant. We highlight the environment's important characteristics: the communication system, the total number of processes in the system, the size of each committee, the different type of processes and their number at a given height, the rewarding mechanism, and the selection mechanism. We must choose the value of these parameters before launching the execution. We consider different communication systems, and rewards are allocated by the next committee by using messages they delivered from the previous height – use the combination of all messages and check if they correspond to the correct time and a possible value to send according to the state.

We consider a system where all processes are part of all consensus instances. For clarity, and without loss of generality, we consider a system with $n \geq 4$ processes where all are selected. As stated in Remark 2, selecting all processes is a fair selection mechanism. We can now focus on the impact of the network on rewards. For any height h, there can be at most $\lfloor (n-1)/3 \rfloor$ non h-correct processes in each committee. For a committee, a quorum of $\lceil 2n/3 \rceil$ is needed for any decisions. In the case where there are for any height h some non h-correct processes, we assume that processes have enough information to detect it when distributing rewards. In particular, and for the experiment the Byzantine are specially tagged, and that tag is used only for allocating rewards. When an h-correct process receives a message from a non h-correct, it suspects it, and broadcasts the information. When an h-correct process delivers at least $2\lfloor n/3 \rfloor + 1$ suspicions for a process, it considers it as non h-correct, and does not propose to reward it.

We use MATLAB [30] for our analysis. We analyze three different communication models. First, a synchronous communication, where there is no delay. Then we consider the two following semi-synchronous communication models (i) the system alternates between good and bad periods, where during good periods message delays are upper bounded, and (ii) from an unknown time, message delays are upper bounded (eventually synchronous model). We note that in all these models, consensus can be reached. In the good/bad model, progress for consensus instances are guaranteed during the good periods. Note that in the eventually synchronous model, once the global stabilization time (GST) happens all message delays are upper bounded. If for a process, the GST happens during height h, then for all height $h' > h$, the message delays are upper bounded.

In each configuration of the communication model, we ran the experiment 50 times, and took the mean. 0 represents if a process did not receive a reward, and 1 if the process received a reward for the corresponding height. Due to the space limitation, we only present the experiment of the eventual synchronous model.

Eventually Synchronous Model

We consider an eventual synchronous model where all processes are correct, and part of each committee instances. Recall that eventual synchronous is a system where after a finite but unknown time (the GST), message delay is upper bounded for the rest of the execution. In our examples, we consider that the GST happens during height 10.

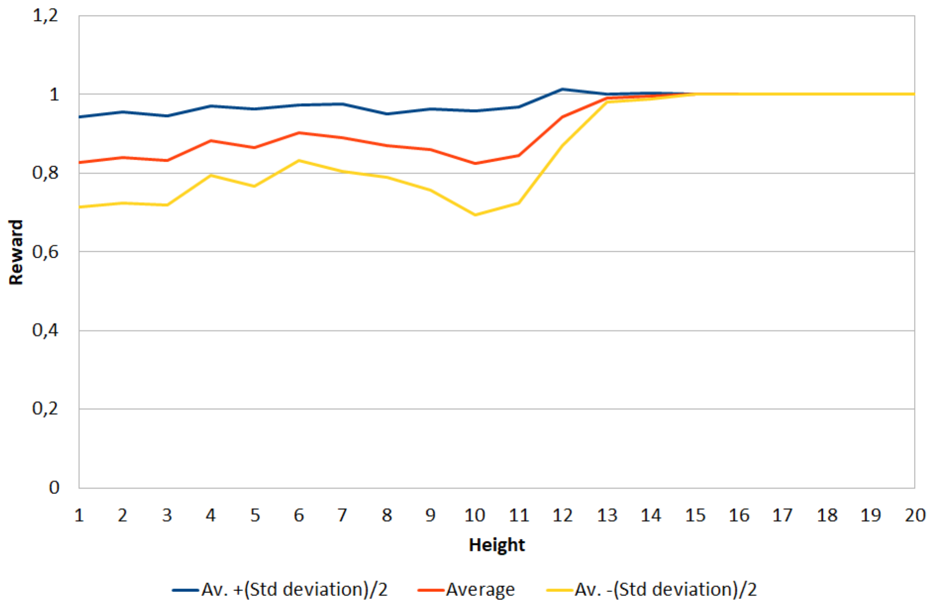


Figure 1 Evolution of Rewards in an Eventual Synchronous System, where Global Stabilization Time happens during height 10.

On Fig. 1, we present the evolution of reward for each height. We draw the mean of the average reward of each participant (red curve). The blue and yellow curves represent the standard deviation. We can see the set in which the processes are rewarded. When the blue and yellow curves converge, it means that all processes have on average the same reward. We notice that from height 10, the evolution of the reward is increasing. Approximately, from height 14, all processes are rewarded. Before height 10, there is a fluctuation in the evolution of rewards allocated because of the asynchronous periods; processes are not necessarily rewarded even if they participate. Their messages were not received on time. Once the GST happens, message delays become upper bounded but some processes still have a time-out shorter than the bound. These processes still increase their time-out until they receive all messages or detect incorrect behavior. When all processes deliver messages during their corresponding rounds, they allocate the rewards to all and only correct processes. It means that the reward mechanism is eventually fair.

7 Conclusions and Future Works

The originality of our contribution is the study of the impact of network conditions on the fairness of the rewarding in committee-based blockchains prone to Byzantine behavior. We proved that the reward mechanism is (eventually) fair iff the system communication is (eventually) synchronous and Byzantine processes are detectable. Our study opens interesting future research directions in particular the extension to other types of behaviors such as rational or amnesic. Furthermore, we are interested in studying the impact of network attacks on the fairness of rewarding. Another interesting direction is the design of self-adaptive fair rewarding schemes.

References

1 Marcos K Aguilera. A pleasant stroll through the land of infinitely many creatures. *ACM Sigact News*, 35(2):36–59, 2004.

2 Yackolley Amoussou-Guenou, Antonella Del Pozzo, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. Correctness of Tendermint-Core Blockchains. In *OPODIS 2018, December 17-19, 2018, Hong Kong, China*, pages 16:1–16:16, 2018.

3 Yackolley Amoussou-Guenou, Antonella Del Pozzo, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. Dissecting tendermint. In *Networked Systems - 7th International Conference, NETYS 2019, Marrakech, Morocco, June 19-21, 2019*, pages 166–182, 2019.

4 Emmanuelle Anceaume, Antonella Del Pozzo, Romaric Ludinard, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. Blockchain Abstract Data Type. In *The 31st ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2019, Phoenix, AZ, USA, June 22-24.*, pages 349–358, 2019.

5 E. Buchman, J. Kwon, and Z. Milosevic. The latest gossip on BFT consensus. *CoRR*, abs/1807.04938v1, July 2018. URL: `https://arxiv.org/abs/1807.04938v1`.

6 Tushar Deepak Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. *J. ACM*, 43(2):225–267, 1996.

7 Tyler Crain, Vincent Gramoli, Mikel Larrea, and Michel Raynal. (Leader / Randomization / Signature)-free Byzantine Consensus for Consortium Blockchains, 2017.

8 Christian Decker, Jochen Seidel, and Roger Wattenhofer. Bitcoin meets strong consistency. In *Proceedings of the 17th International Conference on Distributed Computing and Networking, Singapore, January 4-7, 2016*, pages 13:1–13:10, 2016.

9 Carole Delporte-Gallet, Stéphane Devismes, Hugues Fauconnier, Franck Petit, and Sam Toueg. With finite memory consensus is easier than reliable broadcast. In *Principles of Distributed Systems, 12th International Conference, OPODIS 2008, Luxor, Egypt, December 15-18, 2008. Proceedings*, pages 41–57, 2008.

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

11 Elli Androulaki *et al.* Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. In *Proceedings of the Thirteenth EuroSys Conference, EuroSys 2018, Porto, Portugal, April 23-26, 2018*, pages 30:1–30:15, 2018.

12 Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security - 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014*, pages 436–454, 2014.

13 Ittay Eyal and Emin Gün Sirer. Majority is not enough: bitcoin mining is vulnerable. *Commun. ACM*, 61(7):95–102, 2018.

14 Giulia C. Fanti, Leonid Kogan, Sewoong Oh, Kathleen Ruan, Pramod Viswanath, and Gerui Wang. Compounding of wealth in proof-of-stake cryptocurrencies. In Ian Goldberg and Tyler Moore, editors, *Financial Cryptography and Data Security - 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18-22, 2019*, pages 42–61, 2019.

15 M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2), April 1985.

16 Nissim Francez. *Fairness.* Texts and Monographs in Computer Science. Springer, 1986.

17 J. A. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Proc. of the EUROCRYPT International Conference*, 2015.

18 Guy Golan-Gueta, Ittai Abraham, Shelly Grossman, Dahlia Malkhi, Benny Pinkas, Michael K. Reiter, Dragos-Adrian Seredinschi, Orr Tamir, and Alin Tomescu. SBFT: A scalable and decentralized trust infrastructure. In *49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2019, Portland, OR, USA, June 24-27, 2019*, pages 568–580, 2019.

19 Fabíola Greve, Murilo Santos de Lima, Luciana Arantes, and Pierre Sens. A time-free byzantine failure detector for dynamic networks. In *2012 Ninth European Dependable Computing Conference, Sibiu, Romania, May 8-11, 2012*, pages 191–202, 2012.

20 Rachid Guerraoui and Jingjing Wang. On the unfairness of blockchain. In *Networked Systems - 6th International Conference, NETYS 2018, Essaouira, Morocco, May 9-11, 2018*, pages 36–50, 2018.

21 Önder Gürcan, Alejandro Ranchal Pedrosa, and Sara Tucci-Piergiovanni. On cancellation of transactions in bitcoin-like blockchains. In *On the Move to Meaningful Internet Systems. OTM 2018 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2018, Valletta, Malta, October 22-26, 2018*, pages 516–533, 2018.

22 Önder Gürcan, Antonella Del Pozzo, and Sara Tucci-Piergiovanni. On the bitcoin limitations to deliver fairness to users. In *On the Move to Meaningful Internet Systems. OTM 2017 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2017, Rhodes, Greece, October 23-27, 2017*, pages 589–606, 2017.

23 Maurice Herlihy and Mark Moir. Enhancing accountability and trust in distributed ledgers. *CoRR*, abs/1606.07490, 2016.

24 Dimitris Karakostas, Aggelos Kiayias, Christos Nasikas, and Dionysis Zindros. Cryptocurrency Egalitarianism: A Quantitative Approach. In *International Conference on Blockchain Economics, Security and Protocols (Tokenomics 2019), Paris, France, May 06-07*, 2019.

25 Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, pages 357–388, 2017.

26 Kim Potter Kihlstrom, Louise E. Moser, and P. M. Melliar-Smith. Byzantine fault detectors for solving consensus. *Comput. J.*, 46(1):16–35, 2003.

27 E. Kokoris-Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford. Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing. In *Proceedings of the 25th USENIX Security Symposium*, 2016.

28 Nicolas Lagaillardie, Mohamed Aimen Djari, and Önder Gürcan. A Computational Study on Fairness of the Tendermint Blockchain Protocol. *Information*, 10(12):378, 2019.

29 Kfir Lev-Ari, Alexander Spiegelman, Idit Keidar, and Dahlia Malkhi. Fairledger: A fair blockchain protocol for financial institutions. In *23rd International Conference on Principles of Distributed Systems, OPODIS 2019, December 17-19, 2019, Neuchâtel, Switzerland*, pages 4:1–4:17, 2019.

30 MATLAB. *version 9.6 (R2019a)*. The MathWorks Inc., Natick, Massachusetts, 2019.

31 S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. `https://bitcoin.org/bitcoin.pdf` (visited on 2019-08-15), 2008.

32 Dev Ojha and Christopher Goes. F1 Fee Distribution. In *International Conference on Blockchain Economics, Security and Protocols (Tokenomics 2019), Paris, France, May 06-07*, 2019.

33 Rafael Pass and Elaine Shi. The sleepy model of consensus. In *ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, pages 380–409, 2017.

34 M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, April 1980.

35 Fahad Saleh. Blockchain Without Waste: Proof-of-Stake. SSRN Scholarly Paper ID 3183935, Social Science Research Network, Rochester, NY, January 2019.

36 Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan-Gueta, and Ittai Abraham. Hotstuff: BFT consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019.*, pages 347–356, 2019.

Decentralization in Open Quorum Systems: Limitative Results for Ripple and Stellar

Andrea Bracciali
Department of Computer Science, University of Stirling, UK
http://www.cs.stir.ac.uk/~abb/
abracciali@gmail.com

Davide Grossi
Bernoulli Institute for Maths, CS and AI, University of Groningen, The Netherlands
Amsterdam Center for Law and Economics, University of Amsterdam, The Netherlands
Institute for Logic, Language and Computation, University of Amsterdam, The Netherlands
http://www.davidegrossi.me
d.grossi@rug.nl

Ronald de Haan
Institute for Logic, Language and Computation, University of Amsterdam, The Netherlands
https://staff.science.uva.nl/r.dehaan/
me@ronalddehaan.eu

Abstract

Decentralisation is one of the promises introduced by blockchain technologies: fair and secure interaction amongst peers with no dominant positions, single points of failure or censorship. Decentralisation, however, appears difficult to be formally defined, possibly a continuum property of systems that can be more or less decentralised, or can tend to decentralisation in their lifetime. In this paper we focus on decentralisation in quorum-based approaches to open (permissionless) consensus as illustrated in influential protocols such as the Ripple and Stellar protocols. Drawing from game theory and computational complexity, we establish limiting results concerning the decentralisation vs. safety trade-off in Ripple and Stellar, and we propose a novel methodology to formalise and quantitatively analyse decentralisation in this type of blockchains.

2012 ACM Subject Classification Security and privacy → Distributed systems security

Keywords and phrases Blockchain, decentralization, game theory, computational complexity

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2020.5

Related Version Earlier versions of the paper are available at https://export.arxiv.org/abs/1911.08182.

Acknowledgements We are indebted to the anonymous reviewers of Tokenomics 2020 for several punctual and helpful suggestions on an earlier version of the paper.

1 Introduction

To allow "any two willing parties to transact directly with each other without the need for a trusted third party" [29] was one of the main motivations for the introduction of the Bitcoin blockchain, and several earlier attempts at digital currencies. A blockchain is a distributed state machine in charge of guaranteeing the correctness and trustability of data,[1] e.g., monetary transactions in the case of Bitcoin. State updates are recorded in a chain of data blocks. Data are protected by replication of the state, i.e., of the chain of

[1] Blockchains can also make the computation trustable, e.g., guaranteeing the fair and untamperable execution of agreements among peers encoded as programs.

blocks, within a network of peers. The blockchain protocol must guarantee some form of distributed consensus allowing peers to agree on the information contained in the blockchain, e.g., who has been paid, and that no double spending of virtual coins has occurred, without the supervision of a centralised authority – a currency without a central bank.

Context: the quest for decentralisation

In this paper we address decentralisation in distributed consensus. Decentralisation is a key concern in *permissionsless* blockchains, where participation is allowed in a generally unrestricted way. Permissionless blockchains are clearly exposed to the presence of Byzantine peers, i.e., dishonest peers trying to exploit the network and not bound to the blockchain protocol. Byzantine distributed consensus is a long-standing problem, from Lamport's characterisation [25] and the FLP impossibility result [12], to the subsequent research on data replication and consistency based on Byzantine Fault-tolerant consensus (BFT), [27, 36]. Several proposals are currently competing in a multi-billion market, addressing the so-called *blockchain trilemma*, i.e., achieving *security*, *scalability* and *decentralisation* together.

One of the breakthroughs of Bitcoin was the introduction of the Proof-of-Work (PoW) [9] as a mechanism to enable a probabilistic Byzantine distributed consensus. Informally speaking, by solving a computationally hard problem one of the peers is entitled to create the next block, cryptographically linked to the previous ones. Under the assumption that Byzantine computational power is suitably limited within the network, the probability that enough work can be channeled to alter block history decreases with the ageing of the blocks [15]. Bitcoin reaches finality with an acceptable probability in about one hour (6 blocks), with limited transactions per second.[2]

In *Proof-Of-Stake* (PoS) blockchains peers contribute to the definition of the next block with a probability proportional to the stake (coins), rather than computational power, they detain in the system. Safety follows from the honest peers detaining the majority of stake. Scalability improves in Proof-of-Stake, but the management of security typically results in being more complex.

The *BFT paradigm* has also been proposed for blockchain consensus, providing scalability in transaction per second thanks to low transaction latency and high throughput. BFT, however, is more constrained in terms of the scalability in the number of peers [40], since the number and identity of peers needs to be known and in some cases fixed [5]. This kind of blockchain has been proposed, for instance, for financial services, where a limited number of known and certified peers need to exchange fast and numerous transactions. It is worth remarking that if consensus requires control on peers, a centralised authority might be required, with implications also on identity, privacy and censorship.

Research question

In this paper we focus on BFT blockchains based on quorum systems [39]. In such systems consensus emerges from neighbourhoods of peers. The properties of such neighborhoods, together with assumptions on Byzantine failure thresholds, are then essential to guarantee the liveness and safety of the consensus protocol, that is, whether honest peers are able to eventually reach consensus on a correct next state. At the heart of these protocols is a notion of trust between peers: nodes select which other nodes to trust, and listen to, in the network. The paper focuses on the following research question:

To what extent can consensus be decentralized, when based on peer-to-peer trust relations?

[2] More technical and comprehensive introductions to blockchains can be found in [2, 30, 41].

We take an analytical approach and establish theorems that point to the existence of inherent tradeoffs between safety and decentralization for approaches based on such trust networks. We use tools from game theory (specifically the theory of command games [20, 19]) and computational complexity theory. The interface of methods from theoretical economics and computational complexity have proven extremely prolific in other areas of computer science and artificial intelligence, such as computational social choice theory [3, 18]. Our paper showcases these methods for general investigations on blockchain consensus and decentralisation.

We will specifically consider Ripple [6, 37] and Stellar [28], two quorum-based blockchains attempting to extend the applicability of the BFT paradigm from a permissioned to a permissionless setting, aiming at improving decentralisation. Ripple provides frictionless global payments and corporate-oriented efficient transactions. It currently relies on a list of "authorised" validators[3] in charge of the correctness of transactions. Access is permissioned and each peer will need to have in their neighbourhood of trust a number of validators from the list. While the list was originally entirely composed by Ripple validators, today third-party validators, e.g., private companies and universities, have been included. Stellar provides payments and asset management to corporate and individuals, and aims to push decentralisation further by offering open membership and allowing peers to autonomously define their trust networks, i.e., the set of validators that they trust. However, strong constraints hold on the topology of such trust networks.

Both Ripple and Stellar have been object of criticism with respect to the level of decentralisation of their current implementations, and the need for further research on protocols like Ripple and Stellar is emphasised, for instance, in [4].

Related work

Even though at the time of writing Ripple and Stellar are, respectively, the third and twelfth blockchain systems in terms of market capitalization,[4] little foundational work exists on their protocols. Correctness analyses of Ripple have been proposed in [6], and of Stellar in [28, 17]. A specific study on the issue of decentralisation in Stellar has also very recently been presented in [23]. The paper investigates the network of Stellar's peer-to-peer trust relations by means of an extended version of PageRank used to evaluate nodes' influence. Findings about the current status show centralisation on two critical validators, which are controlled by the Stellar Foundation. Our paper contributes further general results on the level of decentralization that could reasonably be achieved in systems like Ripple and Stellar.

Paper contribution and outline

The paper makes three contributions. *First* it develops a novel theoretical framework, rooted in economic theory (command games [20, 19], power indices [33, 1]), to ascertain the influence that peers can exert on each other in quorum systems based on trust networks. This contributes a novel methodology for a much needed quantitative evaluation of decentralisation in blockchain with respect to its consensus layer. The proposed methods are applied to Ripple and Stellar (Theorem 25). *Second*, it establishes an "impossibility of decentralisation" result for a class of consensus protocols of the Ripple type (Theorem 16), which are based on

[3] At the time of writing the list consists of about 30 validators, available at `https://xrpcharts.ripple.com`

[4] Source: `https://coinmarketcap.com/all/views/all/`. Retrieved on 11th May 2020.

trust networks with a fixed threshold of tolerable Byzantine peers. This results shows that, in such systems, a necessary condition for safety is the existence of validators that must be trusted by every peer in the network, hindering the possibility of full decentralisation. *Third*, it develops an appraisal of computational barriers to decentralization in protocols like Stellar, that are based on so-called federated Byzantine agreement systems. Specifically, we show that constraints that are necessary to guarantee the safety of the network require peers to be able to solve problems that are computationally intractable in principle (Theorems 21 and 22). This result identifies computational difficulties for the construction of safe peer-to-peer trust networks, thereby pointing to computational barriers for the full decentralisation of Stellar.

A Byzantine model of trust network is introduced in Section 2, impossibility and intractability results are presented in Section 3, and decentralisation measures in Section 4. Section 5 concludes. All proofs are provided in a technical appendix.

2 Preliminaries

The paper is concerned with systems based on the following high-level blueprint: nodes hold opinions on a value (e.g, whether a transaction should be recorded or not); they validate an opinion when they observe enough nodes, among those they trust, that hold the same opinion (agreement); some nodes may be Byzantine; and no two honest nodes should be able to validate opinions with different values (safety). Crucially, nodes are able to autonomously decide which other nodes to trust. This section defines the above set-up formally.

2.1 Opinions and Opinion Profiles

Let N be the set of nodes, with $n = |N|$, and let $H \subseteq N$ the set of honest nodes and $B = N \setminus H$ the set of Byzantine nodes. The opinion of a honest node $i \in N$ is a value in $\{0, 1\}$. At any given time, the collection of each node's opinions defines an opinion profile that associates a "genuine" opinion from $\{0, 1\}$ to every honest node (the opinion that the node reveals to the network). And to each Byzantine node in $N \setminus H$ it associates a function from honest nodes to opinions. This function represents the value that each Byzantine node would reveal to each honest node.

▶ **Definition 1.** *An* opinion profile $\mathbf{o} : N \to \{0,1\} \cup \{0,1\}^H$ *such that* $\mathbf{o}(i) \in \{0,1\}$ *if* $i \in H$ *and* $\mathbf{o}(i) \in \{0,1\}^H$ *if* $i \in B$*. Given* $x \in \{0,1\}$*,* $\overline{x}$ *represents the element of singleton* $\{0,1\} \setminus \{x\}$*.*

The definition makes some simplifications, which are worth flagging. Even from the perspective of binary valued opinions, it would be more realistic to work with a ternary set of opinions containing 1, 0 plus an undefined value representing undecided nodes. Also, the definition rules out the possibility for a Byzantine node to reveal inconsistent values (that is, 1 *and* 0 to a same node). However, such simplifications are not fundamental, and the results we present would carry over to these more general settings.

We define then what it means for a node to observe an agreement among other nodes, in a given opinion profile.

▶ **Definition 2.** *Let an opinion profile* $\mathbf{o}$ *be given. A set of nodes* $C \subseteq N$ *is said to agree (in* $\mathbf{o}$*) from the perspective of* $i \in N$ *if for all* $j, k \in H \cap C$ *and* $j', k' \in B \cap C$*,* $\mathbf{o}(j) = \mathbf{o}(k) = \mathbf{o}(j')(i) = \mathbf{o}(k')(i)$*.*

That is, a node i observes agreement in a set of nodes C whenever the honest nodes in that set hold the same opinion, and that opinion is also the opinion revealed to i by the Byzantine nodes in C.

2.2 Byzantine Trust Networks

We now shift to the definition of the structures of peer-to-peer trust relations underpinning consensus in the systems we are focusing on.

▶ **Definition 3.** *A* Byzantine trust network *(BTN) is a tuple $\mathcal{T} = \langle N, H, T_i, \mathcal{C}_i \rangle$ where:*

- *$N = \{1, \ldots, n\}$ is a finite set of nodes.*
- *$H \subseteq N$ is the set of honest nodes. $B = N \backslash H$ is the set of Byzantine nodes. We denote with $b_i = \frac{|B \cap T_i|}{|T_i|}$ the ratio of Byzantine nodes in T_i, and with $b = \max \{b_i\}_{i \in H}$ the largest such ratio.*
- *$T_i \subseteq N$, for each node $i \in H$, is the non-empty set of nodes that i trusts, i.e., its* trust set.[5]
- *$\mathcal{C}_i \subset 2^{T_i}$, with $i \in H$. We refer to $\mathcal{C}_i$ as the set of* winning coalitions *for node i. For ease of presentation we will refer to winning coalitions also via a function $\mathcal{C} : N \to 2^{2^N}$ assigning the set $\mathcal{C}_i$ of subsets of N to each $i \in H$.*

We will sometimes assume that, for all i, $i \in T_i$. We will also sometimes assume that for all $i \in H$, $\{i\} \in \mathcal{C}_i$. In such a case the BTN is said to be vetoed.

Intuitively, a winning coalition $C \in \mathcal{C}_i$ for i is a set of nodes such that, if all its members agree from the perspective of i, then their opinion is validated by i (cf. [32]). We define the notion of validation formally below (Definition 8). When $\{i\}$ belongs to $\mathcal{C}_i$, i cannot validate an opinion unless it also holds such opinion (it holds a veto for its own validation).

▶ Remark 4. In the Stellar white paper [28] BTN are referred to as *federated Byzantine agreement systems* (FBAS), or as *federated Byzantine quorum systems* in [17], and the winning coalitions of a node are referred to as *quorum slices*. BTNs are also known structures in the economic theory literature, where they are referred to as *command games* [20, 19], or as *simple game structures* [22].

A natural class of BTNs is obtained by associating a quota, or threshold, $q_i \in (0.5, 1]$ to each honest node i:[6]

▶ **Definition 5.** *A Quota Byzantine Trust Network (QBTN) is a BTN such that for all $i \in H$ there exists a quota $q_i \in (0.5, 1]$ such that:*

$$\mathcal{C}_i = \{C \subseteq T_i \mid |C| \geq q_i \cdot |T_i|\}.$$

A QBTN is therefore denoted by a tuple $\mathcal{T} = \langle N, H, T_i, q_i \rangle$. A QBTN is said to be uniform *whenever $q_i = q_j$ for any $i, j \in H$. It is said to be* effective *whenever $q_i \in (0.5 + b, 1 - b]$ for any $i \in H$.*

Intuitively, QBTNs are BTNs where the winning coalitions of a node are determined by a numerical quota: i's validation is determined whenever at least a fraction q_i of nodes in T_i hold that opinion. This quota should be set in such a way that: *i)* the quota is met whenever the honest nodes in T_i agree, and *ii)* if the quota is met for an opinion x, then there is at least

[5] Ripple and Stellar refer to trust sets as unique node lists (UNLs).
[6] Cf. [16].

an honest majority of nodes with opinion x in the trust set of i. That is, $q_i \in (0.5 + b, 1 - b]$, which is the case for what we called effective QBTNs. For this constraint to be met the fraction of Byzantine nodes in each T_i should therefore fall in the interval $[0, 0.25)$.

▶ Remark 6. It is worth observing that in this context the largest possible ration of Byzantine nodes b in a BTN plays a slightly different role than in the standard BFT failure models where, classically, safety would require $b < \frac{1}{3}|T_i|$ [31, 8, 25]. In the standard BFT failure models one is interested in making sure that any two simple majorities intersect in some honest node in order for the system not to fork. In a BTN, the role of b is slightly different: it is to guarantee that an honest node can always safely validate an opinion because a honest majority exists in its trust set, that agrees on that opinion.

The Ripple consensus protocol [6, 37] is based on uniform QBTNs with quotas $q_i = 0.8$ and $b = 0.2$. The Stellar consensus protocol as described in [28] does not rely on QBTNs but requires the more general class of vetoed BTNs.

▶ Remark 7. In Definition 3 we associate winning coalitions only to honest nodes. We do this for simplicity but it should be clear that trivial collections of winning coalitions can be associate also to Byzantine nodes. Since the validation by a Byzantine node i is not influenced by any other node its trivial collection of winning coalitions is the set $\{C \subseteq N \mid \{i\} \subseteq C\}$, that is, the set of all coalitions containing i. Intuitively, this amounts to stating that i is the only node influencing its own opinion.

2.3 Validation and Safety

Intuitively, a node i validates an opinion whenever a winning coalition of nodes trusted by i agrees on that opinion, from the perspective of i. Given an opinion profile $\mathbf{o}$ we denote by

$$T_j^{\mathbf{o}}(x) = \{i \in T_j \mid \mathbf{o}(i) = x \text{ if } i \in H, \text{ and } \mathbf{o}(i)(j) = x \text{ if } i \in B\} \tag{1}$$

the set of nodes trusted by i who hold opinion x (if they are honest), or reveal opinion x to i (if they are Byzantine).

▶ **Definition 8.** *Let a BTN and an opinion profile* $\mathbf{o}$ *be given. Way say that* $i \in N$ validates $x \in \{0, 1\}$ *(in* $\mathbf{o}$*) if* $T_i^{\mathbf{o}}(x) \in \mathcal{C}_i$.

In a QBTN, an honest node i validates an opinion whenever there are at least $q_i \cdot |T_i|$ nodes with the same opinion among the nodes it trusts.

▶ Remark 9. BTNs are a generalization of so-called Byzantine quorum systems [27]. Each BTN naturally induces the set of quora $\{Q \subseteq N \mid \forall i \in Q, \exists C \in \mathcal{C}_i : C \subseteq Q\}$. In words, a *quorum* is a set of nodes that contains a winning coalition for each node in the set (cf. [28]). Intuitively, it is a set of nodes that have the means to stably validate an opinion. For a set of quora to be a Byzantine quorum system, quora also need to pairwise intersect. We will come back to this in Section 3.2.

To introduce safety formally, we need some auxiliary notions. Let $s : N \to 2^N$ be a function picking, for any agent i, one coalition out of $\mathcal{C}_i$. Each function s induces an operator $F_s : 2^N \to 2^N$ such that $F_s(C) = \bigcup_{i \in C} s(i)$, collecting, for each i in C, the winning coalition $s(i)$ picked by function s. We can then recursively construct sets of nodes by joining winning coalitions of nodes in earlier sets. Such a construction reaches a fixpoint where for each node in the set a winning coalition is already contained in the set. Formally, for $C \subseteq N$: $F_s^0(C) = C$ and $F_s^{n+1}(C) = F_s(\bigcup_{0 \leq k \leq n} F_s^k(C))$. Define then $F_s^{\star}(C) = \bigcup_{0 \leq n} F_s^n(C)$. As N

is finite, there exists a non-negative integer n such that $F_s^n(C) = F_s^\star(C)$. Observe that for any $C \subseteq N$, $F_s^\star(C)$ is such that for all $i \in F_s^\star(C)$ there is a $D \in \mathcal{C}_i$ such that $D \subseteq F_s^\star(C)$. That is, $F_s^\star(C)$ is a quorum (cf. Remark 9).

▶ **Definition 10.** *Let a BTN and an opinion profile* $\mathbf{o}$ *be given. We then say that an opinion profile* $\mathbf{o}$ *is:*

- forked *(or, is a fork) if there are two honest nodes $i, j \in H$ such that i validates x and j validates $\overline{x}$ in* $\mathbf{o}$.
- strongly forked *(or, is a strong fork) if there are two honest nodes $i, j \in H$ and a function s such that all nodes in $F_s^\star(\{i\})$ agree on x and all nodes in $F_s^\star(\{j\})$ agree on $\overline{x}$.*

So, a fork is a profile where two honest nodes i and j have validated two different opinions. A strong fork is a fork where, in addition, there is a winning coalition for i agreeing on x and a winning coalition for j agreeing on $\overline{x}$, *and* all nodes in that winning coalition for i also have a winning coalition agreeing on x and all nodes in that winning coalition for j have a winning coalition agreeing on $\overline{x}$, *and* so on. In short, a strong fork is a "stable" fork.

▶ **Definition 11.** *A BTN is* safe *if there exists no forked profile for it. It is* weakly safe *if there exists no strongly forked profile for it.*

Safety rules out the possibility that two honest nodes may settle on different opinions, and therefore the possibility that the run of any consensus protocol on the BTN would generate a stream of opinion profiles that contains a profile where two nodes have validated different values. Weak safety allows for forks of only a limited kind. It rules out the possibility that forks are of a "deep" kind in the sense that they involve all winning coalitions upon which the diverging opinions are rooted. Clearly safety implies weak safety but not vice versa.

It is finally worth stressing that the above notions of safety and weak safety are protocol independent: they concern all consensus protocols where validation depends locally on the values relayed by trusted nodes on a given BTN, like the Ripple or Stellar protocols.[7]

3 (De)centralisation and (In)tractability

This section explores inherent limitations present in the above notion of safety for BTNs, establishing general limitative results for the class of consensus protocols based on them, such as Ripple and Stellar. First, it focuses on uniform QBTNs (Definition 5), as exemplified by the Ripple consensus protocol, showing that safety drastically limits the freedom of nodes in selecting trustees. Second, it focuses on safety for general BTNs (Definition 3), as exemplified by the Stellar consensus protocol, showing that, even though safety in such settings allows for more freedom on the part of nodes, it does require single nodes to solve decision problems that are, in principle, computationally intractable.

3.1 Safety Implies Centralization in Uniform QBTNs

We show that the safety of uniform QBTNs implies that nodes cannot be fully free to choose their trust set. The result builds on ideas and techniques from [6].

We first define some notation:

$$\beta_{ij} = \min\left\{|T_i \cap T_j|, b \cdot |T_i|, b \cdot |T_j|\right\}. \tag{2}$$

[7] However, it is worth stressing we are not considering protocols where validation may depend on information richer than just nodes' opinions.

Intuitively, β_{ij} denotes the maximum possible number of Byzantine nodes present in the intersection of the trust sets of i and j. Such a number equals the maximum amount of Byzantine nodes assumed by the node, either i or j, that tolerates fewer Byzantine nodes. Such a number cannot obviously exceed the size of the intersection itself.

▶ **Lemma 12** ([6])**.** *Let $\langle N, H, T_i, q_i \rangle$ be an effective QBTN. For any profile $\mathbf{o}$ and node $i \in H$, if $|T_i^{\mathbf{o}}(x)| > 0$ then for any $j \in H$,*

$$|T_j^{\mathbf{o}}(x) \cap H| \geq |T_i \cap T_j| + |T_i^{\mathbf{o}}(x)| - |T_i| - \beta_{ij} \tag{3}$$

$$|T_j^{\mathbf{o}}(\overline{x})| \leq |T_j| - |T_i \cap T_j| - |T_i^{\mathbf{o}}(x)| + |T_i| + \beta_{ij} \tag{4}$$

This lemma establishes a lower bound on the number of honest nodes with opinion x that a honest node j can observe in its trust set, given the number of nodes (not necessarily honest) that another honest node i observes. It is used in the proof of Lemma 14. Notice that the bound in (3) and (4) are not necessarily strict, as illustrated in the following example.

▶ **Example 13.** Let $\langle N, H, T_i, q_i \rangle$ be such that: $N = \{1, \ldots, 9\}$, $B = \{5\}$ (recall $N = H \cup B$), $T_1 = T_2 = T_3 = T_4 = \{1, 2, 3, 4, 5\}$ and $T_6 = T_7 = T_8 = T_9 = \{5, 6, 7, 8, 9\}$, $q_1 = q_2 = q_4 = q_5 = 1 - \frac{1}{9}$. Let then $\mathbf{o}$ be such that $\mathbf{o}(1) = \mathbf{o}(2) = \mathbf{o}(3) = \mathbf{o}(4) = 1$, $\mathbf{o}(6) = \mathbf{o}(7) = \mathbf{o}(8) = \mathbf{o}(9) = 0$, finally $\mathbf{o}(5)$ be such that $\mathbf{o}(5)(1) = \mathbf{o}(5)(2) = \mathbf{o}(5)(3) = \mathbf{o}(5)(4) = 0$ and $\mathbf{o}(5)(6) = \mathbf{o}(5)(7) = \mathbf{o}(5)(8) = \mathbf{o}(5)(9) = 1$. So no honest node in $\{1, 2, 3, 4, 5\}$ can see a honest node with opinion 0 and, vice versa, no honest node in $\{5, 6, 7, 8, 9\}$ can see a honest node with opinion 1. But honest nodes in both set can see a (Byzantine) node with opposite opinion. Let now $j = 1$ and $i = 2$. We have, $|T_j^1 \cap H| = 5 + 5 - 5 - 1 = 4$ and $|T_j^0| = 5 - 5 - 5 + 5 + 1 = 1$.

▶ **Lemma 14.** *Let $\mathcal{T} = \langle N, H, T_i, q_i \rangle$ be a safe, uniform and effective QBTN. Then for all $i, j \in H$: $|T_i \cap T_j| > \frac{b}{1-b} \cdot (|T_i| + |T_j|)$.*

Notice that the maximum size of the intersection of two trust sets is obviously $\frac{1}{2}(|T_i| + |T_j|)$.[8] Lemma 14 establishes a lower bound on the size of the intersection of trust sets required by safety. The intuition behind the lemma is the following one. In order to make it impossible for two honest nodes to validate opposite values, their trust sets should intersect to the extent that any two winning coalitions for the two nodes would also have to intersect *and* contain at least one honest node.

▶ **Lemma 15.** *Let $\langle N, H, T_i, q_i \rangle$ be a uniform BTN. If for all $i, j \in H$, $|T_i \cap T_j| > 0.25 \cdot (|T_i| + |T_j|)$, then $\bigcap_{i \in H} T_i \neq \emptyset$.*

Intuitively, if any two trust sets have a large enough intersection (larger than a quarter of their combined size or, equivalently, larger than half their average size) then they must all intersect.

▶ **Theorem 16.** *In uniform QBTNs with effective quotas, if $b \geq 0.2$ then safety implies the existence of nodes that are trusted by all honest nodes.*

Proof. The result follows directly from Lemmas 14 and 15 and the observations that: in effective QBTNs where $b \geq 0.2$ it follows that $b \leq 0.25$; and for $b \in [0.2, 0.25]$ we have that $\frac{b}{1-b} \geq 0.25$ as desired. ◀

[8] This is the case, for instance, in QBTNs where $|T_i| = |T_j|$ for all $i, j \in H$.

$$\begin{tikzcd}
 & 1 \arrow[dl, leftrightarrow] \arrow[dr, leftrightarrow] & & & & 4 \arrow[dl, leftrightarrow] \arrow[dr, leftrightarrow] & \\
2 \arrow[rr, leftrightarrow] & & 3 & & 5 \arrow[rr, leftrightarrow] & & 6
\end{tikzcd}$$

Figure 1 Example from [28] of a vetoed BTN lacking QI. Arrows denote which nodes each node trusts (reflexive arrows omitted). $\mathcal{C}(1) = \mathcal{C}(2) = \mathcal{C}(3) = \{\{1,2,3\}\}$ and $\mathcal{C}(4) = \mathcal{C}(5) = \mathcal{C}(6) = \{\{4,5,6\}\}$.

If we understand decentralisation as the property of trust networks in which nodes have full freedom on whom to trust in the network, then the theorem can be interpreted as a general impossibility result for consensus based on QBTNs: if quotas are uniform and set appropriately w.r.t. the postulated maximum share of Byzantine nodes in trust sets, and if such share is large enough, then the existence of nodes that are trusted by everyone is a necessary condition for the safety of consensus. Furthermore, beyond limiting the choice of nodes, a (limited) set of such nodes clearly represents a dominant position and a risk factor.

In general, Theorem 16 is relevant for any consensus protocol that could be run on uniform QBTNs. In particular, it applies to the Ripple consensus protocol when the maximum fraction b of Byzantine nodes in trust sets is set to 0.2. In a way, Theorem 16 provides an ex-post analytical justification to the current design of the Ripple trust network where all trust sets are required to include a same subset of nodes (cf. [6]). Currently Ripple relies on a single UNL mostly controlled by Ripple, although plans for further decentralisation are under discussion.[9]

3.2 Safety and Quorum Intersection in BTNs

In this and the next section we consider the general case of (vetoed) BTNs, to which Theorem 16 does not apply. This more general setting applies instead to Stellar as described in its white paper [28], where the Stellar consensus protocol does not presuppose uniformity of quotas. Actually, Stellar aims to offer open membership and freedom in choosing its own trust networks, which, together with BFT good scalability, would yield a decentralised and efficient blockchain. In such a setting an intuitive necessary condition for safety is that trust networks are sufficiently "interconnected", in the following sense.

▶ **Definition 17** ([28]). *A vetoed BTN enjoys quorum intersection (QI) whenever for any two sets $Q_1, Q_2 \subseteq H$, if Q_1 and Q_2 are quora, then $Q_1 \cap Q_2 \neq \emptyset$.*

▶ **Example 18.** In Figure 1 the quora are $\{1,2,3\}, \{4,5,6\}, \{1,2,3,4,5,6\}$. This BTN does not enjoy QI, but both of its disjoint components (with support $\{1,2,3\}$ and $\{4,5,6\}$) do. Suppose instead that $\{1,2,3,5\} \in \mathcal{C}(3)$, that is, 3 also trusts 5. Then the system would satisfy QI with quora: $\{4,5,6\}, \{1,2,3,4,5,6\}$.

▶ **Example 19.** In a BTN $\mathcal{T}$ where $\forall i \in N$, $\mathcal{C}(i) = \{N\}$, the set N of all nodes is the unique quorum, and $\mathcal{T}$ trivially enjoys quorum intersection.

In fact, there is a close relationship between quorum intersection and the property of weak safety:

▶ **Theorem 20.** *A vetoed BTN is weakly safe iff any two quora intersect and such intersection contains at least one honest node.*

[9] Cf. `https://xrpcharts.ripple.com/`.

Clearly, nodes in a BTN cannot know which nodes are Byzantine so their best effort in order to guarantee weak safety is to guarantee QI is not violated.[10]

3.3 The Intractability of Maintaining Quorum Intersection

Quorum intersection is in fact assumed by all the existing correctness analyses of Stellar [28, 17]. It is furthermore stressed in [28, p. 9] that: "[...] it is the responsibility of each node i to ensure $\mathcal{C}_i$ [notation adapted] does not violate quorum intersection." The key question, from a safety perspective, becomes therefore whether single nodes can reasonably be tasked with maintaining QI. Apart from incentive issues, which have also been flagged [23], we argue that this is a problematic requirement from a merely computational standpoint. This might not be an issue in the current, small-scale, Stellar configuration (although an instance of QI failure has been recently reported [26]), but it is something to be considered in a path towards full decentralisation with a full-scale number of nodes and validators. As our analysis below shows, maintaining QI is a computationally intractable problem.

We present two results. First we show that deciding whether a given BTN satisfies QI is intractable.[11] Second, we show that deciding whether adding a new trust set with winning coalitions preserves QI on a given BTN is also computationally intractable (again coNP-hard). This is arguably the decision problem that nodes need to solve when linking to the Stellar network. We argue that these results point to a possible computational bottleneck for the scalability of the consensus model of Stellar.

We first define the problem of deciding whether QI holds in a given BTN.
Quorum-Intersection

Input: A BTN $\mathcal{T} = \langle N, \mathcal{C} \rangle$ where the sets $\mathcal{C}(i)$ for $i \in N$ are listed explicitly.[12]
Question: Is it the case that for each two quora Q_1, Q_2, $Q_1 \cap Q_2 \neq \emptyset$?

▶ **Theorem 21.** *Quorum-Intersection is coNP-complete.*

The intractability result of Theorem 21 says that it may be computationally hard, in practice, to check QI. Such a result is robust in the sense that the related problem of checking whether QI holds after the insertion of one new winning coalition by a node into a system that already satisfies QI, is also coNP-complete. (actually coNP-hard).

Coalition-Addition-Quorum-Intersection

Input: Two BTNs $\mathcal{T} = \langle N, \mathcal{C} \rangle$ and $\mathcal{T}' = \langle N, \mathcal{C}' \rangle$, such that $\mathcal{T}'$ satisfies QI, and such that $\mathcal{T}$ is obtained from $\mathcal{T}'$ by adding one single coalition to $\mathcal{C}'(i)$ for some $i \in N$, where the sets $\mathcal{C}(i)$ and $\mathcal{C}'(i)$ for all $i \in N$ are listed explicitly.
Question: Is it the case that for each two quora Q_1, Q_2 of $\mathcal{T}$ $Q_1 \cap Q_2 \neq \emptyset$?

▶ **Theorem 22.** *Coalition-Addition-Quorum-Intersection is coNP-complete.*

[10] However, a BTN could be complemented by a more fine-grained failure model consisting of a set of sets of possible Byzantine nodes representing the possible failure scenarios that nodes may encounter (cf. [17]).
[11] An equivalent result has also been recently presented in [24]. That paper provides a proof of NP-completeness (via reduction from the Set Splitting Problem) of the complementary problem for which we prove coNP-completeness (via 3SAT).
[12] For the purpose of this and the following result we do not need to take into consideration the H and T_i elements of a BTN (Definition 3). We therefore omit them for conciseness.

4 Quantifying Influence on Consensus in BTNs

Theorem 16 showed that, in uniform QBTNs, safety implies the existence of nodes that are trusted by all honest nodes. While this can definitely be interpreted as a high level of centralisation required by safety, it is worth trying to precisely quantify the effect that the existence of all-trusted nodes has on consensus. In PoW and PoS protocols it is straightforward, at least by first approximation, to understand what the influence of each node is on the consensus process: each node will be able to determine a fraction of blocks corresponding to the node's share of total hashing power (PoW) or of total stakes (PoS). For consensus based on voting over trust structures, like in Ripple and Stellar, quantifying nodes' influence in a principled way is not obvious. This section proposes a methodology for such quantification that leverages the theory of voting games.

4.1 Influence Matrices

A BTN (Definition 3) associates to each honest node i a structure $\langle T_i, \mathcal{C}_i \rangle$. Such structures are known in game theory as simple games, that is a set of agents endowed with a set of winning coalitions. Such structures have been extensively studied to provide exact quantifications of power, for instance, in voting. In this section we show how techniques based on simple games provide a principled way to quantify influence in BTNs. We study the influence of j over i in a BTN as the power of j in the simple game that the BTN associates to i.

The *Penrose-Banzhaf index* [33, 1] of a node j in the simple game $\langle T_i, \mathcal{C}_i \rangle$ of node i is

$$\mathsf{B}_i(j) = \frac{1}{2^{n-1}} \sum_{C \subseteq N \setminus \{j\}} v(C \cup \{j\}) - v(C) \tag{5}$$

where v is the characteristic function of $\mathcal{C}_i$.[13] Essentially, the index counts the number of times in which j is decisive in turning a losing coalition into a winning one, that is, one that can determine the validation of an opinion by i. Equivalently, $\mathsf{B}_i(j)$ can be interpreted as the probability that j determines whether i validates a specific opinion, assuming all other agents in T_i have opinions distributed uniformly at random.

The normalised version $\mathsf{NB}_i(j)$ of the Penrose-Banzhaf index is:

$$\mathsf{NB}_i(j) = \frac{\mathsf{B}_i(j)}{\sum_{k \in N} \mathsf{B}_i(k)} \tag{6}$$

For any agent $j \notin T_i$ we stipulate $\mathsf{NB}(j) = 0$, as nodes that i does not trust cannot influence i's opinion directly – in game-theoretic jargon they are "dummy agents" in i's simple game. Byzantine nodes are assigned degenerate simple games containing a singleton winning coalition that has themselves as only member (cf. Remark 7 above). Byzantine agents cannot be influenced: for all $i \in N \setminus H$ from the degenerate simple game associated to i, $\mathsf{NB}_i(i) = 1$, and $\mathsf{NB}_i(j) = 0$ for each $j \neq i$.

Given a BTN, we associate to each honest node i a vector $[\mathsf{NB}_i(1), \ldots, \mathsf{NB}_i(n)]$ of normalized Penrose-Banzhaf indices capturing the influence that each node has on i. Clearly $\sum_{j \in N} \mathsf{NB}_i(j) = 1$ and $\mathsf{NB}(j) > 0$ only if $j \in T_i$, for any $i \in N$. Notice that the vector of a Byzantine node i is therefore degenerate: $\mathsf{NB}_i(i) = 1$ and $\mathsf{NB}_i(j) = 0$ for each $j \neq i$. It

[13] That is, for any $C \subseteq N$, $v(C) = 1$ iff $C \in \mathcal{C}_i$.

follows that each BTN $\mathcal{T}$ induces a stochastic $N \times N$ matrix

$$I(\mathcal{T}) = \begin{bmatrix} I_{11} & I_{12} & I_{13} & \dots & I_{1n} \\ I_{21} & I_{22} & I_{23} & \dots & I_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ I_{n1} & I_{n2} & I_{n3} & \dots & I_{nn} \end{bmatrix}$$

where I_{ij} denotes the normalized Penrose-Banzhaf index $\mathsf{NB}_i(j)$ of node j in the simple game associated to i. We call such matrix $I(\mathcal{T}) = [I_{ij}]_{i,j \in N}$ the *influence matrix* (of $\mathcal{T}$). We will drop reference to $\mathcal{T}$ when no confusion arises. The matrix encodes the direct influence that each node has on each other in the sense of being decisive for the validation of an opinion. Matrices of this type have a long history in the mathematical modeling of influence in economics and the social sciences dating back to [13, 7], and have recently received renewed attention [21].[14] Similar matrices, but based on the Shapley-Shubik power index [38] instead of the Penrose-Banzhaf one, have been studied in [20, 19].[15]

▶ **Example 23.** Consider the following BTN with no Byzantine nodes and consisting of 6 agents all having a same set of 5 agents as trust set: $N = H = \{1, 2, 3, 4, 5, 6\}$, $T_i = \{1, 2, 3, 4, 5\}$ for all $i \in N$ and $q_i = 0.8$ for all $i \in N$. By (5) and (6) for each $i \in \{1, 2, 3, 4, 5, 6\}$ we have $\mathsf{B}_j(i) = \frac{2}{8}$ and $\mathsf{NB}_j(i) = \frac{1}{5}$, for each node $j \in \{1, 2, 3, 4, 5\}$. The influence matrix describing this BTN consists of 6 identical row vectors $\begin{bmatrix}\frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0.\end{bmatrix}$ Consider now a variant of the above BTN where node 5 is Byzantine. The influence matrix describing this variant consists of 5 identical vectors $\begin{bmatrix}\frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0.\end{bmatrix}$ for the rows corresponding to nodes $1 - 4$ and 6, and the degenerate row vector $\begin{bmatrix}0 & 0 & 0 & 0 & 1 & 0.\end{bmatrix}$ for the row of node 5. That is, all the nodes in T_i have the same influence on honest nodes, but no honest node influences 5.

4.2 Limit Influence

Influence matrices describe the extent of direct influence between nodes. But influence is not only direct: by influencing nodes k that in turn influence node i, a node j can exert indirect influence on the opinions that i validates. This type of indirect influence is captured by the powers of the influence matrix:

$I^1_{ij} = I_{ij}$ represents the probability that j can directly sway i to validate a value x. This is j's direct influence on i.

$I^2_{ij} = \sum_{k \in N} I_{ik} \cdot I_{kj}$ represents the probability that j can sway i's validation in two steps, by swaying the validation of intermediate nodes k which in turn sway i's validation directly. This is j's indirect (2-step) influence on i.

I^k_{ij} more generally represents j's indirect (k-step) influence on i.

So the influence (direct or indirect) of j on i in a BTN is given by the total probability of all ways in which j can determine the value of i's validation. Formally this amounts to $\lim_{t\to\infty}(I^t)_{ij}$, provided such limit exists. In yet other words, this denotes the likelihood that j is decisive for i to validate an opinion.

We are then in the position to quantify what the influence is of each node on every other node by taking the limit of the power of the influence matrix of the BTN $\mathcal{T}$, that is:

$$I(\mathcal{T})^\infty = \lim_{t \to \infty} I(\mathcal{T})^t \tag{7}$$

[14] See also [34, 35] for an overview of such models.
[15] For a comparison between these two power indeces we refer the reader to [11].

If the limit matrix in (7) exists, we say that the influence matrix $I(\mathcal{T})$ is *regular*. We say that it is *fully regular* when its limit matrix exists and it is such that all rows are identical.[16] Intuitively, regularity means that it is possible to precisely quantify the influence of each node on each other node; full regularity means that every node has the same influence on every other node.

▶ **Example 24.** Consider again the two BTNs introduced in Example 23. In the first case, where all nodes are honest, all nodes belonging to some trust set have positive and – given the symmetry built in the example – the same influence:

$$\begin{bmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 \end{bmatrix}^2 = \lim_{t\to\infty} \begin{bmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 \end{bmatrix}^t$$

In the second case, where node 5 is Byzantine, the only node having positive influence (total influence 1 in this example) is precisely 5:

$$\lim_{t\to\infty} \begin{bmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 \end{bmatrix}^t = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

In other words, the only node having influence on which values will be validated by other nodes, and therefore on agreement, is the Byzantine node.

4.3 Limit Influence in Ripple and Stellar

Theorem 16 established that in uniform QBTNs, and therefore Ripple, safety requires centralisation in the sense of demanding the existence of a non-empty set of nodes trusted by all other nodes. While this does not apply in general to Stellar, recent studies have highlighted that Stellar enjoys a similar level of centralisation.[17]

Here we put the above methodology at work to study limit influence in centralised BTNs, that is BTNs where nodes exist that are trusted by all nodes. We show (Theorem 25) that: the existence of nodes trusted by all nodes makes it possible to establish limit influence (first claim); this limit influence is such that every node has the same limit influence on every other node (second claim) when at most one Byzantine node exists in the BTN; but if only just one all-trusted node trusts a Byzantine node, no honest node has limit influence on any other honest node (third claim). That is, in a centralised BTN the power of deciding whether an opinion can become consensus or not is, in principle, all in the hands of Byzantine nodes.

[16] The "regularity" and "full regularity" terminology are borrowed from [14] and [34].
[17] Data analysis of the current Stellar network has shown [23] that one of the three Stellar foundations validators is included in all trust sets. If we treat the Stellar foundation to be operating as one node, Stellar satisfies *de facto* the same level of centralisation that we have shown is analytically required for Ripple.

▶ **Theorem 25.** *Let $\mathcal{T}$ be a BTN. If $\mathcal{T}$ is such that $\bigcap_{i \in H} T_i \neq \emptyset$ then:*
1) *$I(\mathcal{T})$ is regular;*
2) *$I(\mathcal{T})$ is fully regular if in addition $\mathcal{T}$ is such that $|B| \leq 1$;*
3) *and, if there exists $j \in \left(\bigcap_{i \in H} T_i\right) \cap H$ such that $T_j \cap B \neq \emptyset$ then for all $j, k \in H$, $I(\mathcal{T})^{\infty}_{jk} = 0$.*

Again, it is worth noticing that this is a general protocol-independent result: it concerns all protocols working on centralized BTNs where consensus is determined through validations locally dependent on trusted nodes. In particular, it applies to the setup of the Ripple trust network under the assumption of safety (by Theorem 16) and to the current setup of the Stellar trust network.

5 Conclusions

We addressed decentralisation in the specific context of BFT consensus based on open quorum systems, showcasing the relevance of tools from economic theory (command games, power indices) and computational complexity theory. In doing so we focused on a general class of consensus, linking decentralisation to a precise measure of the influence of each peer in the network (a theme extensively studied in economics), an analysis of the structural properties of the consensus network, and the computational complexity of problems related to safe consensus. The obtained limiting results on Ripple and Stellar are coherent with the current practice and the proposals that industry is putting forward to improve decentralisation. We argue that the obtained results show this is a promising approach to the formal analysis of decentralisation.

Our results point to several avenues of future research. We are planning to extend our analysis to other blockchains based on BFT consensus that are currently being developed, noticeably Cobalt [10] as an evolution of the Ripple/Stellar tradition. More generally, we also want to explore the applicability of the methodology beyond the framework of Byzantine trust networks, since measures of the relative influence of peers are of interest for other blockchain frameworks, e.g. PoS. At the same time, we also intend to build on such measures to address the relationships between influence, decentralisation and, crucially, revenue. Properly understanding such mechanisms will serve to the long-term goal of designing more reliable and robust blockchains. On the application side, the development of a prototype analysis toolkit and collection of relevant data is also an ongoing activity.

References

1 J. Banzhaf. Weighted voting doesn't work: A mathematical analysis. *Rutgeres Law Review*, 19:317–343, 1965.

2 J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. Kroll, and E. Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 104–121. IEEE, 2015.

3 F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. Procaccia, editors. *Handbook of Computational Social Choice*. Cambridge University Press, 2016.

4 C. Cachin and M. Vukolic. Blockchain consensus protocols in the wild. Technical report, CoRR abs/1707.01873, 2017.

5 M. Castro and B. Liskov. Practical byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, pages 173–186, Berkeley, CA, USA, 1999. USENIX Association. URL: `http://dl.acm.org/citation.cfm?id=296806.296824`.

6 B. Chase and MacBrough E. Analysis of the XRP ledger consensus protocol. Technical report, Ripple Research, 2018.

7 Morris H. DeGroot. Reaching a Consensus. *Journal of the American Statistical Association*, 69(345):118–121, 1974.

8 D. Dolev. Byzantine generals stike again. *Journal of Algorithms*, 3(1):14–30, 1982.

9 C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In Ernest F. Brickell, editor, *Advances in Cryptology — CRYPTO' 92*, pages 139–147, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

10 MacBrough E. Cobalt: BFT governance in open networks. Technical report, Ripple Research, 2018.

11 D. Felsenthal and M. Machover. *The Measurement of Voting Power*. Edward Elgar Publishing, 1998.

12 M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, April 1985. `doi:10.1145/3149.214121`.

13 J. French. A formal theory of social power. *Psychological Review*, 61:181–194, 1956.

14 F. Gantmacher. *The Theory of Matrices*. AMS Chealsea Publishing, 1959.

15 J. Garay, A. Kiayias, and Nikos L. The bitcoin backbone protocol: Analysis and applications. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*. Springer, 2015. `doi:10.1007/978-3-662-46803-6_10`.

16 H. Garcia-Molina and D. Barbara. How to assign votes in a distributed system. *J. ACM*, 32(4):841–860, October 1985. `doi:10.1145/4221.4223`.

17 C. García-Pérez and A. Gotsman. Federated Byzantine Quorum Systems. In J. Cao, F. Ellen, L. Rodrigues, and B. Ferreira, editors, *22nd International Conference on Principles of Distributed Systems (OPODIS 2018)*, volume 125 of *LIPIcs*, pages 17:1–17:16, 2018. `doi:10.4230/LIPIcs.OPODIS.2018.17`.

18 U. Grandi. Social choice on social networks. In U. Endriss, editor, *Trends in Computational Social Choice*, pages 169–184. COST, 2018.

19 X. Hu and L. Shapley. On authority distributions in organizations: Controls. *Games and Economic Behavior*, 45:153–170, 2003.

20 X. Hu and L. Shapley. On authority distributions in organizations: Equilibrium. *Games and Economic Behavior*, 45:132–152, 2003.

21 M. O. Jackson. *Social and Economic Networks*. Princeton University Press, Princeton, NJ, USA, 2008.

22 D. Karos and H. Peters. Indirect control and power in mutual control structures. *Games and Economic Behavior*, 92, 2015.

23 M. Kim, Y. Kwon, and Y. Kim. Is stellar as secure as you think? In *IEEE Security and Privacy on the Blockchain (IEEE S&B '19)*. IEEE, 2019.

24 L. Lachowski. Complexity of the quorum intersection property of the federated byzantine agreement system, 2019. URL: `https://arxiv.org/abs/1902.06493`.

25 L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

26 M. Lokhava, G. Losa, D. Maziéres, G. Hoare, N. Barry, E. Gafni, J. Jove, and Jed McCaleb R. Malinowsky. Fast and secure global payments with Stellar. In *Proceedings of SOSP'19*. ACM, 2019.

27 D. Malkhi and M. Reiter. Byzantine quorum systems. *Distributed Computing*, 11:203–213, 1998.

28 D. Mazierès. The stellar consensus protocol: A federated model for internet-level consensus. Stellar Development Foundation, 2016.

29 S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Bitcoin project white paper*, 2009.

30 A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder. *Bitcoin and Cryptocurrency Technologies*. Princeton University Press, 2016.

31 M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the Association for Computing Machinery*, 27(2):228–234, 1980.

32 D. Peleg. Local majorities, coalitions and monopolies in graphs: a review. *Theor. Comput. Sci.*, 282(2):231–257, 2002.

33 L. Penrose. The elementary statistics of majority voting. *Journal of the Royal Statistical Society*, 109(1):53–57, 1946.

34 A. Proskurnikov and R. Tempo. A tutorial on modeling and analysis of dynamic social networks. Part I. *Annual Reviews in Control*, 43:65–79, 2017.

35 A. Proskurnikov and R. Tempo. A tutorial on modeling and analysis of dynamic social networks. Part II. *Annual Reviews in Control*, 45:166–190, 2018.

36 F. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys*, 22(4):299–319, 1990.

37 D. Schwartz, N. Youngs, and A. Britto. The ripple protocol consensus algorithm. Technical report, Ripple Labs, 2014.

38 L. Shapley and M. Shubik. A method for evaluating the distribution of power in a committee system. *American Political Science Review*, 48:787–792, 1954.

39 M. Vukolic. *Quorum Systems with Applications to Storage and Consensus*. Morgan & Claypool Publishers, 2012.

40 M. Vukolic. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In *Proceedings of iNetSec'15*, volume 9591 of *LNCS*, pages 112–125, 2015.

41 R. Wattenhofer. *Distributed Ledger Technology: The Science of the Blockchain*. Createspace Independent Publishing Platform, 2017.

A Technical appendix

A.1 Proofs of Section 2

A.1.1 Lemma 12

Proof. $\boxed{(3)}$ By assumption $|T_i^{\mathbf{o}}(x)| > 0$. So j observes at least $|T_i^{\mathbf{o}}(x) \cap T_j \cap H|$ nodes with opinion x in T_j. Those are the honest nodes among the nodes with opinion x that both i and j can observe. So

$$|T_j^{\mathbf{o}}(x) \cap H| \geq |T_i^{\mathbf{o}}(x) \cap T_j \cap H|.$$

Now among the nodes in $T_i^{\mathbf{o}}(x) \cap T_j$ there are at most β_{ij} Byzantine nodes that could reveal the opposite opinion $\overline{x}$ to j. So,

$$|T_i^{\mathbf{o}}(x) \cap T_j \cap H| \geq |T_i^{\mathbf{o}}(x) \cap T_j| - \beta_{ij}.$$

The claim is finally established by the following series of (in)equalities:

$$\begin{aligned}|T_i^{\mathbf{o}}(x) \cap T_j \cap H| &\geq |T_i^{\mathbf{o}}(x) \cap T_j| - \beta_{ij}\\ &\geq |T_i^{\mathbf{o}}(x)| - |T_i \backslash T_j| - \beta_{ij}\\ &= |T_i^{\mathbf{o}}(x)| - |T_i| + |T_i \cap T_j| - \beta_{ij}\end{aligned}$$

$\boxed{(4)}$ By assumption $|T_i^{\mathbf{o}}(x)| > 0$. So, by (3), $|T_j^{\mathbf{o}}(x) \cap H| = |T_i \cap T_j| + |T_i^{\mathbf{o}}(x)| - |T_i| - \beta_{ij}$ whenever only the honest nodes in T_j have opinion x. It follows that

$$\begin{aligned}|T_j^{\mathbf{o}}(\overline{x})| &\leq |T_j| - (|T_i \cap T_j| + |T_i^{\mathbf{o}}(x)| - |T_i| - \beta_{ij})\\ &= |T_j| - |T_i \cap T_j| - |T_i^{\mathbf{o}}(x)| + |T_i| + \beta_{ij}.\end{aligned}$$

This completes the proof. ◀

A.1.2 Lemma 14

Proof. The proof consists of two sub-arguments. $\boxed{\text{First}}$ we show that safety implies that, for all $i, j \in H$:

$$|T_i \cap T_j| > b \cdot (|T_i| + |T_j|) + \beta_{ij} \tag{8}$$

By safety (Definition 11), if $|T_i^{\mathbf{o}}(x)| \geq q_i|T_i|$ with $i \in H$, then for all $j \in H$ $T_j^{\mathbf{o}}(\overline{x}) < q|T_j|$. Assume $T_i^{\mathbf{o}}(x) \geq q|T_i|$ with $i \in H$. By Lemma 12 and safety we have:

$$\begin{aligned} T_j^{\mathbf{o}}(\overline{x}) &\leq |T_j| - |T_i \cap T_j| - q|T_i| + |T_i| + \beta_{ij} \\ &< q|T_j|. \end{aligned}$$

From $|T_j| - |T_i \cap T_j| - q|T_i| + |T_i| + \beta_{ij} < q|T_j|$ and the assumption on the effectiveness of q (that is, $q \in (0.5 + b, 1 - b]$) we thus obtain

$$\begin{aligned} b \cdot (|T_i| + |T_j|) + \beta_{ij} &\leq (1 - q) \cdot (|T_i| + |T_j|) + \beta_{ij} \\ &= |T_j| - q|T_j| - q|T_i| + |T_i| + \beta_{ij} \\ &< |T_i \cap T_j| \end{aligned}$$

as desired.[18]

$\boxed{\text{Second}}$ We show that safety also implies that, for all $i, j \in H$:

$$b \cdot (|T_i| + |T_j|) + \beta_{ij} > \frac{b}{1-b}(|T_i| + |T_j|) \tag{9}$$

We have established that safety implies that for all $i, j \in H$, $|T_i \cap T_j| > b \cdot (|T_i| + |T_j|) + \beta_{ij}$ (8), that is, the size of the intersection of the trust sets of i and j should be larger than the maximum possible fraction of Byzantine nodes times the combined size of the trust sets, plus β_{ij}. Now recall the definition of β_{ij} (2). By (8) it cannot be the case that $\beta_{ij} = |T_i \cap T_j|$. So $\beta_{ij} = b|T_k|$ where T_k is the the smallest set between T_i and T_j. Now assume, w.l.o.g. that $|T_i| \geq |T_j|$ and so that $|T_j| = x \cdot |T_i|$ with $x \in (0, 1]$. By (8) we have:

$$\begin{aligned} |T_i \cap T_j| &> b(|T_i| + |T_j|) + \beta_{ij} \\ &= b(|T_i| + x|T_i|) + bx|T_i| \\ &= b|T_i|(1 + 2x) \end{aligned}$$

From this, and the fact that a set is always at least as large as its intersection with another we obtain a lower bound for x by the following series of inequalities:

$$\begin{aligned} x|T_i| &\geq |T_i \cap T_j| \\ x|T_i| &> b|T_i|(1 + 2x) \\ x &> b(1 + 2x) \\ x &> b + 2bx \\ x - 2bx &> b \\ x(1 - 2b) &> b \\ x &> \frac{b}{1 - 2b} \end{aligned}$$

[18] Cf. [6, Proposition 4].

By substituting $\frac{b}{1-2b}$ for x in (8) we thus obtain a lower bound for $|T_i \cap T_j|$ in b. We then reformulate (8) in terms of the combined size $\alpha = |T_i| + |T_j|$ of the two trust sets:

$$\begin{aligned}
|T_i \cap T_j| &> b(\underbrace{|T_i| + x|T_i|}_{\alpha}) + bx|T_i| \\
&> b(|T_i| + \frac{b}{1-2b}|T_i|) + b\frac{b}{1-2b}|T_i| \\
&= b\alpha + b\frac{b}{1-2b}\frac{1-2b}{1-b}\alpha && \text{as} |T_i| = \alpha\frac{1-2b}{1-b} \\
&= b\alpha(1 + \frac{b}{1-b}) \\
&= \frac{b}{1-b}\alpha
\end{aligned}$$

So safety implies that the size of the intersection of T_i and T_j must be larger than the fraction $\frac{b}{1-b}$ of the combined size of the two sets. ◀

A.1.3 Lemma 15

Proof. The proof is by induction on $|H|$. $\boxed{\text{Base}}$ if $|H| = 1$ the claim holds trivially. $\boxed{\text{Step}}$ Now assume the claim holds for $|H| = m$ (IH). We prove it holds for $|H| = m+1$. So assume for all $i, j \in H$, $|T_i \cap T_j| > 0.25 \cdot (|T_i| + |T_j|)$, and let k be the $m+1^{\text{th}}$ node in H. By IH we know that $\bigcap_{i \in H\setminus\{k\}} T_i \neq \emptyset$. Now take one of the smallest (w.r.t. size) T_i with $i \in H\setminus\{k\}$ and call it T_j. There are two cases. $\boxed{|T_k| \leq |T_j|}$. Then $|T_j \cap T_k| > 0.5 \cdot |T_k|$. Since T_j was smallest amongst the T_i, it also hols that $\forall i \in H$ $|T_i \cap T_k| > 0.5 \cdot |T_k|$. From this we conclude that $\bigcap_{i \in H} T_i \neq \emptyset$. $\boxed{|T_k| > |T_j|}$ Then $|T_j \cap T_k| > 0.5 \cdot |T_j|$. Since T_j was smallest amongst the T_i, it also hols that $\forall i \in H$ $|T_i \cap T_j| > 0.5 \cdot |T_j|$, from which we also conclude $\bigcap_{i \in H} T_i \neq \emptyset$. ◀

A.1.4 Theorem 20

Proof. $\boxed{\text{Left to right}}$ Straightforwardly proven by contraposition. $\boxed{\text{Right to left}}$ Proceed by contraposition and assume there is a profile $\mathbf{o}$, a function s and agents 1 and 2 such that all $k \in C_1 = F^\star_s(\{1\})$ agree on x and all $k \in C_2 = F^\star_s(\{2\})$ agree on $\overline{x}$. Observe that C_1 and C_2 are quora containing (since the BTN is vetoed) 1 and 2. There are two cases. Either $C_1 \cap C_2 = \emptyset$, or if that is not the case then $C_1 \cap C_2 \subseteq B$ as only Byzantine nodes can reveal different opinions to different nodes. Hence C_1 and C_2 are either disjoint or their intersection contains only Byzantine nodes. ◀

A.1.5 Theorem 21

Proof. To see that the problem is contained in coNP, we describe a nondeterministic polynomial-time algorithm to decide whether $\mathcal{T} = \langle N, \mathcal{C}\rangle$ does not have the quorum intersection property. The algorithm guesses two disjoint sets $Q_1, Q_2 \subseteq N$. Then, for each $u \in [2]$ and for each $i \in Q_u$, the algorithm checks if there is some $C \in \mathcal{C}(i)$ such that $C \subseteq Q_u$. That is, the algorithm verifies that Q_1 and Q_2 are quora (which is the case if and only if all checks succeed). Clearly, all checks can be performed in polynomial time. Thus, deciding whether $\mathcal{T}$ has the quorum intersection property is in coNP.

To show coNP-hardness, we reduce from the coNP-complete propositional unsatisfiability problem (UNSAT). Let φ be a propositional formula containing the propositional variables $x_1, \dots, x_n$. Without loss of generality, we may assume that φ is in 3CNF, i.e., that $\varphi = c_1 \wedge \dots \wedge c_m$ and that for each $j \in [m]$, $c_j = (T_{j,1} \vee T_{j,2} \vee T_{j,3})$, where $T_{j,1}, T_{j,2}, T_{j,3}$ are literals. We construct a command game $\mathcal{T} = \langle N, \mathcal{C} \rangle$ that has the quorum intersection property if and only if φ is unsatisfiable.

We let:

$$N = \{z_0, z_1\} \cup \{c_j \mid j \in [m]\} \cup \{y_i, p_i, n_i \mid i \in [n]\}.$$

That is, we have nodes z_0, z_1, a node c_j for each clause of φ, and nodes y_i, p_i, n_i for each variable occurring in φ.

We define the sets of winning coalitions of the nodes in N as follows:

$$\begin{aligned}
\mathcal{C}(z_0) &= \{\{z_0, y_1, \dots, y_n\}\}; & \\
\mathcal{C}(z_1) &= \{\{z_1, c_1, \dots, c_m\}\}; & \\
\mathcal{C}(y_i) &= \{\{y_i, p_i\}, \{y_i, n_i\}\} & \textit{foreach } i \in [n]; \\
\mathcal{C}(c_j) &= \{\{c_j, \sigma(T_{j,1})\}, \{c_j, \sigma(T_{j,2})\}, \{c_j, \sigma(T_{j,3})\}\} & \textit{foreach } j \in [m]; \\
\mathcal{C}(p_i) &= \{\{p_i, z_0\}, \{p_i, z_1\}\} & \textit{foreach } i \in [n]; \textit{ and} \\
\mathcal{C}(n_i) &= \{\{n_i, z_0\}, \{n_i, z_1\}\} & \textit{foreach } i \in [n];
\end{aligned}$$

where for each positive literal x_i, we let $\sigma(x_i) = p_i$; and for each negative literal $\neg x_i$, we let $\sigma(\neg x_i) = n_i$.

We argue that $\mathcal{T} = \langle N, \mathcal{C} \rangle$ has the quorum intersection property if and only if φ is unsatisfiable. We show the equivalent statement that $\mathcal{T} = \langle N, \mathcal{C} \rangle$ does **not** have the quorum intersection property if and only if φ is **satisfiable**.

($\Rightarrow$) Suppose that there exist $Q_1, Q_2 \in \mathbf{Q}^{\mathcal{T}}$ such that $Q_1 \cap Q_2 = \emptyset$. We may assume without loss of generality that Q_1 and Q_2 are core quora. We know that neither Q_1 nor Q_2 can contain both z_0 and z_1, because each quorum of $\mathcal{T}$ must contain either z_0 or z_1 (by the specific construction of $\mathcal{T}$). Thus, we may assume that $z_0 \in Q_2$ and $z_1 \in Q_1$.

Then also $\{y_1, \dots, y_n\} \subseteq Q_2$. Moreover, for each $i \in [n]$, we know that then either $p_i \in Q_2$ or $n_i \in Q_2$ (and not both). We also know that $\{c_1, \dots, c_m\} \subseteq Q_1$. Now define the truth assignment $\alpha : \{x_1, \dots, x_n\} \to \{0, 1\}$ as follows. For each $i \in [n]$, we let $\alpha(x_i) = 1$ if $n_i \in Q_2$ and we let $\alpha(x_i) = 0$ if $p_i \in Q_2$.

We show that α satisfies φ. Take an arbitrary clause c_j of φ. Due to the construction of $\mathcal{C}(c_j)$, we know that Q_1 contains (at least) one of $\sigma(T_{j,1}), \sigma(T_{j,2}), \sigma(T_{j,3})$. Take some $u \in [3]$ such that $\sigma(T_{j,u}) \in Q_1$. We show that α satisfies $T_{j,u}$. To derive a contradiction, suppose the contrary, i.e., that α does not satisfy $T_{j,u}$. Then $\sigma(T_{j,u}) \in Q_2$ (by the construction of α), and thus $Q_1 \cap Q_2 \neq \emptyset$, which contradicts our initial assumption that $Q_1 \cap Q_2 = \emptyset$. Thus, we can conclude that α satisfies $T_{j,u}$. This concludes our proof that φ is satisfiable.

($\Leftarrow$) Conversely, suppose that φ is satisfiable, i.e., that there is some truth assignment $\alpha : \{x_1, \dots, x_n\} \to \{0, 1\}$ that satisfies all clauses of φ. For each clause c_j, define $\rho(c_j)$ to be some literal $T_{j,u}$ in c_j that is satisfied by α. Moreover, for each $i \in [n]$, let $\mu(x_i) = n_i$ if $\alpha(x_i) = 1$ and $\mu(x_i) = p_i$ if $\alpha(x_i) = 0$. Consider the following two sets $Q_1, Q_2 \subseteq N$:

$$\begin{aligned}
Q_1 &= \{z_1, c_1, \dots, c_m\} \cup \{\sigma(\rho(c_j)) \mid j \in [m]\}; \text{ and} \\
Q_2 &= \{z_0, y_1, \dots, y_n\} \cup \{\mu(x_i) \mid i \in [n]\};
\end{aligned}$$

It is straightforward to verify that Q_1 and Q_2 are both quora, i.e., that $Q_1, Q_2 \in \mathbf{Q}^{\mathcal{T}}$. Moreover, since it holds that $Q_1 \cap Q_2 = \emptyset$, we know that $\mathcal{T}$ does not satisfy the quorum intersection property. ◀

A.1.6 Theorem 22

Proof sketch. Membership in coNP follows directly from Proposition 21. We show coNP-hardness by modifying the reduction given in the proof of Proposition 21. We describe a reduction from UNSAT. Let φ be a propositional formula containing the propositional variables $x_1, \ldots, x_n$. Without loss of generality, we may assume that φ is in 3CNF. Moreover, without loss of generality, we may assume that $\varphi[x_1 \mapsto 1]$ is unsatisfiable. We construct the command game $\mathcal{T} = \langle N, \mathcal{C} \rangle$ as in the proof of Proposition 21. Moreover, we transform $\mathcal{T}$ into $\mathcal{T}'$ by removing the coalition $\{y_1, n_1\}$ from $\mathcal{C}(y_1)$. By a similar argument as the one used in the proof of Proposition 21, we know that the command game $\mathcal{T}'$ satisfies the quorum intersection property, because $\varphi[x_1 \mapsto 1]$ is unsatisfiable. Moreover, $\mathcal{T}$ satisfies the quorum intersection property if and only if φ is unsatisfiable. ◀

A.2 Proofs of Section 4

A.2.1 Theorem 25

Proof. We first need to introduce some auxiliary notation. Given an influence matrix I, $\mathcal{G}(I) = \langle N, E \rangle$ denotes the (directed) graph of I, where $E = \{ij \mid I_{ji} > 0\}$. Intuitively $ji \in E$ whenever j influences i (i.e., has a positive Banzhaf-Penrose index in i's simple game). By assumption there exist nodes that influence all other honest nodes (themselves included). Let now $E(\bigcap_{i \in H} T_i) = \{i \in N \mid \exists j \in \bigcap_{i \in H} T_i, ij \in E\}$, that is, the set of nodes that influence some node that influences all honest nodes. We distinguish three cases.

$\boxed{|B| = 0}$ So there are no Byzantine nodes in $\mathcal{T}$, and $N = H = E(\bigcap_{i \in H} T_i)$ It follows that $\mathcal{G}(I(\mathcal{T}))$ is strongly connected (there exists a path from every node to every node) and aperiodic (there are no two cycles in the graph whose length is divided by an integer larger than 1). Trivially, it is also closed (there exists no node outside $\mathcal{G}(I(\mathcal{T}))$ that influences nodes in $\mathcal{G}(I(\mathcal{T}))$). The full regularity of I then follows from known results on influence matrices (cf. [34, Lemma 11]): if $\mathcal{G}(I)$ contains only one strongly connected component and is aperiodic, then I is fully regular.

$\boxed{|B| = 1 \text{ and } E(\bigcap_{i \in H} T_i) \cap B \neq \emptyset}$ So there exists exactly one Byzantine node in N, which furthermore belongs to $E(\bigcap_{i \in H} T_i)$ (that is, it influences at least one honest node influencing in turn all honest nodes). Call i such Byzantine agent. Recall that, by construction, $ii \in E$. So the subgraph consisting of i and the self-loop ii is the only closed, aperiodic, strongly connected component of $\mathcal{G}(I(\mathcal{T}))$. Full regularity therefore follows by know results as in the previous case.

$\boxed{|B| \geq 1 \text{ or } E(\bigcap_{i \in H} T_i) \cap B = \emptyset}$ So there exist several Byzantine nodes in N or there are Byzantine nodes which do not influence nodes in $\bigcap_{i \in H} T_i$. Both such cases determine the existence, by arguments analogous to those provided for the previous two cases, of several closed, aperiodic strongly connected components in $\mathcal{G}(I(\mathcal{T}))$. The regularity of I then follows again from known results on influence matrices (cf. [34, Theorem 12], [21, Theorem 8.1]: if all closed strongly connected components of $\mathcal{G}(I)$ are aperiodic, then I is regular.

In all three cases I is regular, proving claim 1). In the first two cases ($|B| \leq 1$) I is furthermore fully regular, establishing claim 2). Finally, to prove claim 3) we reason as follows. If there exists a honest agent in $\bigcap_{i \in H} T_i$ trusting a Byzantine agent, then the only closed strongly connected components of $\mathcal{G}(I)$ are the Byzantine nodes. In the limit, such nodes will therefore be the only ones having positive influence. ◀

VeriOSS: Using the Blockchain to Foster Bug Bounty Programs

Andrea Canidio
IMT School for Advanced Studies, Lucca, Italy
INSEAD, Fontainebleau, France
andrea.canidio@imtlucca.it

Gabriele Costa
IMT School for Advanced Studies, Lucca, Italy
gabriele.costa@imtlucca.it

Letterio Galletta
IMT School for Advanced Studies, Lucca, Italy
letterio.galletta@imtlucca.it

Abstract

Nowadays software is everywhere and this is particularly true for free and open source software (FOSS). Discovering bugs in FOSS projects is of paramount importance and many *bug bounty programs* attempt to attract skilled analysts by promising rewards. Nevertheless, developing an effective bug bounty program is challenging. As a consequence, many programs fail to support an efficient and fair bug bounty market. In this paper, we present *VeriOSS*, a novel bug bounty platform. The idea behind VeriOSS is to exploit the blockchain technology to develop a fair and efficient bug bounty market. To this aim, VeriOSS combines formal guarantees and economic incentives to ensure that the bug disclosure is both reliable and convenient for the market actors.

2012 ACM Subject Classification Security and privacy → Software security engineering; Software and its engineering → Formal software verification; Security and privacy → Economics of security and privacy

Keywords and phrases Bug Bounty, Decentralized platforms, Symbolic-reverse debugging

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2020.6

Funding This research is partially funded by IMT PAI project "VeriOSS".

1 Introduction

Free and open source software (FOSS) is becoming more and more popular.[1] Operating systems and applications that we use daily are often developed and maintained by consortia of partner industries and communities of developers. FOSS is even mandatory in some cases, e.g., cryptographic functions are publicly developed for transparency and revision.

Bug bounty programs are essential to attract skilled software analysts for the detection, disclosure and correction of software errors. In a bug bounty program, a *bounty issuer* (BI) offers a reward to any *bounty hunter* (BH) who discovers a bug in a piece of software. The offered reward usually depends on the typology and criticality of the bug. For instance, Google promises to pay up to 15000$ for a sandbox escape vulnerability in the Chrome web browser.[2]

Often, BI is the software developer or owner, e.g., Google in the example above. However, a bounty can be also issued for third-party software. This is the case for FOSS components

[1] `https://www.forbes.com/sites/taylorarmerding/2019/01/09/the-future-of-open-source-software-more-of-everything/`
[2] `https://www.google.com/about/appsecurity/chrome-rewards`

involved in some critical systems, either open or proprietary. A prominent example of bounties for third-parties software is the *Free and Open Source Software Audit* (FOSSA) project, sponsored by the European Commission and offering bounties of up to several hundreds of thousands of euros for vulnerabilities discovered in 14 major FOSS.[3] According to the project executives, FOSSA is a response to *Heartbleed*, a severe security vulnerability that affected OpenSSL in 2014.[4]

Bug bounty programs are subject to numerous challenges. The main one is BI's lack of commitment with respect to the eligibility of bugs. Usually, a BH is expected to disclose all details of a bug to the BI who decides on the severity of the bug and therefore how much to pay. Clearly, the BI has strong incentives to "downgrade" the bug or declare it not eligible for the bounty. In this way, the BI depresses the payment to the BH who, at that point, has no more bargaining power. For example, in 2016 the majority of the security report received by Google were considered invalid.[5] This makes the bounty market inefficient and pushes BHs to look for other opportunities, such as gray and black markets.[6]

As a partial answer to this problem, mediation platforms have been created, in an effort to obtain better terms for the BHs. For instance, HackerOne[7] and Integriti[8] support ethical hackers in submitting their reports and collecting rewards. A second answer is to transform a bug into an exploit, that is an attack leveraging it. This increases the bargaining power of the BH toward the BI, and in fact some platforms exclusively focus on exploits.[9]

In this position paper, we present the design and the underlying ideas of VeriOSS, a blockchain-based platform for bug bounties. Our goal is to increase the reward for BH, so to foster more bug hunting and, consequently, decrease the appeal of grey and black markets. To do that, VeriOSS drives both BI and BH though a bug disclosure protocol. The protocol starts from the BI issuing a bug reward contract where a precise characterization of the eligible bugs is provided together with the offered reward. When a BH claims the bounty, she must provide enough information for the BI to check the eligibility without revealing the details for reproducing the bug. If the BI accepts the transaction, a remote debugging protocol is executed between the BI and the BH. At each step, BI computes a challenge that BH can only solve by continuing the debug process and revealing part of the execution trace reproducing the bug. In exchange, BI provides a commitment to pay a fraction of the total reward through a smart contract. Eventually, BI and BH either complete the protocol or interrupt it. In both cases, since BH and BI negotiate the partial rewards at each step, the protocol ensures a fair trade between the revealed information and the reward.

The rest of the paper is organized as follows. The next section introduces some preliminary notions. We present the design of VeriOSS and of its main components in Section 3. Section 4 discusses the economic incentives that drive the protocol execution. We discuss the threat model, some limitations and future extensions in Section 5. Finally, Section 6 compares our proposal with the literature, and Section 7 draws some conclusions.

[3] `https://juliareda.eu/fossa`
[4] `http://heartbleed.com`
[5] `https://sites.google.com/site/bughunteruniversity/behind-the-scenes/charts/2016`
[6] The activities occurring on gray and black markets are hard to document. However, Hacking Team's recently hacked emails provide a glimpse on the workings of these markets. See `https://tsyrklevich.net/2015/07/22/hacking-team-0day-market/`.
[7] `https://www.hackerone.com`
[8] `https://www.intigriti.com`
[9] For instance, Zerodium `https://zerodium.com` that offers up to 2,000,000$ for a zero-day exploit.

2 Preliminary notions

2.1 Program semantics

A program s is a finite sequence of statements $c_1, \dots, c_k$. Statements can be of various types, e.g., assignments of values to variables or conditional branches. A computation is carried out through atomic steps. Each step has the effect of modifying the program state σ and to update the sequence of statements to be run. As usual in program semantics, the state is a finite mapping from variables (in the scope of the current statement) to values [18]. Thus, a *program configuration* is a pair $\langle \sigma, s \rangle$. A step is $\langle \sigma, s \rangle \to \langle \sigma', s' \rangle$ to denote that in the state σ the program s executes one of its statements, becomes s' and modifies the state in σ'. For brevity, we write $\langle \sigma, s \rangle \to^* \langle \sigma', s' \rangle$ as a shorthand for a finite sequence of steps $\langle \sigma, s \rangle \to \dots \to \langle \sigma', s' \rangle$. Moreover, when a computation terminates, i.e., the destination configuration contains an empty sequence of statements, we simply write the final state as $\langle \sigma, s \rangle \to^* \sigma'$. We refer to [18] for a general presentation on the formal semantics of programming languages.

2.2 Hoare logics

The goal of program verification is to prove that a program s complies with a given specification. The specification is often defined in terms of *preconditions* and *postconditions*. Intuitively, a precondition is a property P that is assumed to hold in the initial state (from which the computation of s starts) and a postcondition is a property Q that must be guaranteed to hold in the final state (assume-guarantee reasoning). In symbols, the problem is encoded as an *Hoare triple* $\{P\}s\{Q\}$. The triple is valid if $\forall \sigma, \sigma'. \; P(\sigma) \land \langle \sigma, s \rangle \to^* \sigma' \Rightarrow Q(\sigma')$. The proof system used for reasoning about the validity of Hoare triples is called *Hoare logics*. We write $\models \{P\}s\{Q\}$ when there exist a proof of validity.

3 VeriOSS

In this section we introduce the main components of VeriOSS and how they interact. Briefly, VeriOSS has two goals: (i) support the honest BH in collecting a reward under the assumption of an untrusted BI; and (ii) protect BI against untrusted BHs claiming an undeserved reward. In particular, VeriOSS achieves these two goals by (i) requiring BI to provide a precise description of the eligible bugs; and (ii) driving the BH disclosure and rewarding process.

3.1 Workflow overview

The general workflow of VeriOSS is depicted in Figure 1. Initially, BI publishes a bounty on the blockchain. The bounty contains information about the type of bugs BI is interested in and the reward. When BH detects a bug that complies with the issued bounty, she can claim the reward. To do so, BH sends the initial debug information, e.g., the instruction where the bug was detected. This initial disclosure should allow BI to check the eligibility of the bug. If BI agree to continue, a disclosure loop starts. At each iteration, BI synthesizes a challenge for BH to test her knowledge of the bug trace at a specific step. If BH solves the challenge, she receives a partial reward (expressed as a fraction of the total one) and she provides information to continue the disclosure loop. Eventually, the protocol terminates when either the bug is entirely disclosed (proof completed) or one of the participants withdraws.

Below we discuss the components of VeriOSS and their requirements.

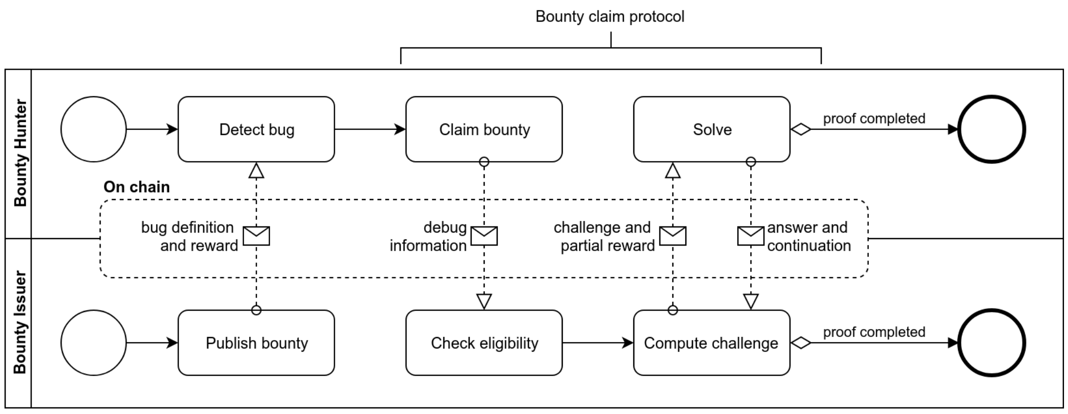


Figure 1 BPMN representation of the workflow.

```
float foo(unsigned char c) {
  int a = c+1;          //@ assert a != 0;
  float z = 255/a;      //@ assert z != 0;
  return 1.0/z;
}
```

Figure 2 A fragment of C code potentially dividing by zero.

3.2 Bug specification

When publishing a bounty, BI has to provide a rigorous description of the bugs that are eligible for the reward. Such a description contractualizes the commitment of BI to pay for a compatible bug. Some classification techniques exist to define bugs and vulnerabilities. For instance, the Common Vulnerability Scoring System[10] (CVSS) aims at describing a vulnerability and measuring its criticality. Also the Common Weaknesses Enumeration[11] (CWE) specification language is used to identify different vulnerability types. Since these approaches focus on describing the severity of vulnerabilities and exploits, they are not suitable for bug bounty programs. In fact, often the BI aims at disclosing bugs even without knowing their possible impact and severity. Moreover, since they have no formal semantics, they can hardly support an automatic validation process.

A more promising direction is to consider specification languages for contract-driven development [14]. These languages are used to define the properties of a piece of code in terms of preconditions (what must be true before the execution) and postconditions (what must be true after the execution). Moreover, they are usually provided with a formal semantics as well as tools for the automatic reasoning. Intuitively, program specifications can be adapted to define the conditions under which a bug shows up. The bug conditions can be expressed as assertions that the program violates during a bugged execution. To clarify, let us consider an example based on the ANSI C Specification Language (ACSL), used by the Frama-C framework [12].

▶ **Example 1.** Consider the C code of Figure 2. If we are interested in spotting out divisions by 0, there are two candidate instructions, i.e., the assignment to `z` and the `return` statement. In terms of properties to be satisfied, the preconditions for the two statements are $\texttt{a} \neq 0$ and

[10] `https://www.first.org/cvss/specification-document`
[11] `https://cwe.mitre.org/cwss/cwss_v1.0.1.html`

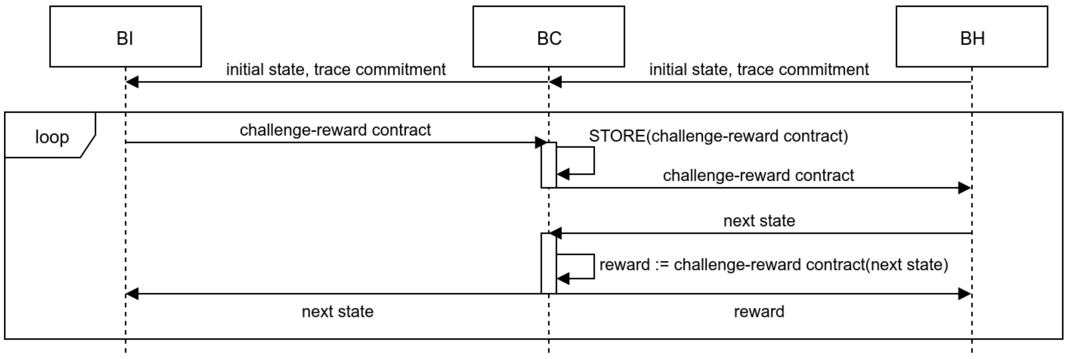

BI
BC
BH
initial state, trace commitment
initial state, trace commitment
loop
challenge-reward contract
STORE(challenge-reward contract)
challenge-reward contract
next state
reward := challenge-reward contract(next state)
next state
reward

Figure 3 The P2K protocol message sequence diagram.

$\texttt{z} \neq 0$, respectively. In ASCL the corresponding assertions are placed right before the target instructions as in Figure 2. Here, the bug is exposed (only) when `c = 255`. As a matter of fact, due to the integer division `255/256`, 0 is assigned to `z`, so violating the second assertion.

3.3 Challenge-response interaction

By definition, the bounty claim protocol is a Proof of Knowledge (PoK) protocol, also called $\Sigma-$protocol (see [10] for further details). A PoK consists of a prover and a verifier interacting through a challenge-response process.

However, our working conditions are slightly different. The reason is that both parties need to prove something: BH must prove she knows the bug and BI must prove she is willing to pay the reward. This is an instance of a *two-party fair exchange* protocol [15] that we call *Pay-per-Knowledge* (P2K).

The main difference between a standard PoK is that the two parties play both roles, i.e., prover and verifier. Their individual goal is to acquire the other's knowledge/reward. Also, the global goal of the protocol is that the two parties only achieve their individual goals *together*. Notice that "together" does not mean simultaneously. For instance, a party could receive the other's knowledge while providing an *effective commitment* to release her own knowledge (e.g., within a certain time).

Intuitively, a way to implement P2K is to rely on a trusted third party (TTP) that mediate and drive the interaction between the two participants. However, having a TTP is a restrictive assumptions. Smart contracts can support the same kind of operation. Indeed, a smart contract can carry out a certain task when a certain condition is satisfied, e.g., someone knows the answer to a challenge. We discuss this aspect in Section 3.6.

Figure 3 shows the P2K message flow of the bounty claim protocol. The bug disclosure is based on a remote debugging process (see Section 3.4) replicating the execution of a buggy program trace. The protocol starts with BH claiming the bounty by describing the bug without disclosing it. For instance, the bug description can consist of a buggy state reached by the program at the end of the execution trace. This initial disclosure allows the BI to check the eligibility and severity of the bug, without being able to replicate it nor verify its actual existence. Contextually, the BH commits the debug trace. The commitment amounts to the hash values of the program states appearing in the trace. The trace commitment ensures that a dishonest BH can neither craft a trace not diverge from the nominal protocol execution (see Section 3.3).

The challenge-response loop proceeds as follows. BI stores a *challenge-reward* smart contract on the blockchain. Briefly, the smart contract consists of a payment, i.e., a partial reward, activated when a certain input is provided. The contract input is the answer to the challenge computed by BI. In particular, the challenge is solved by a program state from which the buggy state is reachable (in a certain number of steps). BH checks the challenge and the amount. If she agrees with the partial reward, she submits the program state. If this program state correctly solves the challenge, and at the same time is consistent with the obfuscated trace, then the BH can collect the reward. Since the blockchain is public, BI retrieves the submitted state. The loop is repeated by replacing the buggy state with the next state provided by BH. Eventually, the loop terminates when BH provides an initial state of the program or one of the parties retires from the protocol. We describe the challenge generation procedure and the smart contract implementation in Sections 3.5 and 3.6, respectively.

3.4 Remote debugging

The challenge-response protocol described above implements a *remote debugging* process. Remote debugging occurs when the target program runs on a different location, e.g., a remote host. Under our assumptions, BH executes the target program[12] and BI debugs it.

Remote debugging is common and many debug tools support it. However, there is a crucial difference with the (standard, forward) remote debugging process: our debugging procedure proceeds backward. As a matter of fact, the debugging starts from a (buggy) final state and proceeds toward an initial state (*reverse debugging*).

In principle, reverse debugging does not prevent the early disclosure of the execution trace. In fact, in many cases the state of a program is deterministically determined by its predecessors. Hence, BI might infer the predecessor state without interacting with BH.

▶ **Example 2.** Consider again the code of Example 1 and the final state reached when `c = 255`. Such a state is $\sigma = [\texttt{z} \hookleftarrow 0, \texttt{a} \hookleftarrow \texttt{256}, \texttt{c} \hookleftarrow \texttt{255}]$. Trivially, since $\sigma(\texttt{c})$ is defined, the actual parameter of `foo`, i.e., the initial state, is exposed.

To address this issue BH only partially reveals the debug state: she only discloses the variables that are necessary to the current statement, i.e., those occurring in the expressions to be computed.

▶ **Example 3.** We simulate a reverse debug session starting from σ as in Example 2. The state σ refers to the statement `return 1.0/z` (where only the variable `z` appears). Thus, BH sends to BI the state $\sigma_z = [\texttt{z} \hookleftarrow 0]$. In this way, BI effectively verifies that the division by 0 occurs. Still, she cannot easily infer the values of `a` (that determines the value of `z`). As a matter of fact, any state where `a` is larger than 255 is a candidate predecessor. Assuming that each iteration correspond to a single debug step, the next state revealed by BH is $\sigma_a = [\texttt{a} \hookleftarrow \texttt{256}]$. The debug step succeeds when BI verifies that the execution of the current statement on state σ_a results in state σ_z.

3.5 Challenge generation

As stated above, the challenge is a boolean condition that drives a decision procedure encoded as a smart contract. In particular, given a program state σ, the challenge must precisely

[12] In principle, BH might even reply a recorded execution trace without executing the program. This is also called *Post-mortem* debugging.

characterize a state σ' being a valid predecessor of σ in the debug procedure. Moreover, to solve the challenge, both σ and σ' must belong to the execution trace initially committed by BH (see Section 3.3).

A prominent technique for this task is *backward symbolic execution* [16, 2]. Backward symbolic execution is used to obtain valid preconditions for the execution of a statement starting from its postconditions. This is typically achieved by means of a *weakest precondition* calculus [3]. Briefly, given a program s and a postcondition Q, a weakest precondition is the most general predicate P such that $\models \{P\}s\{Q\}$.

▶ **Example 4.** Consider again the ASCL code of Example 1. The predicate $\texttt{z} \neq \texttt{0}$ is a postcondition for the statement `float z = 255/a`. The weakest precondition for the statement is a predicate P such that $P \Rightarrow \texttt{z} \neq \texttt{0}$. Since $\texttt{z} = \texttt{255/a}$ this becomes $P \Rightarrow \texttt{255/a} \neq \texttt{0}$. Moreover, due to the semantics of the integer division operator in C this is equivalent to $P \Rightarrow \texttt{a} \leq \texttt{255}$. Clearly, the most general (weakest) predicate P that satisfies the implication is $\texttt{a} \leq \texttt{255}$.

To generate a challenge, BI can follow the strategy below. First, BI converts the current debug state to a predicate Q defined as $\bigwedge_{x \in Dom(\sigma)} x = \sigma(x)$. The predicate Q is the precondition to the current debug statement. Also, Q is the postcondition of all the previous statements, i.e., those to be debugged to reach the initial state of the execution. Hence, BI selects a number n of backward steps. From the code, BI extracts all the sequences of statements of length n that can precede the current statement. Via backward symbolic execution on the selected statements, BI computes the weakest preconditions for Q. The resulting predicate is the challenge for BH that she answers by providing the actual state that satisfies the precondition.

▶ **Example 5.** Consider the debug session given in Example 3. The final state σ_z results in the predicate $\texttt{z} = \texttt{0}$. Assuming $n = 1$, the challenge for BH is $\texttt{a} > \texttt{255}$ (trivially from Example 4). Then, BH successfully answers by providing σ_a.

It is evident from the example above that the choice of n is critical. In general, the size of a predicate computed through backward symbolic execution can grow exponentially with n [8]. Intuitively, the exponential blow-up is caused by the conditional statements.

In software verification, large predicates pose serious limitations. Indeed, *satisfiability modulo theories* (SMT) [6] is used to verify whether a certain predicate admits a model, i.e., an assignment of values that satisfy the predicate. The SMT problem is computationally hard, but its complexity varies with the underlying theory. In our context, bit-vectors are the most common theory. The SMT problem for bit-vectors is known to be (in the best case) NP-complete [13]. Nevertheless, this is not a limitation in our context as BH already knows a solution to the challenge, that is the program state that she has committed.

3.6 Smart contracts and blockchain

In this section we describe the structure of the smart contracts used by VeriOSS. There are two smart contracts, i.e., the bounty issuing contract and the partial reward contract. The first one is straightforward. Its role is to describe the bug and the offered reward. The second contract requires more attention. As a matter of fact, it is responsible for the partial rewarding defined in Section 3.3.

Figure 4 shows an example *Solidity* [5] contract for the challenge of Example 5, i.e., $\texttt{a} > \texttt{255}$. The contract handles three pieces of information (lines 2-4), i.e., the address of the bounty hunter, the amount of the reward and an expiration time. The main function of the

```
1  contract PartialReward {
2   address public hunter = /* ... */;
3   uint    public reward = /* ... */;
4   uint    public expire = /* ... */;
5
6   function challenge(bytes4[] state) public {
7    if(decommit(state) && solve(state))
8     hunter.transfer(reward);
9   }
10  function solve(bytes4[] state) private returns (bool) {
11   if(state[0] <= 255) /* a ≤ 255 */ return false;
12   return true;
13  }
14  function decommit(bytes4[] state) private returns (bool)
15  { /* check state hash */ }
16  function timeout() public { require(now >= expire);
17   selfdestruct(this); }
18 }
```

Figure 4 An instance of the partial reward smart contract.

contract is `challenge` (line 6). The hunter invokes the function by providing the program state as a list of bytes. Then, the contract invokes two functions, i.e., `decommit` and `solve` (line 7). The former (line 14) decommits the input state (i.e., it checks its hash code against the list initially provided by the BH). The latter verifies that the provided state is a valid solution to the challenge. If both the checks succeed, the contract transfers the reward to the hunter. The function `solve` (line 10) encodes the challenge. It consists of a sequence of conditional statements. Each statement checks whether a single clause of the challenge is violated. In that case, `solve` returns `false`. When all the checks are passed, the function returns `true`. Finally, the contract has a `timeout` function (line 16) to void it when the deadline expires.

Few aspects of the contract of Figure 4 need a further discussion. In the first place, the structure of function `solve`. Clearly, it is the most expensive function in terms of computation and, since on chain computation is not for free [19], efficiency might be an issue. As highlighted in Section 3.3, checking the solution to a challenge is linear in the number of constraints. However, this number grows exponentially with n. Thus, a proper trade-off must be considered.

4 Incentives

From the economic viewpoint, VeriOSS aims at allowing a profitable trading between a seller (BH) and a buyer (BI) of information (the bug). The protocol of Section 3 can accomplish this goal, but BH and BI may refuse to run it. The main reason is *hold-up* [1], i.e., the buyer can refuse to pay after she leaned the information. This could prevent potentially profitable exchanges due to stall between the seller (who wants to be paid before disclosing the information) and the buyer (who wants to evaluate the information before paying it).

VeriOSS overcomes this issue by delegating the verification of the information and the payment of the reward to a smart contract. By itself, however, this is not sufficient to give BI and BH the correct economic incentives to follow the protocol of Section 3. Below we list the incentives problem faced by BI and BH at every step of the protocol, and how VeriOSS addresses them.

1. Since BI puts forward the reward when publishing the initial bounty contract, the reward offered by BI might be inadequate for BH. However, we expect a round of communication between BI and BH to occur beforehand to ensure that BI and BH agree on the reward. Also, due to the guarantees of the P2K protocol, BI and BH can negotiate under the assumption that the counterpart is honest.
2. The cost of the off-chain computation of BI is not negligible. In particular, computing the weakest preconditions may be computationally hard. For this reason, it is crucial that the information initially disclosed by BH provides a proper incentive to set up the challenges. For instance, BH might need to initially reveal some extra details about the debug trace. What is the right amount of information is an open research question, e.g., see [11].
3. A malicious BI could intentionally craft an incorrect challenge. The main motivation here is inferring as much information as possible from BH's answer. For example, BI might submit an unsatisfiable challenge to make the protocol fail even if the provided answer is correct. In this way, BI may collect the next state without paying the partial reward. However, BH can also compute the weakest preconditions and detect an incorrect challenge. In such a case, she can retire from the protocol with no loss.
4. Even if BH has always answered correctly, BI could decide to interrupt the protocol before the end. For instance, BI may believe that the information still to be released by BH is worth than the remaining reward. This boils down to correctly establishing the partial rewards, so to adequately compensate BH while encouraging BI to continue. As long as they correctly price each iteration, BI is not motivated to interrupt the protocol.[13]
5. BH may attempt to renegotiate the reward after BI computes a challenge, i.e., BH can hold up BI. Indeed, since it is costly, BI may accept to pay an higher partial reward to avoid recomputing the challenge. Note, however, that the total reward is established at the beginning of the protocol. Hence, an honest BH would no obtain an higher total payment. Since it reveals that BH is malicious, no BH (malicious or honest) is motivated to renegotiate.

Finally, note that the above analysis assumes the presence of a single BH and a single BI. This is not the case in general. The presence of other BIs and BHs may affect the incentives faced by the protocol's participants, and hence the performance of the protocol. We discuss this issue in the next section.

5 Discussion

In this section we provide a detailed discussion on some aspects that may affect the implementation of VeriOSS, some open issues and future developments.

5.1 Implementation details

The implementation will need to address issues about some aspects we left abstract in the previous sections. A first issue concerns how to represent the commitment trace of BH. Intuitively, this trace can be obtained by computing the hashes of each state in the original debug trace. However, this may be impractical because of the length of the debug trace. Thus, we need an implementation that compresses these traces without compromising the validity of the protocol.

[13] This issue can also be more directly addressed by using an escrow (see Section 5.2).

Another issue is about the number n of iterations of the challenge-response protocol. This choice is quite critical: different choices of n may lead to different cost in term of (i) cryptocurrency paid by the parties and (ii) efficiency of the protocol. Finding a good trade-off is left as future work.

5.2 Threat model

The design of VeriOSS is based on a threat model where both BI and BH do not trust each other and both may be malicious. On the one hand, a malicious BI aims at collecting information about a bug without paying the corresponding reward. Our protocol opposes this behavior and forces BI to behave honestly by (i) requiring a precise specification of the eligible bugs (Section 3), (ii) increasing the bargaining power of the BH, (iii) providing the partial reward mechanism in which a small portions of the bounty is paid in each iteration for each piece of revealed information (Section 3.3).

On the other hand, a malicious BH aims at obtaining an undue reward. For instance, BH might submit a partial or a false bug trace during the remote debug protocol. Also, a malicious BH could attempt a *reply attack* by re-submitting an old, already paid trace. VeriOSS protects honest BIs against malicious BHs by (i) establishing a commitment phase (Section 3.3), (ii) providing a challenges-response protocol. In particular, the trace commitment ensures that past traces are automatically detected, e.g., because they terminate with the very same state of a previously executed trace. Instead, the challenge-response protocol ensures that each step of the trace is correct.

In addition, the protocol can be easily extended by introducing a second smart contract acting as an escrow that collects all the partial rewards and then forwards them to the BH only if the bug is entirely disclosed. In this way, a malicious BH cannot obtain any partial reward and, at the same time, a malicious BI cannot gain by strategically interrupting the protocol. As future work, we plan to further study the robustness of our mechanisms against this attacker model.

5.3 Future extension

Here, we outline some future directions for the development of VeriOSS.

The current design of VeriOSS allows a single BH and a single BI to efficiently exchange the bug trace against a reward. However, this is just an intermediate goal, because the bug bounty market consists of several actors. Currently, our bug disclosure process in exclusive between one BI and one BH. Instead, “open” sessions might allow other parties to interact, e.g., by offering a better reward.

The blockchain used by VeriOSS allows parties that do not know or trust each other to interact. Sometimes this is not desirable, e.g., when knowing the identity of the BI necessary to discriminate between legitimate companies and malicious actors. As a mitigating, we could allow the BI creating the smart contract to “sign” it using its private keys, therefore allowing everybody to verify that a given challenge was indeed created by a reputable BI. This, of course, would not prevent malicious actors from creating their own smart contracts using VeriOSS, but it would make public that a given (supposedly malicious) actor posted a challenge and obtained information regarding a bug. Discriminating between legitimate and malicious BIs is a future work.

Also, many different firms may benefit from discovering and fixing bugs in FOSS. This gives rise to what is known as “free rider” problem. The maximum payment a BH can receive depends on the willingness to pay for the bug report of the firm valuing it the most. Such a payment can be significantly lower than the overall benefit of finding the bug. VeriOSS can include a mechanism to aggregate rewards from several BIs. For instance, this can be achieved by introducing *reward rise contracts* that BIs can use to offer a further incentive toward the disclosure of a certain bug. Again, this is future work.

Finally, the presence of other actors is also relevant for the issue of "responsible disclosure". In most bug bounty programs, all parties are contractually forbidden from publicly disclosing the bug for a period of time. Such a time span may be legally imposed and it is intended to give the BI enough time to fix the bug. In VeriOSS, instead, the bug is immediately public, which implies that a malicious actor could exploit it before the BI manages to implement a remediation. This is also a direction where we plan to improve the protocol.

5.4 Limitations

Here, we briefly discuss some limitations of our proposal. In the design of VeriOSS we assume that BH has a copy of the software to test. This is not a problem for mobile apps, or desktop software that the BH can download from the network and run on her machines. However, when the target software is a web application or web service our remote debugging protocol cannot be applied as is. Indeed, in those situations the BH can mainly interact with the software by providing inputs and receiving outputs (a.k.a. black box testing). Hence, BH does not have access to the full program state, which is partially stored on a remote server.

Another limitation is the assumption that a bounty is only issued for a bug, i.e., a faulty state of the target program. Often, a BI only offers a reward for a bug that actually impacts on the security of the software. Said differently, a BI might ask for an exploit exposing her software to concrete attacks, e.g., data breaches. Thus, the offered reward depends on the value of the assets that an attacker can steal or compromise. At the moment VeriOSS does not support this kind of bounty programs. Furthermore, although formal specification languages can precisely characterize a failure condition, one could argue that some types of bugs cannot be expressed (easily or even at all). For instance, think about the remote code execution caused by a ROP chain [17]. For these reasons, we plan to introduce multiple languages for the specification of bugs and exploits. The main requirement for the bug definition languages is that they must provide a sound eligibility check (so that BI cannot repudiate an eligible bug) and support the challenge-response process.

6 Related work

VeriOSS is made of different components each based on a specific technology. Here, we follow the line of Section 3 and compare each component of VeriOSS with similar proposals.

6.1 Remote attestation

Remote attestation [4] allows a remote host (*the challenger*) to authenticate the hardware and software configuration of another remote host (*the attestator*) which is charge of performing some computation. The attestator is equipped with a suitable *Trusted Platform Module* (TPM) chip, which she uses to attest the states of its software components to the challenger. Typically, this verification is based on digital signatures, i.e., the challenger only verifies that the signatures sent by the attestator are as expected. This basic attestation mechanism can be used as a building block to check other security properties. For example, [9] proposes the implementation of a trusted virtual machine that not only allows running a program but also attesting to a remote entity that the running program satisfies a given set of security properties at run time.

At a first sight, one may think that the challenge-response interaction protocol of Section 3.3 may be implemented using remote attestation. However, this is not the case because mainly remote attestation only allows BI to avoid a malicious BH, but not vice versa. Furthermore, remote attestation requires that BH is equipped with a specific hardware, i.e., TPM chip, that increase the cost of entering the market. Our protocol, instead, provides a mechanism to protect both participants and does not require any specific hardware.

6.2 Remote debugging

Modern development environment allows debugging applications remotely. This is very useful when the development system is different than the production one. The underlying idea is that the debugger is installed on the production server and that it provides a network channel for interacting with the debbuged program. The programmer uses a client that completely abstract the interactions through the network. In this way, debugging a program remotely is almost the same as doing it locally. In particular, this means that the client can stepwise run the program and can inspect its memory.

There are at least two crucial differences between a standard remote debugging and the approach described in Section 3.4. The first is that in standard remote debugging there is only an agent interacting with the program that is the client (the server only makes available the state of the program to the client); then, the client and the server trust each other, or both are under the same administrative domain. Whereas in our setting, BI and BH are two different agents in the system that does not trust each other.

The second important difference is that the standard remote debugging proceeds forwards and the client can access the entire state of the execution. In our approach, instead, the debugging proceeds backward and the BI can access only a specific part of the state of the execution.

6.3 Information flow

Information flow control (IFC) is a mandatory access control mechanism that enforces some restrictions on a piece of data and on all data derived from it. It was introduced in [7] as mechanism to enforce *non-interference* across security levels. IFC is continuously enforced at every information exchange. The underlying idea is that each piece of information is associated with a policy (tags working as metadata) describing its level of secrecy and integrity. Moreover, also entities of a system are associated with a security level, describing the sensitivity of the data they are allowed to handle. The mechanism guarantees that entities with a lower security clearance cannot read/write up to information with higher security level. Symmetrically, it also ensures that entities with higher security clearance cannot write down by making a disclosure of information.

During our remote debugging process the BH should not reveal too much information about the execution state, so that a malicious BI cannot reconstruct all the execution state. To do that, our protocol prescribes that BH shares only a part of the state. To determine which part of the state to be disclosed, the BH should follows an approach based on IFC.

6.4 Secure multi-party computation

A multi-party computation occurs when two or more entities join together to compute a certain function f. More precisely, consider n parties $P_1, \ldots, P_n$, each with its own input x_i, which want to compute $f(x_1, \ldots, x_n)$. A *secure multi-party computation* [10] is a multi-party computation where each participant P_i aims to preserve the privacy of its input x_i.

Our challenge-response protocol fits this setting where the function f to compute consists in the challenge verification process. Indeed, the BI provides as input a pair q made of the challenge and of the commitment contract; whereas the BH provides the corresponding state σ. The function f then perform the relevant checks and return a pair q' containing the reward for BH and the computation state for the BI. However, differently from the case of the secure multi-party computation, the input of BH is private, whereas the one of BI is public (and indeed published on a blockchain).

6.5 Information sharing

The inefficiencies of bug bounty programs are commont to all markets for information and have been known at least since [1]. Several authors have studied mechanisms to resolve these inefficiencies. The most closely related work is [11], which proposes a protocol in which the seller of information sustains several tests. Every time a test is successfully completed, the buyer sends the seller a partial payment. In their baseline model, the tests are such that if the seller really has the piece of information, then she passes the test. If she does not, then she can complete the test with probability $p < 1$, where lower values of p correspond to more stringent (and hence more informative) tests. In the first round of the protocol, the seller reveals some information by sustaining a test for free. After observing the result of the test, the buyer updates his belief regarding whether the seller has the piece of information, and with it the expected benefit of learning it. The buyer then sends a payment to the seller who sustains another test, and so on. The information is thus revealed in stage (by sustaining each test) and to each revelation stage corresponds a partial payment.

Crucially, [11] assumes a total lack of commitment: the buyer is free to withhold the payment to the seller, even after the seller passes the test. In the equilibrium with information revelation the prospect of learning additional information motivates the buyer to follow the protocol. But many other equilibria exist, including some in which no information is ever revealed.

The part of VeriOSS that is most closely related to [11] is the initial exchange of information, in which the BH reveals the initial state. The reason is that, at this stage, the BI is under no obligation to set up the smart contract and start the protocol. This lack of commitment implies that the intuition in [11] applies here as well. However, once the smart contract is set up and the iteration of challenge/response begins, [11] ceases to be relevant. The reason is that the BI can commit to pay the BH if and only if the BH has the correct piece of information. The fact that information is revealed in stages (with corresponding partial payments) is done exclusively for practical reasons. As already discussed, the protocol could run with a single test and a single answer revealing the entire trace, but crafting such test is computationally very expensive. For this reason the information generation protocol is split in different stages.

7 Conclusion

In this paper we presented VeriOSS, a novel paradigm for the construction of bug bounty programs. VeriOSS-based programs provide concrete guarantees that a bounty hunter will receive her rewards without trusting the bounty issuer. Together with other relevant properties natively supported by the blockchain, we expect this to favor the flourishing of the bug bounty market.

References

1 Kenneth Joseph Arrow. Economic welfare and the allocation of resources for invention. In *Readings in industrial economics*, pages 219–236. Springer, 1972.

2 Roberto Baldoni, Emilio Coppa, Daniele Cono D'Elia, Camil Demetrescu, and Irene Finocchi. A survey of symbolic execution techniques. *ACM Comput. Surv.*, 51(3):50:1–50:39, May 2018.

3 Marcello M. Bonsangue and Joost N. Kok. The weakest precondition calculus: Recursion and duality. *Form. Asp. Comput.*, 6(1):788–800, November 1994.

4 George Coker, Joshua D. Guttman, Peter Loscocco, Amy L. Herzog, Jonathan K. Millen, Brian O'Hanlon, John D. Ramsdell, Ariel Segall, Justin Sheehy, and Brian T. Sniffen. Principles of remote attestation. *Int. J. Inf. Sec.*, 10(2):63–81, 2011.

5 Chris Dannen. *Introducing Ethereum and Solidity.* Apress, Berkely, CA, USA, 1st edition, 2017.

6 Leonardo De Moura and Nikolaj Bjørner. Satisfiability modulo theories: Introduction and applications. *Commun. ACM*, 54(9):69–77, September 2011.

7 Dorothy E. Denning. A lattice model of secure information flow. *Commun. ACM*, 19(5):236–243, May 1976.

8 Cormac Flanagan, Cormac Flanagan, and James B. Saxe. Avoiding exponential explosion: Generating compact verification conditions. In *Proceedings of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '01, pages 193–205. ACM, 2001.

9 Vivek Haldar, Deepak Chandra, and Michael Franz. Semantic remote attestation: A virtual machine directed approach to trusted computing. In *Proceedings of the 3rd Conference on Virtual Machine Research And Technology Symposium - Volume 3*, VM'04, pages 3–3. USENIX Association, 2004.

10 Carmit Hazay and Yehuda Lindell. *Efficient Secure Two-Party Protocols: Techniques and Constructions.* Springer-Verlag, Berlin, Heidelberg, 1st edition, 2010.

11 Johannes Hörner and Andrzej Skrzypacz. Selling information. *Journal of Political Economy*, 124(6):1515–1562, 2016.

12 Florent Kirchner, Nikolai Kosmatov, Virgile Prevosto, Julien Signoles, and Boris Yakobowski. Frama-C: A software analysis perspective. *Formal Aspects of Computing*, 27(3):573–609, May 2015.

13 Gergely Kovásznai, Andreas Fröhlich, and Armin Biere. On the complexity of fixed-size bit-vector logics with binary encoded bit-width. In Pascal Fontaine and Amit Goel, editors, *SMT 2012. 10th International Workshop on Satisfiability Modulo Theories*, volume 20 of *EPiC Series in Computing*, pages 44–56. EasyChair, 2013.

14 Bertrand Meyer. Contract-driven development. In *Proceedings of the 10th International Conference on Fundamental Approaches to Software Engineering*, FASE'07, pages 11–11, Berlin, Heidelberg, 2007. Springer-Verlag.

15 Aybek Mukhamedov, Steve Kremer, and Eike Ritter. Analysis of a multi-party fair exchange protocol and formal proof of correctness in the strand space model. In Andrew S. Patrick and Moti Yung, editors, *Financial Cryptography and Data Security*, pages 255–269, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

16 Suzette Person, Guowei Yang, Neha Rungta, and Sarfraz Khurshid. Directed incremental symbolic execution. In *Proceedings of the 32Nd ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '11, pages 504–515. ACM, 2011.

17 Ryan Roemer, Erik Buchanan, Hovav Shacham, and Stefan Savage. Return-oriented programming: Systems, languages, and applications. *ACM Trans. Inf. Syst. Secur.*, 15(1):2:1–2:34, March 2012.

18 Glynn Winskel. *The Formal Semantics of Programming Languages: An Introduction.* MIT Press, Cambridge, MA, USA, 1993.

19 Daniel Davis Wood. Ethereum: A Secure Decentralised Generalised Transaction Ledger, 2014. (White paper).

A Foundation for Ledger Structures

Chad Nester
Tallinn University of Technology, Estonia

Abstract

This paper introduces an approach to constructing ledger structures for cryptocurrency systems with basic category theory. Compositional theories of resource convertibility allow us to express the material history of virtual goods, and ownership is modelled by a free construction. Our notion of ownership admits an intuitive graphical representation through string diagrams for monoidal functors.

2012 ACM Subject Classification Theory of computation → Categorical semantics

Keywords and phrases String Diagrams, Category Theory, Blockchains

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2020.7

Funding *Chad Nester*: This research was supported by the ESF funded Estonian IT Academy research measure (project 2014-2020.4.05.19-0001).

1 Introduction

Modern cryptocurrency systems consist of two largely orthogonal parts: A consensus protocol, and the ledger structure it is used to maintain. While consensus protocols have received a lot of attention (see e.g. [10, 7]), the design space of the accompanying ledger structures is barely explored. The recent interest in smart contracts has led to the development of sophisticated ledger structures with complex behaviour (see e.g. [1, 13]). These efforts have been largely *ad hoc*, and the resulting ledger structures are difficult to reason about. This difficulty also manifests in the larger system, which has contributed to several unfortunate incidents involving blockchain technology [2].

A strong mathematical foundation for ledger structures would enable more rigorous development of sophisticated blockchain systems. Further, the ability to reason about the ledger at a high level of abstraction would facilitate analysis of system behaviour. This is important: users of the system must understand it in order to use it with confidence. The formalism we propose has an intuitive graphical representation, which would make this kind of rigorous operational understanding possible on a far wider scale that it would otherwise be.

Blockchain systems are largely concerned with recording the material history of virtual objects, with a particular focus on changes in ownership. The resource theoretic interpretation of string diagrams for symmetric monoidal categories gives a precise mathematical meaning to this sort of material history. Building on this, we consider string diagrams augmented with extra information concerning the ownership of resources. We give these diagrams a precise mathematical meaning in terms of strong monoidal functors, drawing heavily on the work of [9], where our augmented diagrams originated. We show that an augmented resource theory has the same categorical structure as the original, in the sense that the two corresponding categories are equivalent. Finally, we give a simple example of a ledger structure using our machinery.

2 Monoidal Categories as Resource Theories

We assume familiarity with some basic category theory, in particular with symmetric monoidal categories. A good reference is [8]. Throughout, we will write composition in diagrammatic order. That is, the composite of $f : X \to Y$ and $g : Y \to Z$ is written $fg : X \to Z$. We may also write $g \circ f : X \to Z$, but we will *never* write $gf : X \to Z$. We will make heavy use of string diagrams for monoidal categories (see e.g. [11]), which we read from top to bottom (for composition) and left to right (for the monoidal tensor). Our string diagrams for ownership are in fact the string diagrams for monoidal functors of [9].

2.1 Resource Theories

We begin by observing (after [4]) that a symmetric strict monoidal category can be interpreted as a theory of resource convertibility: Each object corresponds to collection of resources with $A \otimes B$ denoting the collection composed of both A and B and the unit I denoting the empty collection. Morphisms $f : A \to B$ are then understood as a way to convert the resources of A to those of B.

For example, consider the free symmetric strict monoidal category on the set

$$\{\texttt{bread}, \texttt{dough}, \texttt{water}, \texttt{flour}, \texttt{oven}\}$$

of atomic objects, subject to the following additional axioms:

$$\texttt{mix} : \texttt{water} \otimes \texttt{flour} \to \texttt{dough} \qquad\qquad \texttt{knead} : \texttt{dough} \to \texttt{dough}$$

$$\texttt{bake} : \texttt{dough} \otimes \texttt{oven} \to \texttt{bread} \otimes \texttt{oven}$$

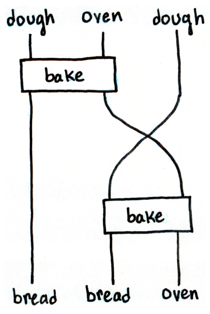
This category can be understood as a theory of resource convertibility for baking bread. The morphism `mix` represents the process of combining water and flour to form a bread dough, `knead` the process of kneading the dough, and `bake` the process of baking the dough in an oven to yield bread (and an oven). While this model has many failings as a theory of bread, it suffices to illustrate the idea. The axioms of a symmetric strict monoidal category provide a natural scaffolding for this theory to live in. For example, consider the morphism

$$(\texttt{bake} \otimes 1_{\texttt{dough}})(1_{\texttt{bread}} \otimes \sigma_{\texttt{oven},\texttt{dough}}\texttt{bake})$$

where $\sigma_{A,B} : A \otimes B \xrightarrow{\sim} B \otimes A$ is the braiding. This morphism has type

$$\texttt{dough} \otimes \texttt{oven} \otimes \texttt{dough} \to \texttt{bread} \otimes \texttt{bread} \otimes \texttt{oven}$$

and describes the transformation of two pieces of dough into two loaves of bread by baking them one after the other in an oven. We obtain a string diagram for this morphism by drawing our objects as wires, and our morphisms as boxes with inputs and outputs. Composition is represented by connecting output wires to input wires, and we represent the tensor product of two morphisms by placing them beside one another. Finally, the braiding is represented by crossing the involved wires. For the morphism in question, we obtain:



We will think of our ledger systems in terms of such string diagrams: The state of the system is a string diagram describing the *material history* of the resources involved, the available resources correspond to the output wires, and changes are effected by appending resource conversions to the bottom of the diagram. From now on we understand a *resource theory* to be a symmetric strict monoidal category with an implicit resource-theoretic interpretation.

2.2 How to Read Equality

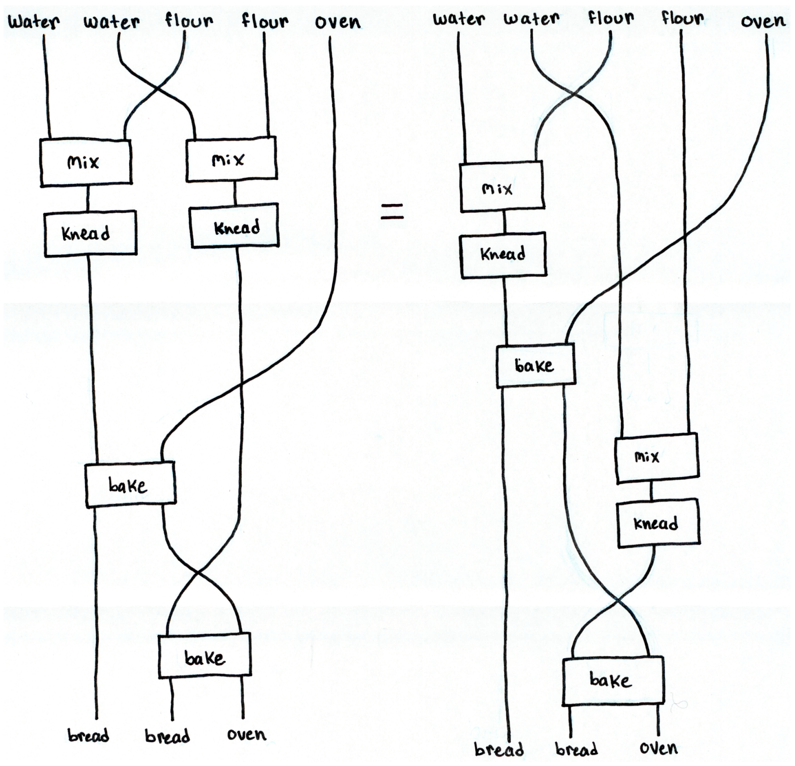
Suppose we have a resource theory $\mathbb{X}$, and two resource transformations $f, g : A \to B$. Each of f and g expresses a different way to transform an instance of resource A into an instance of resource B, but these may not have the same effect. For example, consider $\texttt{knead} : \texttt{dough} \to \texttt{dough}$ and $1_{\texttt{dough}} : \texttt{dough} \to \texttt{dough}$ from our resource theory of bread. Clearly these should not have the same effect on the input dough. This is reflected in our resource theory in the sense that they are not made equal by its axioms. For contrast, we can imagine a (somewhat) reasonable model of baking bread in which there is no difference between kneading the dough once and kneading it many times. We could capture this in our resource theory of baking bread by imposing the equation

$$\texttt{knead} = \texttt{knead} \circ \texttt{knead}$$

In this new resource theory, our equation tells us that kneading dough once has the same effect as kneading it twice, or three times, and so on, since the corresponding morphisms of the resource theory are made equal by its axioms. Of course, the material history described by $\texttt{knead} \circ \texttt{knead}$ is not identical to that described by $\texttt{knead}$. In the former case, the kneading process has been carried out twice in sequence, while in the latter case it has only been carried out once. That these morphisms are equal merely means that the effect of each sequence of events on the dough involved is the same.

We adopt the following general principle in our design and understanding of resource theories: *Two transformations should be equal precisely when they have the same effect on the resources involved.*

We further illustrate this by observing that, by the axioms of a symmetric monoidal category (specifically, by naturality of braiding), the following two transformations in the resource theory of baking (expressed as string diagrams) are equal. The transformation on the left describes baking two loaves of bread by first mixing and kneading two batches of dough before baking them in sequence, while the transformation on the right describes baking two loaves of bread by mixing, kneading, and baking the first batch of dough, and *then* mixing, kneading, and baking the second batch. Thus, according to our resource theory the two procedures will yield the same result – not an entirely unreasonable conclusion!



3 String Diagrams for Ownership

Ledgers used by blockchain systems are largely concerned with *ownership*. For example, in the Bitcoin system, each coin is associated with a computable function called the *validator*, which is used to control access to it. Anyone who wishes to use the coin must supply input data, called a *redeemer*, and the system only allows them to use the coin in question in case running the validator on the redeemer terminates in a fixed amount of time. If the validator is defined only on the data that results from `Alice` digitally signing a nonce generated by the system, then that coin can only be used by `Alice`, who then effectively owns it.

Different use cases call for different authentication schemes. For example, a proposed application of blockchain technology is to improve supply chain accountability by requiring participants to log any transfers and transformations of material on a public ledger (see e.g. [5, 12]). Here ownership implies responsibility, and so for Alice to log the transfer of, say, a ton of steel to Bob, *both* Alice and Bob must ratify the transfer via digital signature.

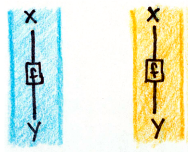
What different use cases have in common is that the resources of the ledger system are associated with ownership data. We leave the interpretation of this ownership data, including the specific details of the authentication scheme unspecified, instead giving a structural account of resource ownership. We develop our account of resource ownership intuitively, and somewhat informally, by introducing addtional features to string diagrams. This is made fully formal in the next section.

3.1 Ownership and Collection Management

Begin by assuming a theory of resources $\mathbb{X}$, and a collection $\mathcal{C}$ of potential resource owners, each of which we associate with a colour for use in our diagrams. Suppose for the remainder that `Alice`, `Bob`, and `Carol` range over $\mathcal{C}$, and are associated with colours as follows:

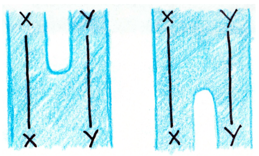
Our goal will be to construct a new theory of resources in which resources and transformations are associated with (owned and carried out by) elements of $\mathcal{C}$. The objects of our new resource theory will be collections of owned objects of $\mathbb{X}$. That is, for each object X of $\mathbb{X}$ and each $\texttt{Alice} \in \mathcal{C}$ we have an object $X^{\texttt{Alice}}$, which we interpret as an instance of resource X owned by $\texttt{Alice}$, along with the empty collection I and composite collections $X^{\texttt{Alice}} \otimes Y^{\texttt{Bob}}$, in which $\texttt{Alice}$'s instance of X exists alongside an instance of Y owned by $\texttt{Bob}$.

Similarly, for each transformation $f : X \to Y$ in $\mathbb{X}$, we ask for transformations $f^{\texttt{Alice}} : X^{\texttt{Alice}} \to Y^{\texttt{Alice}}$ and $f^{\texttt{Bob}} : X^{\texttt{Bob}} \to Y^{\texttt{Bob}}$ for all $\texttt{Alice}, \texttt{Bob} \in \mathcal{C}$, whose presence we interpret as the ability of each owner to effect all possible transformations of resources they own. We draw these annotated transformations as, respectively:



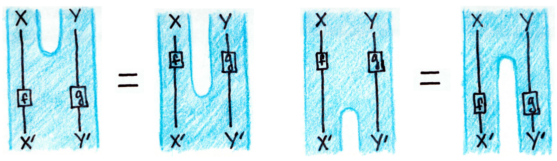
Since we are building a theory of resources we must end up with a symmetric monoidal category, so we also assume the presence of the associated morphisms, such as $f^{\texttt{Alice}} \otimes g^{\texttt{Bob}}$ and $\sigma_{X^{\texttt{Alice}},Y^{\texttt{Bob}}}$.

Next, we account for the formal difference between $X^{\texttt{Alice}} \otimes Y^{\texttt{Alice}}$ and $(X \otimes Y)^{\texttt{Alice}}$. In both situations $\texttt{Alice}$ owns an X and a Y, but in the former they are formally grouped together, while in the latter they are formally separated. We understand this formal grouping of $\texttt{Alice}$'s assets by analogy with physical currency. The situation in which $\texttt{Alice}$'s assets are separated is like $\texttt{Alice}$ having two coins worth one euro, while the situation in which they are grouped together is like $\texttt{Alice}$ having one coin worth two euros. In both cases, $\texttt{Alice}$ posesses two euros, but the difference is important: $\texttt{Alice}$ cannot give $\texttt{Bob}$ half of the two euro coin, but can easily give $\texttt{Bob}$ one of the two one euro coins. This distinction is also present in cryptocurrency systems, where there is an operational difference between having funds spread across many addresses and having them collected at one address. Reflecting both the reality of such systems and the principle that one ought to be able to freely reconfigure the formal grouping of things that they own, we ask that for each X, Y objects of $\mathbb{X}$ and each $\texttt{Alice} \in \mathcal{C}$ our new resource theory has morphisms $\phi_{X,Y} : X^{\texttt{Alice}} \otimes Y^{\texttt{Alice}} \to (X \otimes Y)^{\texttt{Alice}}$ and $\psi_{X,Y} : (X \otimes Y)^{\texttt{Alice}} \to X^{\texttt{Alice}} \otimes Y^{\texttt{Alice}}$. We draw these morphisms, respectively, as follows:



These changes of formal grouping should not interact with the resource transformations of our original theory $\mathbb{X}$, since it ought not matter whether $\texttt{Alice}$ combines (splits) her resources before or after transforming them. That is, we we require:

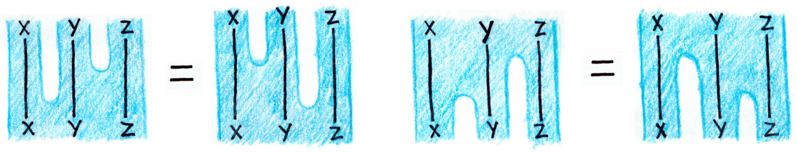
[G.1] $\phi^{\texttt{Alice}}_{X,Y}(f \otimes g)^{\texttt{Alice}} = (f^{\texttt{Alice}} \otimes g^{\texttt{Alice}})\phi^{\texttt{Alice}}_{X',Y'}$
[G.2] $(f \otimes g)^{\texttt{Alice}}\psi^{\texttt{Alice}}_{X',Y'} = \psi^{\texttt{Alice}}_{X,Y}(f^{\texttt{Alice}} \otimes g^{\texttt{Alice}})$



As it stands, there are many non-equal ways for `Alice` to reconfigure the formal grouping of their assets. Since these should all have the same effect, we need them all to be equal as morphisms in our resource theory. It suffices to ask that the $\phi^{\texttt{Alice}}$ and $\psi^{\texttt{Alice}}$ maps give, respectively, associative and coassociative operations, and that they are mutually inverse. That is (associativity and coassociativity):

[G.3] $(\phi^{\texttt{Alice}}_{X,Y} \otimes 1^{\texttt{Alice}}_{Z})\phi^{\texttt{Alice}}_{X\otimes Y,Z} = (1^{\texttt{Alice}}_{X} \otimes \phi^{\texttt{Alice}}_{Y,Z})\phi^{\texttt{Alice}}_{X,Y\otimes Z}$

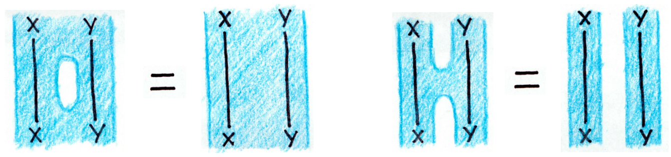
[G.4] $\psi^{\texttt{Alice}}_{X\otimes Y,Z}(\psi^{\texttt{Alice}}_{X,Y} \otimes 1^{\texttt{Alice}}_{Z}) = \psi^{\texttt{Alice}}_{X,Y\otimes Z}(1^{\texttt{Alice}}_{X} \otimes \psi^{\texttt{Alice}}_{Y,Z})$



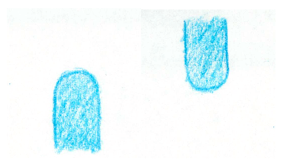
and (mutually inverse):

[G.5] $\psi^{\texttt{Alice}}_{X,Y}\phi^{\texttt{Alice}}_{X,Y} = 1^{\texttt{Alice}}_{X\otimes Y}$

[G.6] $\phi^{\texttt{Alice}}_{X,Y}\psi^{\texttt{Alice}}_{X,Y} = 1^{\texttt{Alice}}_{X} \otimes 1^{\texttt{Alice}}_{Y}$



To complete our treatment of these formal resource groupings, we must deal with the empty case $I^{\texttt{Alice}}$. We insist that `Alice` may freely create and destroy such empty collections via morphisms $\phi^{\texttt{Alice}}_{I} : I \to I^{\texttt{Alice}}$ and $\psi^{\texttt{Alice}}_{I} : I^{\texttt{Alice}} \to I$:



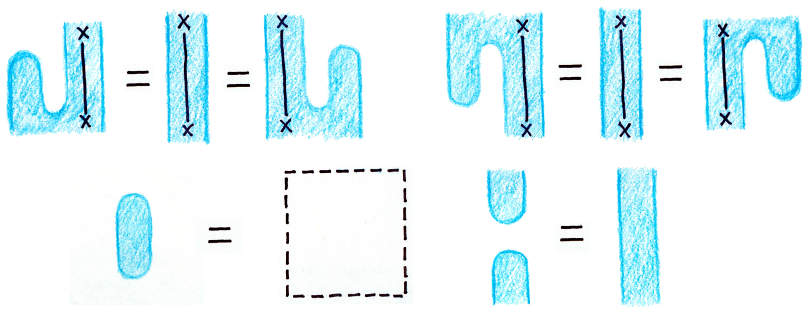
subject to the following axioms, which state that adding or removing nothing from a group or resources has the same effect as doing nothing, and that ϕ_I and ψ_I are mutually inverse, which together ensure that even with ϕ_I and ψ_I in the mix, any two formal regroupings with the same domain and codomain are equal.

[G.7] $(\phi^{\texttt{Alice}}_{I} \otimes 1^{\texttt{Alice}}_{X})\phi^{\texttt{Alice}}_{I,X} = 1^{\texttt{Alice}}_{X} = (1^{\texttt{Alice}}_{X} \otimes \phi^{\texttt{Alice}}_{I})\phi^{\texttt{Alice}}_{X,I}$

[G.8] $\psi^{\texttt{Alice}}_{I,X}(\psi^{\texttt{Alice}}_{I} \otimes 1^{\texttt{Alice}}_{X}) = 1^{\texttt{Alice}}_{X} = \psi^{\texttt{Alice}}_{X,I}(1^{\texttt{Alice}}_{X} \otimes \psi^{\texttt{Alice}}_{I})$

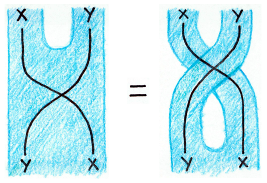
[G.9] $\phi^{\texttt{Alice}}_{I}\psi^{\texttt{Alice}}_{I} = 1_I$

[G.10] $\psi^{\texttt{Alice}}_{I}\phi^{\texttt{Alice}}_{I} = 1^{\texttt{Alice}}_{I}$



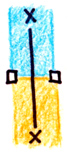
Finally, we ask that ϕ and ψ behave coherently with respect to the symmetry maps. It suffices to require that

[G.11] $\phi^{\texttt{Alice}}_{X,Y}\sigma^{\texttt{Alice}}_{X,Y} = \sigma_{X^{\texttt{Alice}},Y^{\texttt{Alice}}}\phi^{\texttt{Alice}}_{Y,X}$



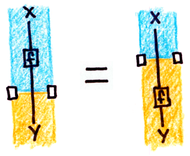
3.2 Change of Ownership

Of course, ownership is not static over time. We require the ability the *change* the owner of a given collection of resources. To this end we add morphisms $\gamma^{\texttt{Alice},\texttt{Bob}}_X : X^{\texttt{Alice}} \to X^{\texttt{Bob}}$ to our new resource theory for each object X of $\mathbb{X}$, each $\texttt{Alice},\texttt{Bob} \in \mathcal{C}$. We depict these new morphisms in our string diagrams as follows:



As with regrouping, change of ownership should not interact with resource transformations, in the sense that:

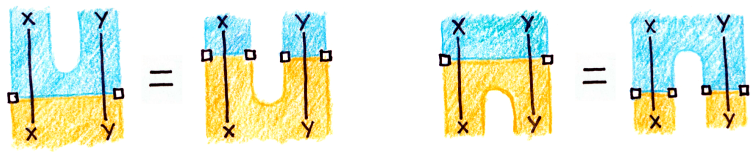
[O.1] $f^{\texttt{Alice}}\gamma^{\texttt{Alice},\texttt{Bob}}_Y = \gamma^{\texttt{Alice},\texttt{Bob}}_X f^{\texttt{Bob}}$



Further, change of ownership must behave coherently with respect to the regrouping morphisms in the sense that:

[O.2] $\phi^{\texttt{Alice}}_{X,Y}\gamma^{\texttt{Alice},\texttt{Bob}}_{X\otimes Y} = (\gamma^{\texttt{Alice},\texttt{Bob}}_X \otimes \gamma^{\texttt{Alice},\texttt{Bob}}_Y)\phi^{\texttt{Bob}}_{X,Y}$

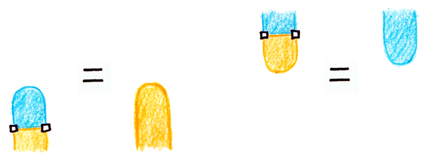
[O.3] $\gamma^{\texttt{Alice},\texttt{Bob}}_{X\otimes Y}\psi^{\texttt{Bob}}_{X,Y} = \psi^{\texttt{Alice}}_{X,Y}(\gamma^{\texttt{Alice},\texttt{Bob}}_X \otimes \gamma^{\texttt{Alice},\texttt{Bob}}_Y)$



For completeness, we axiomatize the interaction of change of ownership with empty collections by requiring that:

[**O.4**] $\phi_I^{\texttt{Alice}}\gamma_I^{\texttt{Alice,Bob}} = \phi_I^{\texttt{Bob}}$

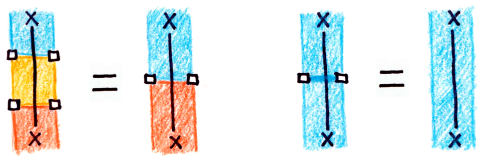
[**O.5**] $\gamma_I^{\texttt{Alice,Bob}}\psi_I^{\texttt{Bob}} = \psi_I^{\texttt{Alice}}$



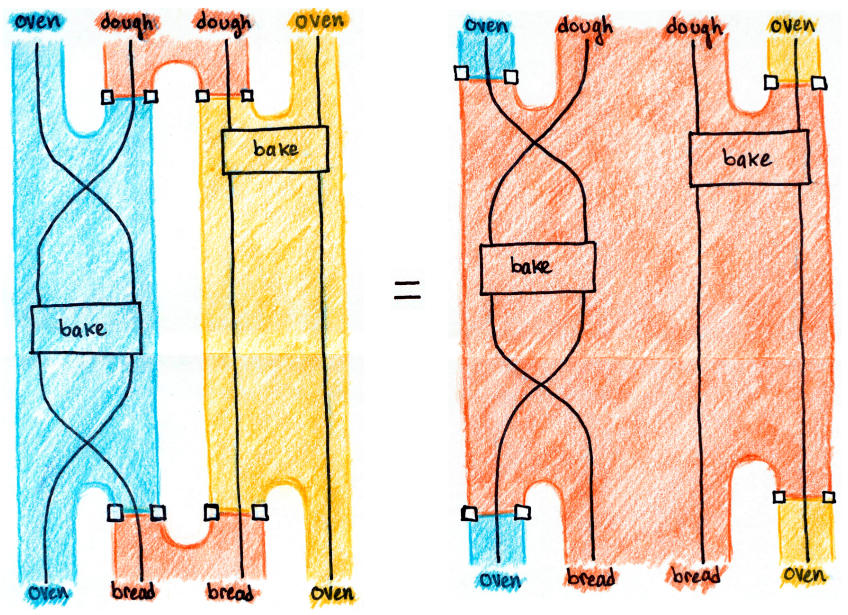
Finally, we insist that if `Alice` gives something to `Bob`, and `Bob` then gives it to `Carol`, this has the same effect as `Alice` giving the thing directly to `Carol`. Similarly, if `Alice` gives something to `Alice`, we insist that this has no effect.

[**O.6**] $\gamma_X^{\texttt{Alice,Bob}}\gamma_X^{\texttt{Bob,Carol}} = \gamma_X^{\texttt{Alice,Carol}}$

[**O.7**] $\gamma_X^{\texttt{Alice,Alice}} = 1_X^{\texttt{Alice}}$



We end up with a rather expressive diagrammatic language. For example, if we begin with the resource theory of bread, then our new resource theory is powerful enough to show:



which captures the fact that the sequence of events on the left in which `Carol` gives `Alice` and `Bob` each a portion of dough to bake in their ovens, after which they give the resulting bread to `Carol` *has the same effect* as the sequence of events on the right in which `Alice` and `Bob` give their ovens to `Carol`, who bakes the portions of dough herself before returning the ovens to their original owners. Notice that our diagrammatic representation of this is *much* easier to understand than the corresponding terms in linear syntax!

4 Categorical Semantics

In this section we show how our augmented string diagrams can be given precise mathematical meaning. Specifically, from a resource theory and a set whose elements we think of as entities capable of owning resources, we construct a new resource theory in which all resources are owned by some entity. We finish by showing how to model a simple cyrptocurrency ledger with our machinery.

4.1 Interpreting String Diagrams with Ownership

If $\mathbb{X}$ is a theory of resources and $\mathcal{C}$ is our set, we treat $\mathcal{C}$ as the corresponding discrete category, writing $A : A \to A$ for the identity maps, and form the product category $\mathbb{X} \times \mathcal{C}$. Write objects and maps of this product category as $X^A = (X, A)$ and $f^A = (f, A)$ respectively. Now, define $\mathcal{C}(\mathbb{X})$ to be the free strict symmetric monoidal category on $\mathbb{X} \times \mathcal{C}$ subject to the following additional axioms:

$$\frac{A \in \mathcal{C} \qquad X, Y \text{ objects of } \mathbb{X}}{\phi^A_{X,Y} : X^A \otimes Y^A \to (X \otimes Y)^A \text{ in } \mathcal{C}(\mathbb{X})} \qquad \frac{A \in \mathcal{C}}{\phi^A_I : I \to I^A \text{ in } \mathcal{C}(\mathbb{X})}$$

$$\frac{A \in \mathcal{C} \qquad X, Y \text{ objects of } \mathbb{X}}{\psi^A_{X,Y} : (X \otimes Y)^A \to X^A \otimes Y^A \text{ in } \mathcal{C}(\mathbb{X})} \qquad \frac{A \in \mathcal{C}}{\psi^A_I : I^A \to I \text{ in } \mathcal{C}(\mathbb{X})}$$

$$\frac{A, B \in \mathcal{C} \qquad X \text{ an object of } \mathbb{X}}{\gamma^{A,B}_X : X^A \to X^B \text{ in } \mathcal{C}(\mathbb{X})}$$

and subject to equations [**G.1**–**11**] and [**O.1**–**7**] for `Alice`, `Bob`, `Carol` $\in \mathcal{C}$, X, Y, Z objects of $\mathbb{X}$, and f, g morphisms of $\mathbb{X}$.

Clearly, $\mathcal{C}(\mathbb{X})$ is the new resource theory our coloured string diagrams live in. We think of objects X^A and morphisms f^A as being owned and carried out, respectively, by $A \in \mathcal{C}$. The free monoidal structue gives us the ability to compose such transformations sequentially and in parallel, and the additional axioms ensure our ownership interpretation of $\mathcal{C}(\mathbb{X})$ is reasonable.

We can characterize the category-theoretic effect of axioms [**G.1**–**11**] and [**O.1**–**5**] as follows:

▶ **Proposition 1.** *For any symmetric monoidal category $\mathbb{X}$ and any set $\mathcal{C}$, there is a strong symmetric monoidal functor*

$$A : \mathbb{X} \to \mathcal{C}(\mathbb{X})$$

for each $A \in \mathcal{C}$. Further, there is a monoidal and comonoidal natural transformation

$$\gamma^{A,B} : A \to B$$

between the functors corresponding to any two $A, B \in \mathcal{C}$.

Proof. Define $A : \mathbb{X} \to \mathcal{C}(\mathbb{X})$ by $A(X) = (X, A)$ on objects, and $A(f) = (f, A)$ on maps. For identity maps, $A(1_X) = (1_X, A) = 1_{(X,A)} = 1_{A(X)}$ since $(1_X, A)$ is the identity on (X, A) in $\mathbb{X} \times \mathcal{C}$. For composition, $A(fg) = (fg, A) = (f, A)(g, A) = A(f)A(g)$.Thus A defines a functor. A is strong symmetric monoidal via the ϕ^A and ψ^A maps together with [**G.1**] through [**G.11**]. Consider $A, B : \mathbb{X} \to \mathcal{C}(\mathbb{X})$ corresponding to $A, B \in \mathcal{C}$. Define $\gamma^{A,B} : A \to B$ to have components $\gamma^{A,B}_X$. Then $\gamma^{A,B}$ is a monoidal and comonoidal via [**O.1**] through [**O.5**]. ◀

Notice that we did not use [**O.6**–**7**] above. These axioms are motivated by our desire to model resource ownership, but they have an important, if subtle, effect on the theory: they allow us to show that $\mathbb{X}$ and $\mathcal{C}(\mathbb{X})$ are equivalent as categories. This means that any suitably categorical structure is present in $\mathbb{X}$ if and only if it is present in $\mathcal{C}(\mathbb{X})$ as well. For example, products in $\mathbb{X}$ manifest as products in $\mathcal{C}(\mathbb{X})$, morphisms that are monic in $\mathbb{X}$ remain monic in $\mathcal{C}(\mathbb{X})$, and so on. We may be confident that our addition of ownership information has not broken any of the structure of $\mathbb{X}$, or added anything superfluous!

▶ **Proposition 2.** *There is an adjoint equivalence between $\mathbb{X}$ and $\mathcal{C}(\mathbb{X})$ for each functor corresponding to some $A \in \mathcal{C}$.*

Proof. We show that each $A : \mathbb{X} \to \mathcal{C}(\mathbb{X})$ is fully faithful, and essentially surjective, beginning with the latter. To that end, suppose that P is an object of $\mathcal{C}(\mathbb{X})$. We proceed by structural induction: If P is I, then ϕ_0 witnesses $I \simeq A(I)$. If P is an atom (X, A), then $(X, A) = A(X)$. If P is $Q \otimes R$ for some Q, R, then by induction we have that $Q \simeq A(X_1)$ and $R \simeq A(X_2)$ for some objects X_1, X_2 of $\mathbb{X}$. We may now form

$$Q \otimes R \simeq A(X_1) \otimes A(X_2) \xrightarrow{\phi^A_{X_1,X_2}} A(X_1 \otimes X_2)$$

which witnesses $P \simeq A(X_1 \otimes X_2)$. Thus, A is essentially surjective. To see that A is fully faithful, let $U : \mathcal{C}(\mathbb{X}) \to \mathbb{X}$ be the obvious forgetful functor. The required bijection $\mathbb{X}(X, Y) \simeq \mathcal{C}(\mathbb{X})(A(X), A(Y))$ is given by A in one direction and U in the other. It sufffices to show that any morphism $h : A(X) \to A(Y)$ with $U(h) = f$ is such that $h = A(f)$. Notice that since each $\gamma^{A,B}$ is a monoidal and comonoidal natural transformation, there is a term equal to h in which all γ morphisms occur before all other morphisms (in the sense that f occurs before g in fg). Since $h : A(X) \to A(Y)$ we know that in this equal term the composite of the γ must have type $A(X) \to A(X)$, and must therefore be the identity by repeated application of [**O.6**] and [**O.7**]. This gives a term h' containing no γ maps with $h' = h$. Similarly, since the various ϕ and ψ morphisms are natural transformations, we may construct a term h'' by collecting all instances of ϕ and ψ terms at the beginning of h'. Once collected there, the composite of all the ϕ and ψ must have type $A(X) \to A(X)$, and is therefore equal to the identity. At this point we know that $h'' : A(X) \to A(Y)$ is such that $h'' = A(f_1) \cdots A(f_n)$ for some $f_1, \ldots, f_n$ in $\mathbb{X}$. By assumption $f = U(h) = U(h'') = f_1 \cdots f_n$, and therefore $h'' = A(f)$. ◀

4.2 A Simple Example

In this section we attempt to demonstrate the relevance of the above techniques to the cryptocurrency world by building a resource theory that models a simple ledger structure along the lines of Bitcoin [10]. Let $\mathbb{1}$ be the trivial category, with one object, 1, and one morphism, the identity 1_1. Define $\mathbb{N}$ to be the free symmetric strict monoidal category on $\mathbb{1}$, write 0 for the monoidal unit of $\mathbb{N}$, and n for the n-fold tensor product of 1 with itself for all natural numbers $n \geq 1$. Notice that $n + m$ is $n \otimes m$. We will think of the objects n of $\mathbb{N}$ where $n \geq 1$ as *coins.* Of course, $0 = I$ represents the situation in which no coin in present.

Define $\mathbb{N}_\nu$ to be the result of formally adding a morphism $\nu : 0 \to 1$ to $\mathbb{N}$, write $\nu_0 = 1_0 : 0 \to 0$, and $\nu_n : 0 \to n$ for the n-fold tensor product of ν with itself for $n \geq 1$. These morphisms confer the ability to create new coins, so we imagine their use would be restricted in practice. We will not ask for the ability to destroy coins, although there would be no theoretical obstacle to doing so.

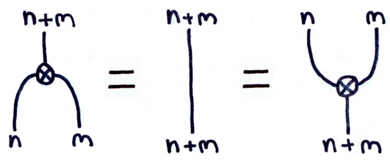
Now, let $\mathcal{C}$ be a collection of colours, which we can think of as standing in for cryptographic key pairs, or simply entities capable of owning coins. Consider $\mathcal{C}(\mathbb{N}_\nu)$. Objects are lists $n_1^{c_1} \otimes \cdots \otimes n_k^{c_k}$, which we interpret as lists of coins, where $n_i^{c_i}$ is a coin of value n_i belonging to

$c_i \in \mathcal{C}$. The morphisms are either ν_n^c for some $c \in \mathcal{C}$, the structural morphisms of a monoidal category, or the ϕ, ψ, and γ morphisms added by our construction. For $n, m \in \mathbb{N}$ and $\texttt{Alice}, \texttt{Bob} \in \mathcal{C}$, the maps $\phi_{n,m}^{\texttt{Alice}} : n^{\texttt{Alice}} \otimes m^{\texttt{Alice}} \to (n+m)^{\texttt{Alice}}$ and $\psi_{n,m}^{\texttt{Alice}} : (n+m)^{\texttt{Alice}} \to n^{\texttt{Alice}} \otimes m^{\texttt{Alice}}$ allow users to combine and split their coins in a value-preserving manner, and the $\gamma_n^{\texttt{Alice},\texttt{Bob}}$ maps allow them to exchange coins.

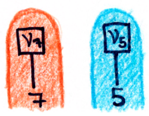
Now, a ledger is a (syntactic) morphism $a : I \to A$ of $\mathcal{C}(\mathbb{N}_\nu)$. A transaction to be included in a consists of a transformation $f : X \to Y$ of $\mathcal{C}(\mathbb{N}_\nu)$ along with information about which outputs of a are to be the inputs of the transformation, which we package as $t = \pi_t(1 \otimes f \otimes 1) : A \to B$. The result of including transaction t in ledger a is then the composite ledger $t \circ a : I \to B$. Put another way, a ledger is given by a list of transformations in $\mathcal{C}(\mathbb{N}_\nu)$:

$$I \xrightarrow{t_1} A_1 \xrightarrow{t_2} \cdots \xrightarrow{t_k} A_k$$

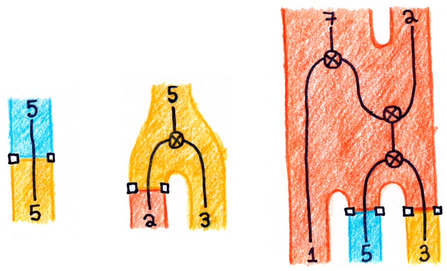
For the purpose of illustration, we differentiate between $m + n$ and $m \otimes n$ in our string diagrams for $\mathbb{N}_\nu$. We do so by means of the string diagrams for (not necessarily strict) monoidal categories (see e.g. [3]), as in:



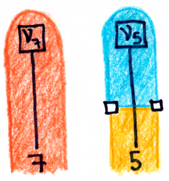
Now, suppose we have a ledger $a : I \to \nu_7^{\texttt{Carol}} \otimes \nu_5^{\texttt{Alice}}$:



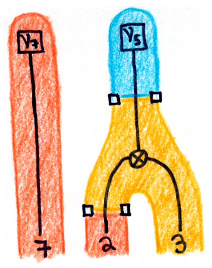
and resource transformations f_1, f_2, f_3 defined, respectively, by:



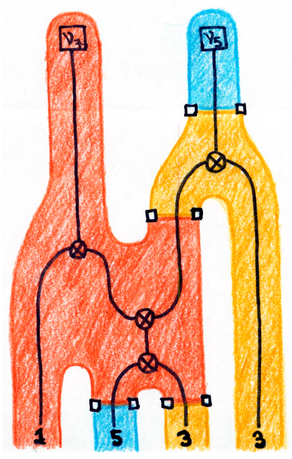
Now, form transaction $t_1 = (1_7^{\texttt{Carol}} \otimes f_1)$ and append it to a to obtain $t_1 \circ a$



Next, form transaction $t_2 = (1_7^{\texttt{Carol}} \otimes f_2)$ and append it to obtain $t_2 \circ t_1 \circ a$



Finally, form transaction $t_3 = (f_3 \otimes 1_3^{\mathsf{Bob}})$ and append it to obtain $t_3 \circ t_2 \circ t_1 \circ a$



In this manner, we capture the evolution of the ledger over time. Of course, we can also reason about whether two sequences of transactions result in the same ledger state by comparing the corresponding morphisms for equality, although in the case of $\mathcal{C}(\mathbb{N}_\nu)$ there isn't much point, since all morphisms $A \to B$ are necessarily equal.

5 Conclusions and Future Work

We have seen how the resource theoretic interpretation of monoidal categories, and in particular their string diagrams, captures the sort of material history that concerns ledger structures for blockchain systems. Additionally, we have shown how to freely add a notion of ownership to such a resource theory, and that the resulting category is equivalent to the original one. We have also shown that these resource theories with ownership admit an intuitive graphical calculus, which is more or less that of monoidal functors and natural transformations. Finally, we have used our machinery to construct a simple ledger structure and show how it might be used in practice.

While we do not claim to have solved the problem of providing a rigorous foundation for the development of ledger structures in its entirety, we feel that our approach shows promise. There are a few diffent directions for future research. One is the development of categorical models for more sophisticated ledger structures, with the eventual goal being to give a rigorous formal account of smart contracts. Another is to explore the connections of the current work with formal treatments of accounting, such as [6].

References

1 N. Atzei, M. Bartoletti, T. Cimoli, S. Lande, and R. Zunino. Unravelling bitcoin smart contracts. In *POST 2018*, volume 10804 of *LNCS*, pages 217–242, 2018.

2 N. Atzei, M. Bartolietti, and T. Cimoli. A survey of attacks on ethereum smart contracts. In *POST 2017*, volume 10204 of *LNCS*, pages 164–186, 2017.

3 J.R.B. Cockett and R.A.G. Seely. Proof theory of the cut rule. In E. Landry, editor, *Categories for the Working Philosopher*, pages 223–261. Oxford University Press, 2017.

4 B. Coecke, T. Fritz, and R.W. Spekkens. A mathematical theory of resources. *Information and Computation*, 250:59–86, 2016.

5 K. Jabbar and P. Bjorn. Infrastructural grind: Introducing blockchain technology in the shipping domain. In *GROUP 2018*, 2018.

6 P. Katis, N. Sabadini, and R.F.C. Walters. On partita doppia. *CoRR*, 1998.

7 A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. *CRYPTO 2017, Part I, volume 10401 of LNCS*, 2017.

8 S. Mac Lane. *Categories for the Working Mathematician*. Springer, 1971.

9 M.B. McCurdy. Graphical methods for tannaka duality of weak bialgebras and weak hopf algebras. *Theory and Applications of Categories*, 26:233–280, 2011.

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

11 Peter Selinger. A survey of graphical languages for monoidal categories. In *New Structures for Physics*, pages 289–355. Springer, 2010.

12 M. Staples, S. Chen, S. Falamaki, A. Ponomarev, P. Rimba, A.B. Tran, I. Weber, X. Xu, and J. Zhu. *Risks and Opportunities for Systems Using Blockchain and Smart Contracts*. Data61 (CSIRO), Sydney, 2017.

13 Gavin Wood. Ethereum: A secure decentralized generalised transaction ledger, 2014. URL: `https://gavwood.com/paper.pdf`.

Parasite Chain Detection in the IOTA Protocol

Andreas Penzkofer
IOTA Foundation, Berlin, Germany

Bartosz Kusmierz
Department of Theoretical Physics, Wroclaw University of Science and Technology, Poland

Angelo Capossele
IOTA Foundation, Berlin, Germany

William Sanders
IOTA Foundation, Berlin, Germany

Olivia Saa
Department of Applied Mathematics, Institute of Mathematics and Statistics, University of São Paulo, Brazil

Abstract

In recent years several distributed ledger technologies based on directed acyclic graphs (DAGs) have appeared on the market. Similar to blockchain technologies, DAG-based systems aim to build an immutable ledger and are faced with security concerns regarding the irreversibility of the ledger state. However, due to their more complex nature and recent popularity, the study of adversarial actions has received little attention so far. In this paper we are concerned with a particular type of attack on the IOTA cryptocurrency, more specifically a *Parasite Chain* attack that attempts to revert the history stored in the DAG structure, also called the *Tangle*.

In order to improve the security of the Tangle, we present a detection mechanism for this type of attack. In this mechanism, we embrace the complexity of the DAG structure by sampling certain aspects of it, more particularly the distribution of the number of approvers. We initially describe models that predict the distribution that should be expected for a Tangle without any malicious actors. We then introduce metrics that compare this reference distribution with the measured distribution. Upon detection, measures can then be taken to render the attack unsuccessful. We show that due to a form of the Parasite Chain that is different from the main Tangle it is possible to detect certain types of malicious chains. We also show that although the attacker may change the structure of the Parasite Chain to avoid detection, this is done so at a significant cost since the attack is rendered less efficient.

2012 ACM Subject Classification Networks → Security protocols

Keywords and phrases Distributed ledger technology, cryptocurrency, directed acyclic graph, security, attack detection algorithm

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2020.8

1 Introduction

With the arrival of Bitcoin [24] a new decentralized payment system based on a trust-less peer-to-peer network has established. Bitcoin is essentially a protocol for reaching consensus between independent entities - that do not need to trust each other - on a chronologically ordered record of transactions. This data structure, which is also called a *blockchain*, is now a cornerstone of many other Distributed Ledger Technologies (**DLT**s) [6], [14], [17], [23].

Despite their great success, events of congestion and the correlated high transaction (**tx**) fees [15] show that limitations in throughput, which can become costly, exist for these types of DLTs. Since these events are effectively created due to the bottleneck of a limit of txs that can be processed, scalability has become a core research topic. Furthermore, this issue also hinders the adoption of the technology for applications such as Internet of Things (IoT) [12].

To overcome scalability issues, several techniques have been proposed ranging from increasing block size and frequency, to side chains [10], "layer-two" structures like Lightning Network [26], Sharding [22], and other consensus mechanisms [14, 17, 7]. Some DLT's have also replaced the blockchain structure with a directed acyclic graph (**DAG**). This approach is used in IOTA [27] and other protocols [19], [9], [5], [29], [4], [20], [8], [16]. DAG-based protocols reach consensus on a *partially* ordered log of transactions, allowing the log to have width and which can increase the throughput of the system.

In this paper, we study the DAG-based IOTA protocol introduced in [27], where transactions are recorded in a DAG dubbed the *Tangle*. The vertices in this DAG are transactions. If there is an edge between two transactions $x \leftarrow y$, we say that y (directly) approves x. If there is a directed path from y to x but no edge, y indirectly approves x. A transaction with no approvers is called a *tip*. Under the IOTA protocol, all incoming transactions attach themselves to the Tangle by approving two (not necessarily distinct) tips.

1.1 The IOTA protocol

Since the DAG structure is heavily determined by the order and manner in which txs are approved, the algorithm used to select the tips which a new transaction will attach itself to, is of critical importance. It is important to note, that the nodes in this protocol are free to choose from the array of available tip selection methods. In this paper we focus on two tip selection algorithms:

Firstly, the *Uniform Random Tip Selection* (**URTS**), is a very basic algorithm: we simply select a tip from the set of all available tips with a uniform random distribution. Despite being the most efficient method numerically, it would also be accompanied by security vulnerabilities and allow for tip selection behavior that is non-beneficial for the safety of the Tangle [27]. However, as we will see in Section 2.2 it is closely related to the next algorithm, which is the one that most resembles the current implementation in the protocol. More particularly, the analytical derivations for the following algorithm depend on the solutions for this URTS algorithm.

The second tip selection algorithm employs a Monte Carlo Markov Chain: here we select the tip at the end of a random walk (**RW**), beginning at the first tx in the Tangle. In the current implementation of the IOTA protocol, the RW can be biased towards transactions with large cumulative weight, which is a tx's own weight plus the sum of all own weights of directly or indirectly approving txs. The amount of bias is determined by a parameter α. When $\alpha = 0$, we dub the RW as *Unbiased Random Walk* (**URW**). Consequently, when $\alpha > 0$, we dub it *Biased Random Walk* (**BRW**). As we will see in Section 2.3 under the current network conditions and implementation of the IOTA protocol, the BRW has very similar properties to the URW and, therefore, it is sufficient to study the URW.

1.2 The Parasite Chain attack

Due to the probabilistic nature of the immutability of the ledger state, cryptocurrencies are subject to certain security concerns [21]. One such concern is that an adversary may revert the ledger to an earlier state if he possesses a sufficient amount of hashing or voting power [28]. In practice, such an adversarial action would result in a fork of the ledger and enable the possibility for a double-spend of funds.

[27] describes several attack scenarios under which an adversary may attempt a double-spend on the IOTA protocol. In this paper we focus on the Parasite Chain (**PC**) attack. In this attack the adversary places a value tx in the main Tangle, whilst also creating a

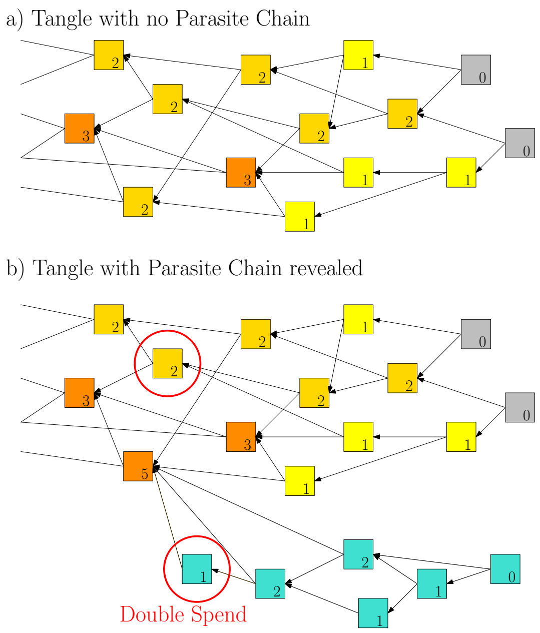

a) Tangle with no Parasite Chain
b) Tangle with Parasite Chain revealed
Double Spend

Figure 1 Examples of parts of a Tangle. Transactions are represented by squares and approvals by arrows. The number of approvers of each transaction is in the bottom right corner of each transaction. Parts (a) and (b) represent the Tangle with and without a conflicting Parasite Chain attached. Transactions that conflict with the most recent part of the main Tangle are shown in blue.

side chain in secret that contains a double-spending tx, see Fig. 1. Once the PC is revealed to the network, the attacker then exploits the tip selection algorithm by leading honest txs to approve the PC instead of the main Tangle. This is possible, through the following mechanism: firstly, by attaching the PC to a particular tx (the *root* tx), the cumulative weight of this root tx can be significantly increased and the RW tip selection algorithm will then be drawn towards this root. In addition, the attacker can also increase the number of links from the PC to this root tx, to increase the probability for the RW to continue onto the PC. If the attack is successful, the part of the Tangle that approves the originally visible double-spend tx is abandoned and the PC becomes the new main Tangle, thereby changing the ledger history.

[11] discuss this attack in further detail and show that for certain values of the parameter α in the BRW, the attack has an increased likelihood to succeed. Furthermore, although the security is improved with increasing value of α, it should not be selected too high, otherwise txs would be left behind and excluded from the ledger. Therefore, α has to be selected in a manner that is a compromise between the safety of the system and avoiding orphanage of txs, i.e. txs not being approved. More generally [11] also showed that the success of the attack also depends on other variables, such as the time that the PC is revealed to the main tangle and the number of root txs the PC is attached to the main Tangle.

In this paper, we present a method to counter the vulnerability of the PC attack by introducing certain detection mechanisms. These detection methods employ the knowledge of the underlying structure of the Tangle, more particularly the likelihood for txs to have a certain number of direct approvers. We show that by measuring the distribution of approvers for a selected set and comparing it to a reference distribution, the Tangle can be checked for abnormalities. Upon flagging a suspicious part of the Tangle as a PC, countermeasures can be taken, such as restarting the RW with an increased α value [25]. Since the tip selection method can be switched or restarted immediately during the detection, an honest tx issuer who detected the PC, would avoid approving tips of the PC and hence contribute towards rendering the attack unsuccessful. We show that an adversary would have to significantly reduce the efficacy of the attack if he wants for the PC to remain undetected.

1.3 Contributions of this paper

Our contributions in this paper are twofold: firstly, we present analytical models that capture the underlying structure of the Tangle, in the form of the likelihood for txs to have a certain number of direct approvers. We present models for the URTS and the URW, and we show that the distribution for the URW is tightly linked with the distribution for URTS. We compare the analytical models to simulation results and show that generally a good agreement is reached in the high load regime. In the low load regimes, the model predicts the values well only if txs are only allowed to approve the same tx once.

Secondly, we describe a method of how to reduce the susceptibility of the IOTA cryptocurrency towards a specific type of double-spend attack, more particularly a Parasite Chain. This enables a proactive tool against malicious actors and improvements for security. We measure how 'distant' a certain sample set of txs is from the derived distributions and we show that we can employ the distance metric, to effectively detect simple versions of a PC. If the attacker decides to avoid the detection methods, he is forced to build the PC in a more complicated way, which is correlated to a decrease of the attack's efficacy. We demonstrate this for two ways of selecting sample sets: through RWs and through collecting the txs that directly or indirectly approve a particular tx, i.e. the future cone of that tx. We conclude that by combining these two methods, a powerful detection tool is provided to honest tx issuers that allows them to make their tip selection more safe and PC attacks less likely to succeed.

2 Model for the Number of Approvers

Due to the complexity of the Tangle, it can be difficult to derive exact solutions which describe certain mechanics. On the other hand, deterministic solutions can provide a sufficiently good picture to describe certain mechanics despite their simplified assumptions. Here, we attempt to discuss the likelihood of a randomly selected tx having a given number of direct approvers n, through a deterministic model. The model is facilitated by employing a linear approximation of the exit probability distribution, which represents the likelihood of a tip being selected.

Since the tip selection is performed probabilistically, the same tip may get selected twice, although this is only likely to happen in the *low load regime*, i.e. if the rate of arriving txs is low. Since the tip selection algorithm is not enforced, it remains up to the node to decide what to do in this case. Here, we consider two particular scenarios that do not require rerunning the tip selection algorithm and which are, therefore, considered numerically efficient options. Initially, we will discuss both scenarios, before focusing on the former in more detail: in the single edge model (**SEM**) only one edge is created instead of two, and in the multi-edge model (**MEM**) two edges are created to the same tx. As we will show, in the *high load regime*, i.e. when the rate of arriving txs is high, SEM and MEM converge to the same distributions. Since the Tangle is built to allow for high throughput, we will mainly focus on the high load regime and the difference between SEM and MEM can be neglected. However, since the protocol should also be analyzed in the low load scenario and the node is free to choose the tip selection algorithm, the two methods are discussed for completeness and comparison.

Once a tx is selected for the tip selection, it is assumed that the Proof-of-Work plus the propagation to the network equal to the time h, and that no other node will be aware of the approval before this delay has passed [27]. Let p_i be the probability of being selected by the tip selection algorithm for the i-th tx, N_i the final number of approvers of i and t_i the time of first approval. We notice that p_i depends on the number of tips and hence may change with the arrival of a new tx. Note that if txs i_1 and i_2 are the approvees of i then no further txs are attached to them later than $t_i + h$, due to the delay h. It is noteworthy that the following applies for URW: after $t_i + h$ none of the directly or indirectly approved txs, i.e. the entire past cone of i, receive any further approvers. The exit probability of the URW at i remains then unaffected after $t_i + h$ by the arrival of new tips and p_i, and also the probability for the random walk to pass through the tx, remains constant after that time. This is not the case if backtracking is allowed, i.e. the RW is allowed to return to the past cone of i once it left it, however, this is currently not implemented in the IOTA protocol.

Throughout this work we will frequently employ the Poisson distribution function

$$P(\gamma, n) = e^{-\gamma}\frac{\gamma^n}{n!} \tag{1}$$

where γ is a rate.

In the following sections we employ

▶ **Lemma 1.** *The number of approvers for a given tx i is given by*

$$N_i = 1 + p_{i0} + Pois(\lambda_i) \tag{2}$$

where Pois$(\cdot)$ *is the Poisson distribution,*

$$\lambda_i = \lambda \int_{t_i}^{t_i+h} dt \begin{cases} 2p_i(t) & \text{for MEM} \\ 2p_i(t) - p_i(t)^2 & \text{for SEM} \end{cases} \tag{3}$$

λ is the tx rate in units of h, and

$$p_{i0} = \begin{cases} p_i(t_i) & \text{for MEM, approximately} \\ 0 & \text{for SEM} \end{cases} \tag{4}$$

Proof. Assume there is a Poisson process of arriving txs [27] with rate λ, i.e. the number of arrivals within the interval h after i receives its first approval is given by the random variable N. Let us consider SEM first. The probability of i receiving an additional approver, once a tx arrives is $p_i^+(t) = 1 - (1 - p_i(t))^2$. Hence from the viewpoint of i, txs arrive at rate $p_i^+\lambda$ (of which all would reference i). Furthermore, for independent events it holds that $\mathbb{P}(N = n) = \mathbb{P}(M_1 + M_2 = n)$, where $\mathbb{P}(X = x)$ is the probability that the random variable X takes the value x, and M_1 and M_2 are again Poisson processes, with rates μ_1 and μ_2. We can, therefore, assume that the arrivals of attachments to i occur through a series of Poisson processes with rate $p_i^+\lambda dt$ at time intervals dt, which leads to the integral form. In the case of MEM, we consider two rates of arriving tx that approve i: txs that approve i twice or once, and their rates of arrival are $p_i(t)^2\lambda$ and $(p_i^+(t) - p_i(t)^2)\lambda$, respectively. With the same argument as above, we can find the integral form

$$N_i = 1 + \text{Pois}(\lambda_{i1}) + 2\text{Pois}(\lambda_{i2})$$

where

$$\lambda_{i1} = \lambda \int_{t_i}^{t_i+h} 2p_i(t)(1 - p_i(t))dt$$

$$\lambda_{i2} = \lambda \int_{t_i}^{t_i+h} p_i^2(t)dt$$

These two Poisson processes are independent and can be further combined. Finally, the first approver also has the likelihood p_{i0} to approve the same tip twice. This assumption does not take into account that for small λ the Poisson process has a significant impact on the number of approvers. More specifically, for small λ the likelihood is increased that a single tip is simultaneously selected by one or multiple txs and hence the probability to have an even number of approving edges is increased compared to an odd number. ◀

Generally the average probability to have n approvers is given by

$$P(n) = \sum_{i=1}^{N} \mathbb{P}(N_i = n) \tag{5}$$

where N is the cardinality of a list of txs that are considered. Since as previously discussed for URW $p_i(t)$ is only variable for a short time (maximally h after i is revealed), we assume $p_i(t) = p_i \forall t$ is approximately true. The integral forms in (3) can, therefore, be further simplified, which we employ in the next section.

2.1 Uniform Random Tip Selection

In the URTS algorithm, tips are selected at random from the set of L available tips and with the most recent assumption $p_i = L^{-1}$ on average. According to [27], the number of tips is approximately $L = 2\lambda$. However, for small λ, the number of tips is limited by one instead. For simplicity, we assume that

$$L = 1 + 2\lambda \tag{6}$$

Using (5) and (2) the probability to have n approvers (or equivalently $n-1$ beyond the first) is then given by the Poisson distribution function

$$P_U(n) = P(\lambda_U, n-1) \tag{7}$$

with the rate

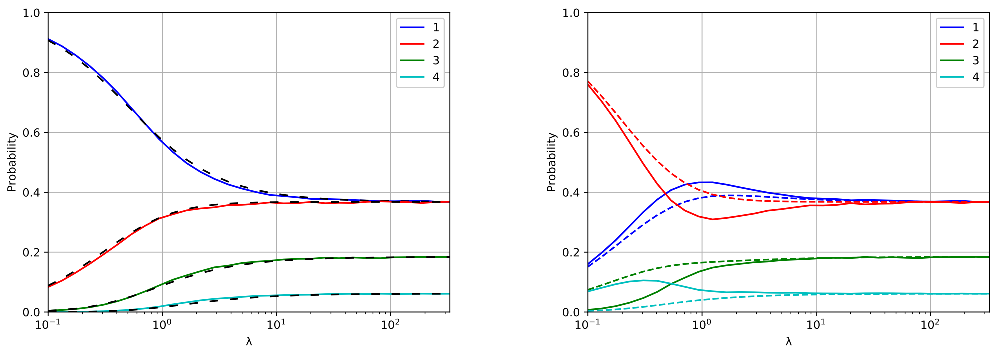
$$\lambda_U = \begin{cases} 2\lambda L^{-1} & \text{for MEM} \\ 2\lambda L^{-1}(1 - 0.5L^{-1}) & \text{for SEM} \end{cases}$$

Note that in the high load regime (i.e. λ is large), the quadratic term can be neglected and MEM and SEM converge to the same rate, as previously discussed.

Fig. 2 shows a comparison of the numerical and analytical values of $P_U(n)$ for both SEM and MEM. It can be seen that for SEM the model represents the numerical values well. In both models, the distribution converges towards the same distribution in the high-load regime. This is because for large λ almost none of the tips are selected twice. However, in the low load regime (i.e. $\lambda < 10$) the predicted values do not match for MEM. This is due to the discrete nature of the Poisson process, which as described above, is not accounted for. Furthermore, even numbers of approvers have a higher probability (see $n = 4$), while odd numbers of approvers are less likely than predicted (see $n = 3$).

2.2 Unbiased Random Walk

The tip selection algorithm that is currently employed in IOTA is the Monte-Carlo-Markov-Chain RW with the parameter α [27]. Typically, analytical solutions are easier to find by initially considering $\alpha = 0$, where effects such as txs that are left behind, or the dependence



a) Single-Edge model (SEM) b) Multi-Edge model (MEM)

Figure 2 Probability for a randomly selected tx, in a Tangle created by URTS, to have n approvers with λ for $n = \{1, .., 4\}$. Values from simulation results and analytically predicted values are shown with continuous and dashed lines, respectively.

of the RW on the cumulative weight, play no role. Here we follow the same path, before increasing the complexity of the analysis by considering $\alpha > 0$ in Section 2.3. We employ SEM for the analysis in this section, since the analytical model provides more accurate results in this case. However, for most of our analysis, both models would be appropriate, since both predict the numerical values well in the high-load regime, which is in the focus in this work. Note that [3] observed that for $\alpha = 0$, the mean tip number converges to $L \approx 2.1 \cdot \lambda$. However, employing this observation does not lead to improvements for this model and hence the same average tip number as for URTS is employed.

For a given number of tips L, we can order the exit probabilities by their likelihood. We then define the L-normalized exit probability $e(x)$ to be the exit probability distribution that is normalized from the index interval $\{1, .., L\}$ onto the relative index interval $(0, 1]$. For large enough λ, the expected exit probability of the $i-th$ most likely tip is then approximately given by

$$\mathbb{E}e_L(i) \approx \int_{(i-1)/L}^{i/L} e(x)dx \tag{8}$$

The L-normalized exit probability can be expressed as

$$e(x) = 1 + f(x) \tag{9}$$

where $x \in [0, 1]$, $\int_0^1 f(x) = 0$ and $f(x) > -1$.

Fig. 3 shows the numerically calculated L-normalized exit probability for URTS and URW, as well as a linear fit for URW. $e(x)$ has numerically been calculated by averaging over 10^3 samples of tip sets. For each tip set, 10^6 tip selections are performed and the tips are ordered by their exit probability $e_L(i)$. Despite the high number of tip selection samples, the values of indices close to 0 (1) are slightly over- (under-) estimated due to stochastic effects, as can be seen for URTS where the value is expected to be $e_U(x) = 1$ over the entire interval. It can be seen that for the fitted curve the numerical values agree well for most of the range, apart from the smaller indices, where the exit probability is noticeably increased, suggesting that certain tips in the Tangle have a considerably higher probability to be selected.

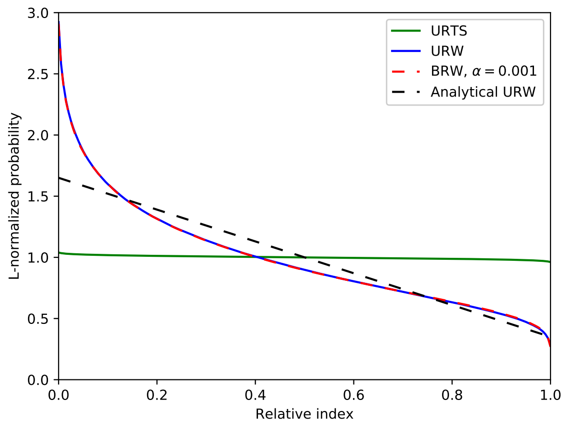


Figure 3 L-normalized exit probability with the ordered relative index number for $\lambda = 100$. The curve for the analytical linear approach is shown for $a = 1.3$.

For a given relative index number x, we can assume a Poisson process of txs and the probability density of having n approvers is then given by

$$p_{URW}(n, x) = P(e_{URW}(x)\lambda_U, n - 1) \tag{10}$$

The expected value is then given by

$$\begin{aligned} P_{URW}(n) &= \int_0^1 dx\; e^{-e_{URW}(x)\lambda_U} \frac{(e_{URW}(x)\lambda_U)^{n-1}}{(n-1)!} \\ &= P_U(n) \int_0^1 dx\; e^{-f(x)\lambda_U} (1 + f(x))^{n-1} \end{aligned} \tag{11}$$

As can be seen, the distribution for the URW can be decomposed into a product of the distribution for URTS times a factor dependent on the exit probability function. This highlights how these two tip selection algorithms are correlated, and why their distribution appears similar.

For URW, the probability to pass through a given tx, is equal to the final value of the exit probability during the time the tx was a tip. Hence to derive the approver statistics observed along the path of RWs, (10) is weight by $e(x)$, and the expected distribution is

$$P^*_{URW}(n) = \frac{1}{b} \int_0^1 dx\; p_{URW}(n, x) e(x) \tag{12}$$

where

$$b = \sum_{n=0}^{\infty} \int_0^1 dx\; p_{URW}(n, x) e(x)$$

normalizes the probability.

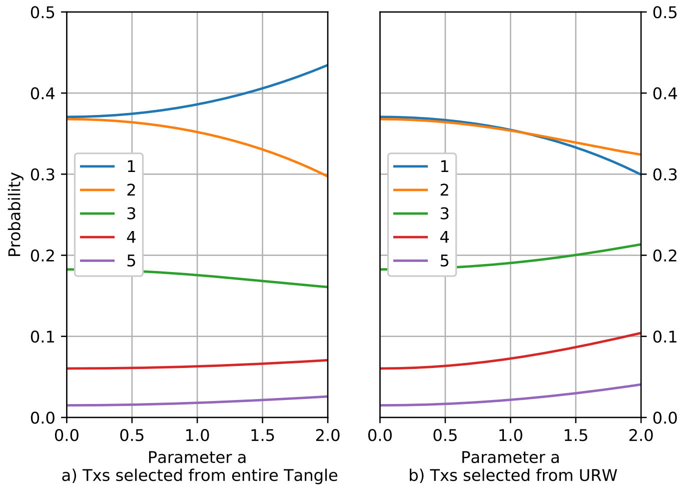


Figure 4 Values of (13) and (14) with the parameter a for several values of approver numbers n. $\lambda = 100$.

Linear approach

A simplified approach is presented by employing the distribution $f(x) = a(x - 0.5)$, $a \in [0, 2]$, where a is limited to achieve only positive values for f. The expected value is then given by

$$P_{URW}(n) = P_U(n)g(n-1) \tag{13}$$

$$P^*_{URW}(n) = \frac{P_U(n)g(n)}{\sum_{m=1}^{\infty} P_U(m)g(m)} \tag{14}$$

where

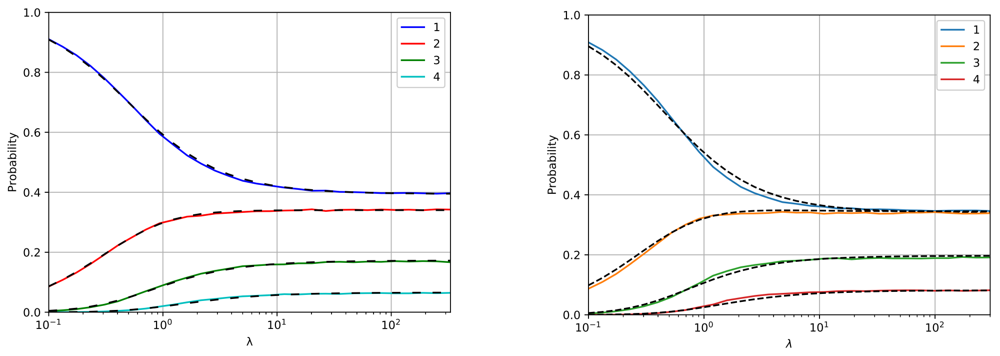
$$g(n) = \frac{1}{a}\sum_{j=0}^{n} \lambda_U^{-j-1} \frac{n!}{(n-j)!} \left[e^{-y\lambda_U}(1+y)^{n-j}\right]_{0.5a}^{-0.5a}$$

Note that for $a \to 0$ (13) converges towards URTS:

$$\frac{P_{URW}(n)}{P_U(n)} = -\sum_{j=0}^{n-1} \frac{(n-1)!}{(n-1-j)!}(n-2-j) = 1$$

Fig. 4 shows the probability to have n approvers with the value of the parameter a. As can be seen in Fig. 4a), introducing a gradient in the exit probability shifts the likelihood of having a certain number of approvers away from the mean towards less (one approver) or higher numbers of approvers (more than 3), corresponding to increased or decreased exit probabilities. On the other hand, it can be seen from Fig. 4b) that the likelihood for a URW to visit sites with a higher number of approvers, is shifted to only txs with higher numbers (more than 2), since these are sites more frequently visited by the URW.

From Figs. 3 and 5, we can see that a good agreement is achieved with the numerical results for $a = 1.3$.



a) Randomly selecting a tx from the entire Tangle b) Randomly selecting a tx from an URW

Figure 5 Probability for a tx to have n approvers with λ for $n = \{1, .., 4\}$; for SEM and URW tip selection, and $a = 1.3$. Values from simulation results and analytically predicted values are shown with continuous and dashed lines, respectively.

2.3 Biased Random Walk

In the IOTA protocol, the RW is made dependent on the cumulative weight of a tx. The cumulative weight w_x of a tx x is defined as the sum of its own weight, plus all own weights of txs directly or indirectly approving the tx. The dependency on the RW is introduced to prevent undesirable behaviors, such as the selection of old tips (lazy tip selection), or certain types of PCs [27]. In the current implementation of the Tangle, the path of the RW depends, therefore, on a parameter α and the cumulative weight of txs encountered along the path. More specifically, the probability to transition from a tx x to a tx y is given by [27]:

$$P(x \rightsquigarrow y) = \frac{\exp(\alpha w_y)}{\sum_z \exp(\alpha w_z)} \tag{15}$$

where the denominator normalizes the probability by summing over all direct approvers z (including y) of x.

It should be noted that the value of the parameter α ought to be selected carefully. Selecting the value too low would result in the BRW having little or no improvement to the security compared to URW, while a too high value would result in txs being orphaned. Typically, orphanage behavior is observed in simulations at about $\alpha\lambda > 1$ [18]. The default value in the IOTA reference implementation of $\alpha = 0.001$ [1] ensures that the Tangle structure is the same at least up to a tx rate of $\lambda = 100$, since no orphans are created and the exit probability is the same, as can be seen in Fig. 3.

With the Minimum Weight Magnitude parameter set to 14 (current setting of the difficulty in the IOTA protocol), the average time for the PoW in a Core i7 platform is about $4.1s$ [2]. At a tx rate in the order of 5 tps (current, over the scope of one day, measured tps by an IRI node [1]), the expected value for the tx rate is $\lambda \approx 20$. Since $\lambda \gg 1$, the difference between SEM and MEM is negligible. Furthermore, since $\lambda\alpha \ll 1$, the difference between the exit probabilities and hence the approver distributions of URW and BRW are negligible, as discussed in the previous paragraph. We, therefore, employ the models developed in Section 2.2 for the following sections.

3 Parasite Chain detection

The vulnerability of the Tangle to the PC attack is first recognized in [27]. A PC is a specific type of a double-spend attack, where the attacker issues a tx on the main Tangle and, simultaneously, issues a double-spending tx, which is not yet revealed. He then continues to issue further txs that approve the double-spend on the PC in secret, to strengthen the PC chain, till the receiver accepts the main Tangle double-spend tx. Upon receiving the goods or value from the to-be-swindled receiver, the adversary then reveals the PC in the hope that it can outpace the incompatible part of the current main Tangle, by attracting as many tip selection RWs as possible. If successful, the part of the Tangle that approves the main Tangle double-spend tx becomes orphaned, and the PC becomes the new main Tangle, thereby invalidating the main Tangle double-spend tx.

In this paper, we focus on a specific Parasite Chain attack where the adversary pins the PC to one or more (k) honest txs in the main Tangle, which are issued at a similar time as the double-spend tx on the main Tangle. We dub this type of a PC k-pinned PC. Generally, an adversary may choose to attach to multiple root txs. However, for simplicity we assume a 1-pinned PC.

We assume that the adversary attempts to issue as many txs as possible that directly approve the PC root, as well as the previously issued PC tx. This approach is taken for two reasons: firstly, by directly or indirectly approving this root tx, the cumulative weight is increased both by the txs in the main Tangle, as well as by the PC. This can significantly increase the cumulative weight of the root, compared to the double-spend tx in the main Tangle. As a result, the RWs that are performed for tip selection will be drawn towards the root. Secondly, by attaching as many direct approvers to the root, the probability for the RW to hop onto the PC once the RW passes through the root, depends on the number of links to the PC. It can, therefore, be increased by directly attaching as many malicious txs as possible. Fig. 6a) shows the most efficient way of attaching a 1-pinned PC, and we dub it a simple PC (**SPC**).

In this section, we describe a methodology for how we can employ the distributions that we derived in the previous section, to detect a pinned PC. More particularly, since the distribution of approvers in the PC in Fig. 6a) is significantly different from the distributions derived in the Section 2, we propose the implementation of a metric that compares the encountered approver distribution to the expected one, and allows to identify PCs. This would allow nodes to actively counter the creation and success of PCs.

We define the following distance metric

$$d_P = \frac{1}{2}\sum_{n=0}^{\infty}|P(n,S) - P_{ref}(n)| \tag{16}$$

where P is the measured distribution, or probability vector, that is deduced by measuring on the sample of size S. In this paper, the sample size is varied between 10 and 100, see Fig. 7. P_{ref} is the reference distribution (equations (13) or (14)) and the factor $\frac{1}{2}$ normalizes the distance.

As discussed, it is more expensive for the attacker to create txs in the PC with higher numbers of approvers. Hence, it is useful to put increased weight on txs with higher numbers of approvers. However, due to the low probability of having a high number of approvers, the difference between the actual value $P(n)$ and $P_{ref}(n)$ can be relatively large for higher values of n. To reward having a high number of approvers (instead of penalizing it because of the high variance), we measure if a tx has more than a given number of approvers, and we

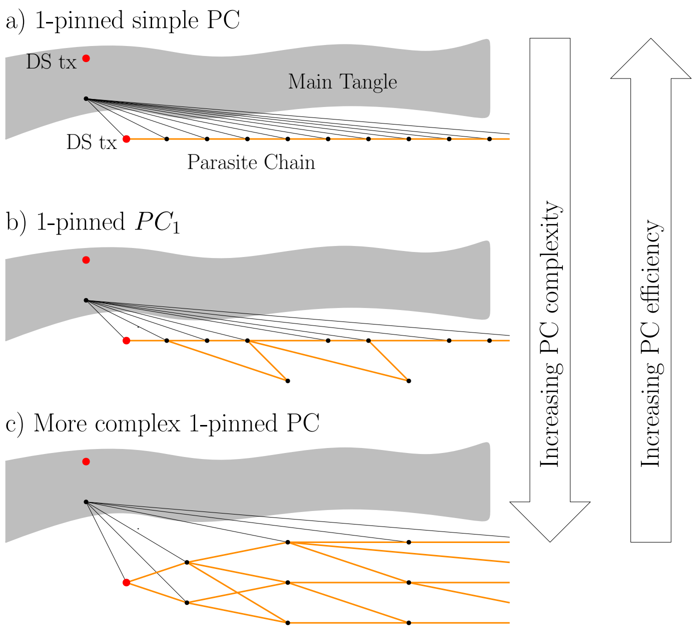


Figure 6 Different types of PCs with an increasing effort of the adversary to hide the PC from the detection methods. The number of indicated malicious txs is kept the same for a)-c). With higher complexity, fewer links to the root tx in the main Tangle are created.

can therefore also employ the distance

$$d_Q = \frac{1}{2}\sum_{n=0}^{\infty} |Q(n) - Q_{ref}(n)| \tag{17}$$

where

$$Q(n) = \sum_{m=n}^{\infty} P(m) \tag{18}$$

is the cumulative probability distribution and Q_{ref} is the reference cumulative probability distribution. As will be shown in the following figures, employing d_Q instead of d_P can lead to a better analysis, whilst adding little or no computational overhead for calculating the more complicated metric d_Q.

We can employ these metrics in the following way: if the measured distance exceeds a critical value η, the detection method rejects this path for the RW. More formally, we say a sample P fails an η-confidence level test if $d_P > \eta$ (or equivalently, the sample Q fails if $d_Q > \eta$). The exact procedure after rejecting a path can be manifold. For example, the node may simply restart the RW from the last point where no suspicious behavior was noted. Or the node may also switch into some kind of safe mode, where the RW is calculated with a higher α value [25].

Generally, the measurement of the distance is made on a limited sample size and the distances d_P and d_Q can, therefore, only take certain discrete values. To determine how likely it is that a measurement is treated as a false-positive, i.e. the sample is part of the main Tangle but has been flagged to be in a PC, we study how likely it is that a certain distance occurs in a random sample. Due to the discreteness of the above distance metrics, we investigate the cumulative probability rather than the actual distribution. Note that for small

Algorithm 1 Integration of detection into tip selection.

```
input  : The starting point of the RW defined as initial_tx
output : The selected tip
1  tx ← initial_tx
2  mode ← standard
3  while tx is NOT a tip do
4      if mode = safe then
5          tx ← safe_step(tx)
6      else
7          tx ← standard_step(tx)
8          if Algorithm 2 = TRUE then
9              mode ← safe
10             tx ← initial_tx
11 return tx
```

sample sizes S, this discreteness can be observed through the clearly visible discontinuity of the derivative of the curves, see e.g. Fig. 7 for $S = 10$. Intuitively, the cumulative probability also provides a visual representation as to how many samples would be below a given distance, and how many false-positives we should expect, when setting η.

3.1 Random walk detection

In this method of creating a sample of approver numbers, we collect and update the sample set of txs while the RW traverses the Tangle. This method is numerically very inexpensive, since the information for the sample set is readily available, as it is already considered when performing the RW. Hence the node may also decide to test samples of different lengths in parallel, in order to distinguish between very local distributions (small sample size, e.g. $S = 10$) and more global distributions (e.g. $S > 50$).

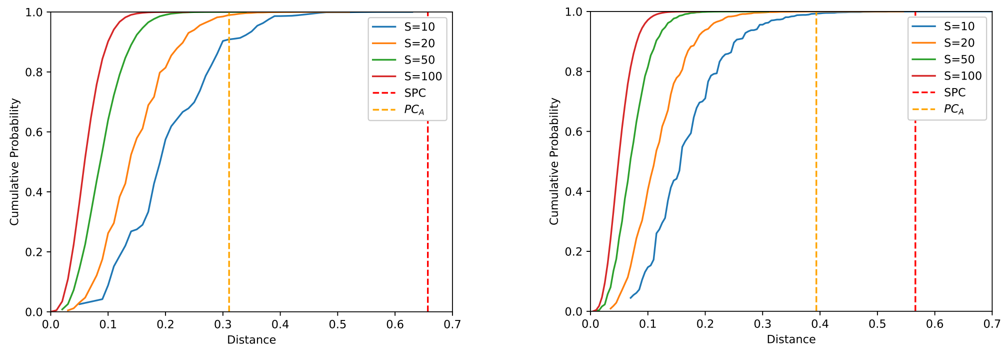
The pseudo code described in Algorithm 1 provides an example high-level overview of how the PC detection mechanism could be implemented in a tip selection algorithm. The additional algorithm 2 is added, which updates and evaluates for every RW step a sample of approver numbers. If the calculated distance d is larger than a given threshold, the algorithm 2 returns a flag and the tip selection method restarts, in this case, from the initial starting point of the RW. Furthermore, the RW is changed from the standard RW (with a *standard_step*) into a safe mode ($safe\_step$). This safe mode could, for example, be a RW with an increased α value, as described in [25].

Fig. 7 shows the cumulative probability for the distance to be above a certain value, where the sample size of considered txs has been fixed to S and P_{ref} is given by (14). It can be seen that for small values of S the discrete nature of the distance is clearly visible, while for larger S it becomes increasingly difficult to observe this phenomenon. Note, that the maximum available number of samples from an RW depends on the depth from which the RW is started. Furthermore, S should be selected such that local changes (i.e. the start of a PC) can be detected, which puts further restrictions on the upper limit of S.

Let us assume the adversary attaches a 1-pinned PC to the Tangle, and that with probability p_R a malicious tx directly approves the PC root, as well as the previous malicious tx that was attached in the same manner. We denote the part of the PC that is constructed

Algorithm 2 Sample Management.

```
  input  : current tx, sample list of approver numbers
  output : bool value wheter PC is detected
1 Add number of approvers of tx to the sample list
2 if $Sample\ size > S$ then
3     Remove the oldest element from the sample list
4 Calculate distance d by applying Eq. (16) or (17)
5 pc_detected ← (d > threshold)
6 return pc_detected
```



a) Distance metric d_P b) Distance metric d_Q

Figure 7 Cumulative probability for the measured distance to be below a given value for several sample sizes S. $\lambda = 100$, URW.

in this way as the *main PC*. The remaining malicious txs are attached in whatever way seems most suitable. Then, the most efficient way the adversary can build the PC is with $p_R = 1$, see Fig. 6a), where all malicious txs are added as direct approvers to the PC root and the rate at which direct approvers are added to it is given by $r = \mu$.

As can be seen from Fig. 7, an SPC is easily detectable, since even for a small number of samples S, the measured distance in the PC is larger than the measurements in the main Tangle. By decreasing p_R and attaching to txs along the main PC (1-pinned PC_1, see Fig. 6b)), the adversary may render the PC undetectable. However, by doing so he weakens the attack since fewer edges are attached to the PC root. Note also that in the PC_1 in Fig. 6b), txs that are not part of the main PC are left behind and due to their low cumulative weight, are therefore unlikely to be selected by the RW as tips.

The selected sample size S should be sufficiently large, since otherwise the detection method may show little success (or return too many false-positives, if the value of d_C is set too low). For example, denote the following PC as PC_A: an adversary employs the strategy PC_1, by adding a link to the main PC with the probability $1 - 0.5P^*_{URW}(2)$. Then

$$d_P = 1 - P^*(1) - P^*(2) \approx 0.26$$

and the attachment to the root is reduced to the rate

$$r = (1 + 0.5P^*_{URW}(2))^{-1}\mu \approx 0.85\mu$$

If S is selected too low (e.g. $S = 10$) this type of PC may remain undetected, if d_P (equation

16) is employed for detection, as can be seen from Fig. 7a). However, it still may be detectable when changing the detection metric to d_Q (equation 17).

The PC would certainly remain undetected, if txs are added to the main PC, such that the encountered distribution is $P^*_{URW}(n)$, i.e. $d = 0$. Under this condition, the rate is reduced to

$$r = (1 + \sum_{n=1} (n-1) P^*_{URW}(n))^{-1} \mu \tag{19}$$

For example, for $\lambda = 100$ and $\alpha = 0.0001$, $r \approx 0.46\mu$, which requires the attacker to deploy the malicious txs in a much less efficient way, hence reducing the efficacy of the entire PC attack.

3.2 Future cone detection

In a pinned PC, the capability of those txs that are pinned to the root, to divert the RW onto the PC relies on their cumulative weight growing as fast as possible. In other words, it is likely that most of the issued malicious txs reference, directly or indirectly, earlier malicious txs. An effective method to consider most of the PC txs is, therefore, to analyze the future cone of the txs that are encountered along an RW, where the future cone of a tx is defined as the set of txs that directly or indirectly approve a given tx.

Note that compared to the method in the previous section, the sample size S can be significantly larger, since the number of samples in the main Tangle would increase exponentially with the distance from the cone's root tx. Until the growth of the future cone reaches the linear phase, where the rate of txs per unit of distance to the root would be constant. However, while the method in Section 3.1 is computationally relatively inexpensive, here we must employ a traverse algorithm to efficiently collect a sample of txs, such as Breath-First Search or Depth-First Search [13]. It is, therefore, recommended to employ this control mechanism only occasionally or if suspicion is raised.

Due to the larger possible sample size and the way it is sampled, the adversary must attempt to achieve a better agreement of the PC with the reference distribution (13). For example, the relatively inexpensive PC in 6b) would exhibit a high distance d_P (d_Q), due to the suspiciously large amount of orphans.

The adversary may still create a PC structure that makes the PC less likely to be detected, as Fig. 6c) indicates. However, since the future cone envelopes the distribution of all txs within a certain distance of the investigated root tx (and in the future cone), the necessary structure becomes significantly more complicated. More specifically, since the reference distribution itself stems from a structure where all links are employed, this leaves effectively little to no spare links for the adversary to attach to the PC root: due to the fact that the PC is created in secret, the only approvers that the txs can receive before being revealed are the txs from within the PC itself. Hence the average number of approvers in the PC is

$$\overline{n}_{PC} = \sum_n n P_{PC}(n) \leq \sum_n n P_{ref}(n) = 2$$

where P_{PC} is the probability distribution within the PC and the inequality arises, since the PC creates links to txs within the main Tangle, without receiving any in return. The second equality in the above equation is due to the fact that the total average number of edges per tx in the Tangle must be equal to the number of approvers a tx selects, which is two in the high load regime. Hence, the PC structure that provides optimal resistance to the detection method would require most links to be employed within the PC. The adversary must, therefore, choose a medium between detectability and efficacy of the PC, by deploying links purely within the PC, i.e. he cannot create direct links to the PC root.

In conclusion, the main observation is that, contrary to the approach in Section 3.1 where adding a few tx according to the method in 6b) may be sufficient, in this approach it is more difficult to reproduce the reference distribution. More precisely, in order to imitate the exact distribution, all adversary txs would need to be deployed for mimicking the distribution.

3.3 Final remarks

Further improvements can be envisioned for the presented detection tools. For example, we may combine the RW detection in Section 3.1 and the future cone detection 3.2, since their distributions are different as can be seen by comparing Figs. 5a) and 5b). Avoiding both detection mechanisms may be noticeably more difficult and their combination could, therefore, lead to an improved success rate. The numerically expensive method in Section 3.2 may also be more suitable for taking larger samples, while the method in Section 3.1 can be easily implemented without much additional numerical effort. Furthermore, the sample size could be continuously varied and alternative sampling mechanisms could be envisioned, which would make it more difficult for the adversary to adopt his strategy.

4 Conclusion

We present models to understand and predict a particular aspect of the Tangle structure, more specifically, the likelihood for a transaction to obtain a particular number of direct approvers. We derive solutions for two tip selection mechanisms: firstly, for a Tangle that is built employing a uniform random selection and secondly, if it is constructed employing a random walk. We show that the distributions for the two tip selection methods are linked tightly and also that the distribution depends on whether samples for transactions are taken from the set of all transactions, or only from transactions which are encountered along random walks.

We then employ the models for the approver distributions to detect a specific double-spend attack on the IOTA cryptocurrency, namely the *Parasite Chain* attack. In this type of attack, an adversary attempts to trick a node's tip selection mechanism into approving a double-spend transaction, by building a side Tangle in secret and revealing it at a suitable moment. Since the efficacy of the attack relies on building a side tangle that directly approves a limited set of transactions in the main Tangle, the underlying structure of that Parasite Chain is noticeably different to the main Tangle. By measuring the *distance* to the derived distributions on direct approvers, it is possible to detect certain forms of this side chain. It is shown that the quality of the detection depends on the sample size and we, therefore, propose two different methods of sampling. Firstly, a numerically inexpensive method is proposed, where the node may record the approver statistics along the path of a random walk. However, since the random walk moves relatively fast through the Tangle, a limited amount of transactions can be sampled. In the second approach, the node chooses to sample the future cone of a given transaction. This would ensure that most of the transactions in the Parasite Chain can be measured against the expected distribution.

Through these methods, certain structures of the Parasite Chain may be detectable. This would allow the honest tx issuers to improve their tip selection algorithm to be more safe and may prevent Parasite Chain attacks from being successful. On the other hand, if the adversary chooses to avoid detection by constructing more complex forms that would be more difficult to detect, he has to deploy a significant proportion of transactions in a less effective manner. In either case, the proposed methods present a tool which can reduce the threat imposed by Parasite Chains.

References

1 https://github.com/iotaledger/iota.go.

2 Elsts A., Mitskas E., and Oikonomou G. Distributed Ledger Technology and the Internet of Things: A Feasibility Study, 2019.

3 Kuśmierz B., Staupe P., and Gal A. Extracting Tangle Properties in Continuous Time via Large-Scale Simulations, 2017.

4 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. URL: `https://eprint.iacr.org/2017/300`.

5 Xavier Boyen, Christopher Carr, and Thomas Haines. Blockchain-Free Cryptocurrencies: A Framework for Truly Decentralised Fast Transactions. Cryptology ePrint Archive, Report 2016/871, 2016. URL: `https://eprint.iacr.org/2016/871`.

6 Vitalik Buterin et al. A next-generation smart contract and decentralized application platform, 2014. URL: `https://github.com/ethereum/wiki/wiki/White-Paper`.

7 Vitalik Buterin and Virgil Griffith. Casper the Friendly Finality Gadget. *ArXiv e-prints*, page arXiv:1710.09437, October 2017. `arXiv:1710.09437`.

8 Tai-Yuan Chen, Wei-Ning Huang, Po-Chun Kuo, Hao Chung, and Tzu-Wei Chao. DEXON: A Highly Scalable, Decentralized DAG-Based Consensus Algorithm. *ArXiv e-prints*, page arXiv:1811.07525, November 2018. `arXiv:1811.07525`.

9 Anton Churyumov. Byteball: A decentralized system for storage and transfer of value, 2016. URL: `https://byteball.org/Byteball.pdf`.

10 Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, et al. On scaling decentralized blockchains. In *International Conference on Financial Cryptography and Data Security*, pages 106–125. Springer, 2016.

11 Andrew Cullen, Pietro Ferraro, Christopher King, and Robert Shorten. Distributed ledger technology for iot: Parasite chain attacks. *CoRR*, 2019.

12 Ali Dorri, Salil S Kanhere, and Raja Jurdak. Towards an optimized blockchain for iot. In *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*, pages 173–178. ACM, 2017.

13 Tom Everitt and Marcus Hutter. A Topological Approach to Meta-heuristics : Analytical Results on the BFS vs . DFS Algorithm Selection Problem . *CoRR*, pages 1–58, 2018. `arXiv:arXiv:1509.02709v2`.

14 Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 51–68. ACM, 2017.

15 Huberman Gur, Leshno Jacob D., and Moallemi Ciamac C. Monopoly without a Monopolist: An Economic Analysis of the Bitcoin Payment System, 2017.

16 K. Karlsson, W. Jiang, S. Wicker, D. Adams, E. Ma, R. van Renesse, and H. Weatherspoon. Vegvisir: A Partition-Tolerant Blockchain for the Internet-of-Things. In *Proceedings of the IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 1150–1158, July 2018. `doi:10.1109/ICDCS.2018.00114`.

17 Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*, pages 357–388. Springer, 2017.

18 Bartosz Kuśmierz and Alon Gal. Probability of being left behind and probability of becoming permanent tip in the Tangle, 2016.

19 Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. Inclusive block chain protocols. In *Proceedings of the 19th International Conference on Financial Cryptography and Data Security*, pages 528–547. Springer, 2015.

20 Chenxing Li, Peilun Li, Dong Zhou, Wei Xu, Fan Long, and Andrew Yao. Scaling Nakamoto Consensus to Thousands of Transactions per Second. *ArXiv e-prints*, page arXiv:1805.03870, May 2018. arXiv:1805.03870.

21 Xiaoqi Li, Peng Jiang, Ting Chen, Xiapu Luo, and Qiaoyan Wen. A survey on the security of blockchain systems. *Future Generation Computer Systems*, August 2017. doi:10.1016/j.future.2017.08.020.

22 Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 17–30. ACM, 2016.

23 Ujan Mukhopadhyay, Anthony Skjellum, Oluwakemi Hambolu, Jon Oakley, Lu Yu, and Richard Brooks. A brief survey of cryptocurrency systems. In *Proceedings of the 14th Annual Conference on Privacy, Security and Trust (PST)*, pages 745–752. IEEE, 2016.

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

25 Ferraro P., King C., and Shorten R. Iota-based directed acyclic graphs without orphans, 2019.

26 Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments, 2016. URL: https://lightning.network/lightning-network-paper.pdf.

27 Serguei Popov. The tangle, 2015. URL: https://iota.org/IOTA_Whitepaper.pdf.

28 Meni Rosenfeld. Analysis of hashrate-based double-spending. *CoRR*, pages 1–13, 2014. arXiv:arXiv:1402.2009v1.

29 Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. Spectre: A fast and scalable cryptocurrency protocol. Cryptology ePrint Archive, Report 2016/1159, 2016. URL: https://eprint.iacr.org/2016/1159.

Implementation Study of Two Verifiable Delay Functions

Vidal Attias
IOTA Foundation, Berlin, Germany
vidal.attias@iota.org

Luigi Vigneri
IOTA Foundation, Berlin, Germany
luigi.vigneri@iota.org

Vassil Dimitrov
IOTA Foundation, Berlin, Germany
vassil@iota.org

Abstract

Proof of Work is a prevalent mechanism to prove investment of time in blockchain projects. However, the use of massive parallelism and specialized hardware gives an unfair advantage to a small portion of nodes and raises environmental and economical concerns. In this paper, we provide an implementation study of two Verifiable Delay Functions, a new cryptographic primitive achieving Proof of Work goals in an unparallelizable way. We provide simulation results and an optimization based on a multiexponentiation algorithm.

2012 ACM Subject Classification Computing methodologies → Simulation evaluation

Keywords and phrases Blockchain, Distributed Ledger, Verifiable Delay Function, Cryptography, Simulation, RSA

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2020.9

1 Introduction

In Distributed Ledger Technology (DLT) project, the protocol can ask participants to invest scarce resources to guarantee their stakes in the good development of the network. These scarce resources can be time, money, hard drive storage, etc. The most famous mechanism of proving investment is Proof of Work (PoW) [2] where a user tries to find a correct input of a hashing function such that the output begins with a certain amount of zeros in its binary representation. This method originally intended to prove the investment of the user's time. However, its parallelizable nature has led to the so-called "mining races", presenting now serious environmental and economical concerns. In some alternative protocols, such as IOTA [22], the network does not make any distinction between miners and users. Hence, an *explicit* rate control mechanism becomes necessary to limit user's transactions and to prevent synchronicity losses between nodes. The basic idea is to impose on every user the computation of certain work to cap their throughput. If this work is performed through PoW, specialized hardware could solve it much faster than low power devices, leading to unfair advantages and potentially leading to a denial of service attacks. Conversely, our suggestion is to use an anti-spam mechanism based on Verifiable Delay Functions (VDFs).

A VDF is a function defined formally by Boneh et al. [7] that runs in a minimum amount of time which cannot be parallelized, but is exponentially easier to verify. One can set a certain difficulty and a certain amount of time of computing as parameters of the VDF. The VDF solution is unique and sound, which means an adversary has negligible chances to find the correct solution by randomly guessing. VDFs have been largely investigated on their theoretical aspect, however, there are no academic results on implementation metrics to our knowledge although some competitions gave some results on FPGA. In this paper, we make the following contributions: (i) we study two constructions proposed by Pietrzak and Wesolowski, (ii) compare their behavior in the experimental aspect, and (iii) we provide an optimization using a multiexponentiation algorithm.

A VDF is composed of three algorithms.

- The *setup* which initializes the environment in which the VDF will be evaluated, for example, RSA group or elliptic curves and setups an input space $\mathcal{X}$ and an output space $\mathcal{Y}$.
- The *evaluation* which takes as an input an element $x \in \mathcal{X}$ and a certain difficulty $\tau \in \mathbb{N}$ and outputs an element $y \in \mathcal{Y}$ and eventually a proof π which can speed up the verification.
- The *verification* where a veryfier takes as an input (x, τ, y, π) and outputs $\top$ if y is indeed the right output otherwise it returns $\bot$.

VDFs also have applications in random number generation [7, 18], RSA accumulators (primitives to produce timestamping and membership testing in a space-efficient way) or Proof of Replication [3, 15], which is an optimized version of Proof of Space [14].

The rest of the paper is organized as follows: in Section 2 we will present the previous work on VDFs and a multiexponentiation method; in Section 3, we will show how these VDFs behave when implemented, and in 4 we will present two VDF optimizations; in Section 5, we will discuss the aforementioned results and how to choose the RSA modulus choice, and we will conclude our paper in Section 6.

2 Related work

A simple way to impose a rate control mechanism in DLT protocols is to ask users to compute some work which takes a roughly predictable time as proposed in [2]. However, this proposal suggests the usage of PoW. Its parallelizable nature leads to very different solution time between specialized and non-specialized hardware. In this section, we present other functions that take a long time to compute but are simple to verify.

Rivest et al. [24] have first defined in 1978 *time-lock* puzzles based on squaring in an RSA group and are tasks inherently sequential with a difficulty easy to set. Bitansky et al. [4] then provided a formal framework for *time-lock* puzzles. Nonetheless, a verifier needs to know a private key to verify the puzzle solution, which prevents from building a universally verifiable mechanism.

Later, Boneh et al. [7] have formalized VDFs providing a high-level framework: in this context, two constructions were proposed by Pietrzak [21] and Wesolowski [26] both using RSA groups, in a similar but new way than *time-lock* puzzles, and differing in their proofs. VDFs in their most general definition, are universally verifiable functions taking a sequential time to compute. De Feo [12] proposed a third construction based on elliptic curves. However, this construction allows only one difficulty per *setup* which is a critical flaw when we need a dynamic challenge.

In the next subsections, we introduce some definitions (Section 2.1), we present the details of Wesolowski and Pietrzak's constructions (Section 2.2 and 2.3 respectively), and we provide optimization for the evaluation of the Wesolowski VDF based on multiexponentiation (Section 2.4).

2.1 RSA environments

The RSA setup [24] is one of the oldest public key ciphering cryptosystems and yet still massively used. The idea is to generate a big number $N = p \cdot q$ with p and q two prime numbers of the same order and N a λ-bits number. Typically we use $\lambda = 2048$ for high security. We then define $\phi(N) = (p-1) \cdot (q-1)$ the Euler's totient function. Then the group $\frac{\mathbb{Z}}{n\mathbb{Z}} = \{0, 1, \ldots, N-1\}$ is called an *RSA* group.

2.2 Efficient VDF - Wesolowski

2.2.1 Setup

The Wesolowski's VDF setup requires λ, and a security parameter k (typically between 128 and 256) as an input. It generates an RSA public modulus N of bit length λ and a cryptographic hashing function $H : \{0,1\}^* \mapsto \{0,1\}^{2k}$. We then define, for any $m \in \{0,1\}^*$, $H_{prime}(m) = \texttt{next\_prime}(H(m))$ returning the closest prime numbers larger or equal to $H(m)$.

2.2.2 Evaluation

The *evaluation* takes as input $\tau \in \mathbb{N}$ and $m \in \{0,1\}^*$, and then computes $x = H(m)$ and solves the challenge $y = x^{2^\tau} \mod N$. It is important to know that if the evaluator knows $\phi(N)$, it can cut this computation because $x^{2^\tau} \mod N = x^{2^\tau \mod \phi(N)} \mod N$ which reduces considerably the exponentiation cost.

2.2.3 Proof

The *proof* begins by computing $l = H_{prime}(x+y)$ and then $\pi = x^{\lfloor 2^\tau / l \rfloor} \mod N$. This can be parallelized and for s cores it takes a $\frac{2\tau}{s \log(\tau)}$ time. At the end of this phase, the evaluator can publicly use the pair (l, π) as a proof of computation. In Algorithm 1 we present a pseudo code of *evaluation* and *proof* phases.

Algorithm 1 Evaluation and proof of the Wesolowski construction.

```
input  : $m \in \{0,1\}^*$, $\tau \in \mathbb{N}$
output : $\pi \in [0, N-1]$, $l$ prime $\in [0, 2^{2k} - 1]$
$x \leftarrow H(m)$
$y \leftarrow x$
for $k \leftarrow 1$ to $\tau$ do
    $y \leftarrow y^2 \mod N$
end
$l \leftarrow H_{prime}(x+y)$
$\pi = x^{\lfloor 2^\tau / l \rfloor} \mod N$
return $(\pi, l)$
```

2.2.4 Verification

A verifier takes as an input (m, τ, l, π) and computes $x = H(x)$ and $r = 2^\tau \mod l$ and then $y' = \pi^l \cdot x^r \mod N$ and finally checks whether $H_{prime}(x + y') = l$. This operation takes a time λ^4 and is thus independant of T. In Algorithm 2 we present a pseudo code for this phase.

Algorithm 2 Verification of the Wesolowski VDF.

```
input  : $x, \tau, \pi, l$
output : $\top$ or $\bot$
$x \leftarrow H(m)$
$r \leftarrow 2^\tau \mod l$
$y \leftarrow \pi^l \cdot x^r \mod N$
if $l = H_{prime}(x + y)$ then
    return $\top$
else
    return $\bot$
end
```

2.2.5 Overhead on the network

The output size of a VDF can be a critical matter in network considerations. Fortunately, Wesolowski's VDF has a tiny footprint. An evaluation output is composed of elements of the RSA group π which is at most λ bits long and a prime number of size at most $2 \cdot k$.

2.3 Simple VDF - Pietrzak

2.3.1 Setup

The setup part of the Pietrzak's VDF is the same the Wesolowski's one.

2.3.2 Evaluation

The evaluation part is also similar, i.e., computing $y = x^{2^\tau} \mod N$.

2.3.3 Proof

We assume $\tau = 2^t$ for the sake of simplicity. Computing the proof uses some variables defined as following. We set $(x_1, y_1) := (x, y)$ and for $i \in [1, t]$,

$$\begin{aligned} \mu_i &:= x_i^{2^{\tau/2^i}} \mod N \\ r_i &:= H(x_i + y_i + \mu_i) \\ x_{i+1} &:= x_i^{r_i} \cdot \mu_i \mod N \\ y_{i+1} &:= \mu_i^{r_i} \cdot y_i \mod N \end{aligned}$$

The proof is thus $\pi = \{\mu_i\}_{i \in [1,t]}$. One can see it is heavier than the Wesolowski's VDF as there are $\log(T)$ numbers of size λ to transmit, which can be about 40KB in usual conditions. This step has a complexity of $\frac{2\tau}{s\sqrt(\tau)}$ with s being the amount of processors. In Algorithm 1 we show pseudocode to compute the verification and proof of the Pietrzak's VDF.

Algorithm 3 Evaluation and proof of the Pietrzak construction.

```
input  : $m \in \{0,1\}^*$, $\tau \in \mathbb{N}$
output : $\pi \in [0, N-1]^t$
$t \leftarrow \lfloor \log_2(\tau) \rfloor$
$x \leftarrow H(m)$
$y \leftarrow h$
for $k \leftarrow 1$ to $\tau$ do
    $y \leftarrow y^2 \bmod N$
end
$(x_1, y_1) \leftarrow (x, y)$
for $i \leftarrow 1$ to $t$ do
    $\mu_i \leftarrow x_i^{2^{\tau/2^i}}$
    $r_i \leftarrow H(x_i + y_i + \mu_i)$
    $x_{i+1} \leftarrow x_i^{r_i} \cdot \mu_i \bmod N$
    $y_{i+1} \leftarrow \mu_i^{r_i} \cdot y_i \bmod N$
end
return $\pi \leftarrow \{\mu_i\}_{i \in [1,t]}$
```

2.3.4 Verification

In the verification part, the verifier will parse π and check that each element is in the SA group $\frac{\mathbb{Z}}{N\mathbb{Z}}$. If so, it will recompose x_{t+1} and y_{t+1} in the same way as the prove part by computing

$$\begin{aligned} r_i &:= H(x_i + y_i + \mu_i) \\ x_{i+1} &:= x_i^{r_i} \cdot \mu_i \bmod N \\ y_{i+1} &:= \mu_i^{r_i} \cdot y_i \bmod N \end{aligned}$$

and then check that

$$y_{t+1} \overset{?}{=} x_{t+1}^2 \bmod N \tag{1}$$

and output $\top$ if it holds or $\perp$ otherwise.

This step can be achieved with a complexity of $\log(T)$. In the Algorithm 4 we present a pseudocode computing this verification.

2.4 Dimitrov's multiexponentiation

The Wesolowski's construction operates double exponentiation in the verification phase when computing $\pi^l \cdot x^r \bmod N$. The use of a modular multiexponentiation algorithm is then profitable. A multiexponentiation is computing $x^a \cdot y^b \mod n$. However, computing x^a mod , $y^b \mod n$, and then their product, even with the optimal algorithms available, is not the optimal way to do so [13]. Thus, Dimitrov et al. [13] proposed two algorithms to compute multiexponentiations with better average performances. It presents two versions, a first one performing similarly to the *binary exponentiation method* [17] and a second one very lookalike but using recoding to reduce the number of multiplications. The first algorithm can be found in Algorithm 5. Computing the two separate exponentiations should require in average $2 \log 2(\max(a, b))$ multiplications while the Dimitrov's algorithms claims to require in average $\frac{7}{4} \log 2(\max(a, b))$ multiplications.

Algorithm 4 Verification of the Pietrzak construction.

```
input  : $x, \tau, \pi$
output : $\top$ or $\bot$
for $\mu$ in $\pi$ do
    if $\mu >= N$ then
        return $\bot$
end
$(x_1, y_1) \leftarrow (x, y)$
for $i \leftarrow 1$ to $t$ do
    $r_i \leftarrow H(x_i + y_i + \mu_i)$
    $x_{i+1} \leftarrow x_i^{r_i} \cdot \mu_i \bmod N$
    $y_{i+1} \leftarrow \mu_i^{r_i} \cdot y_i \bmod N$
end
if $y_{t+1} = x_{t+1}^2$ then
    return $\top$
else
    return $\bot$
end
```

3 Simulation results

We present now the implementation results of the Wesolowski and Pietrzak VDFs to get real-life estimations. We run the simulations for values of τ between 2^{16} and 2^{20} which requires an evaluation time in the order of seconds or minutes. We have also studied the influence of the RSA modulus bit length λ on the performances and used values in $\{512, 1024, 2048\}$. All the x-axes are in $\log_2$-scale if not specified otherwise.

3.1 Evaluation time analysis

In Figure 1, we present the evaluation time for Pietrzak and Wesolowski VDFs with the x-axis in linear scale. We have identical values for Pietrzak and Wesolowski VDFs. We can see a clear linear growth allowing an easy tuning of the difficulty. It is important to see the clear impact of the RSA group's size as it yields a great variation.

3.2 Proof time analysis

In Figure 2, we show the proving time for Pietrzak and Wesolowski VDFs. They are of the same order of the evaluation which is a serious drawback, forcing users to spend more time after having evaluated the VDF. Reducing the proof computation time is one of the important goals of both constructions. As predicted in the theoretical part, Pietrzak VDF achieves better timing.

3.3 Verification time analysis

In Figure 3, we display the verification time for both constructions. We can see that the Pietrzak construction achieves timing under 1 ms even with 2048 bits RSA groups. However, it is more complicated for the Wesolowski VDF which achieves good timing only for RSA groups with a bitlength under 1024. We can also see the dependence on τ for the Pietrzak verification time grows in $O(\log \tau)$ where the Wesolowski one is independent of τ.

Algorithm 5 Dimitrov multiexponentiation algorithm.

```
For $n \in \mathbb{N}$, we have $\{n_i\}_{i \in \mathbb{N}}$ such as $n = \sum_{i=0}^{\infty} n_i 2^i$ and $n_i = 0 \ \forall \ i > \lfloor \log_2(n) \rfloor$
input  : $x, y, a, b, N$
output : $x^a \cdot y^b \mod N$

$h := \texttt{max}(\lfloor \log_2 a \rfloor + 1, \lfloor \log_2 b \rfloor + 1)$
$z = 1$
$q = x \cdot y \mod N$

for $i = h - 1$ down to 0 do
    $z := z * z \mod N$
    if $a_i = 1$ and $b_i = 0$ then
        $z := z * x \mod N$
    else if $a_i = 0$ and $b_i = 1$ then
        $z := z * y \mod N$
    else if $a_i = 1$ and $b_i = 1$ then
        $z := z * q \mod N$
end
return z
```

It is here interesting to see the impact of the security parameter k in the verification time. We run for a 2048 bits RSA group different values of k in $\{96, 128, 160\}$ for the Wesolowski VDF, and observed the effect. We can see that choosing the right value for k plays a great role in the verification time. The plot can be found in the Figure 4.

4 VDF optimizations

In the Wesolowski construction, there is a need for the evaluator and the verifier to compute a multiexponentiation, $\pi^l x^r \mod N$. Using the Dimitrov's multiexponentiation could speed up this part. We can split the computing time of this algorithm into two main parts. On the one hand, the actual modular multiplication which is uncompressible and we rely on the arbitrary precision library, *NTL* [25] to be as fast as possible on these operations. On the other hand, we have the whole bit scanning process which determines which multiplication to make and which can be optimized. We have tried several implementations to attempt optimizing performances and address the fact that we evaluate the *most significant bits* first when multiplying and it is not trivial to find them efficiently.

4.1 Techniques used

4.1.1 Using strings

The original Dimitrov's algorithm [13] requires to find the binary representation g of a and b in the Pekmestzi's "binary-like" complex representation [20] which is actually interlacing a and b's binary representations such as $g_{2k+1} g_{2k} = a_k b_k \forall \in \mathbb{N}$ (these are not numbers products but strings concatenations). The algorithms will scan g by packets of 2 bits in order to make multiplications.

We propose a slight improvement here which bypasses the computation of g but directly tests the value of a_i and b_i without using g. This is what we wrote in Algorithm 5.

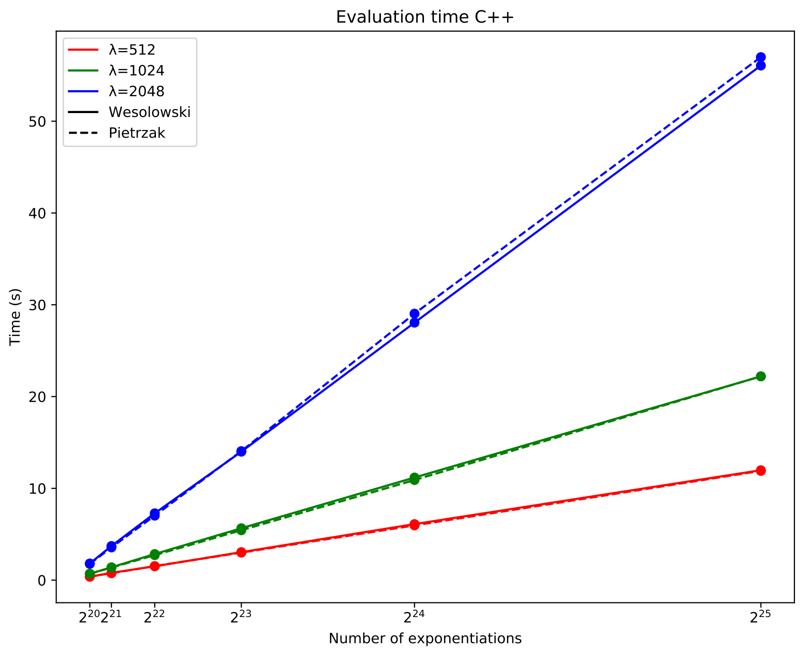


Figure 1 Evaluation time for Pietrzak and Wesolowski VDFs in C++.

4.1.2 Using powers of 4

As the `std::string` C++ structure is quite low when compared to numbers manipulation, we suggest to make use of numbers and consider g as a number such as $g = \sum_{k \in \mathbb{N}} g_k 4^k$ with $g_k = a_k + 2b_k$. It is easy to see that this form covers exactly the four cases encountered when multiplying. Thus we only need to make divisions by 4^i and a modulus 4 to get the values g_i for $i \in \mathbb{N}$. Then we multiply accordingly to the value of g_i.

4.1.3 Using chunked numbers

The problem with the latter improvement is that with big exponents, g can become huge, and then we are forced to consider it as a *NTL* arbitrary precision number and it slows down the execution and the multiple divisions by 4^i require an increasing time as the exponent grows. So we tried to split g in multiple chunks of a fixed size (typically some `unsigned int` or `unsigned char`) and store them. However, as we wanted to have an algorithm working with exponents of unknown size, we need to use a dynamically growing container such as the `std::vector`.

4.1.4 Using chained lists

In our last improvement, we tried to optimize even more this process by using chained lists to get rid of the `std::vector`. We believed this would have better performances because we get rid of the `std::vector` overhead and starting from the *most significant bit* is not a problem anymore because the current pointer points to the *most significant* chunk.

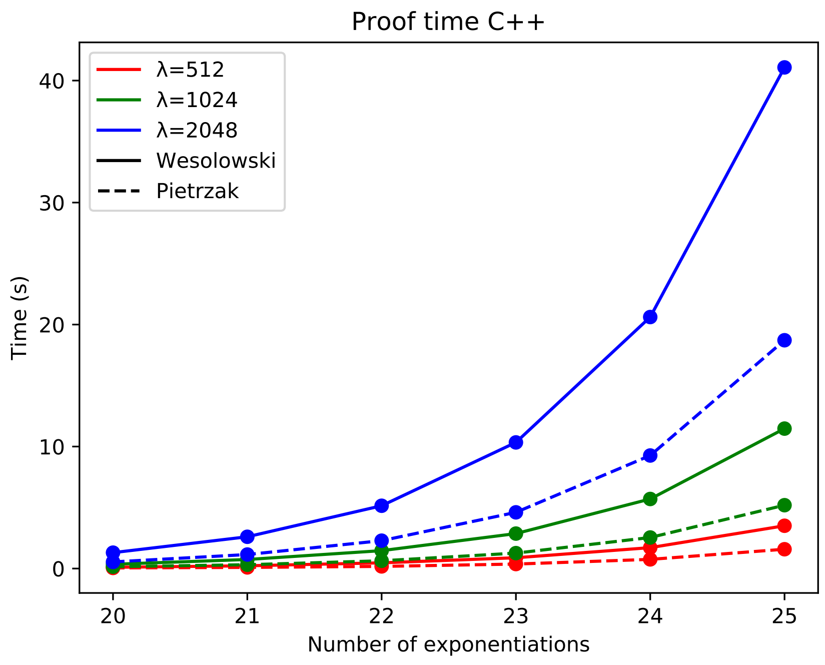


Figure 2 Proof time for Pietrzak and Wesolowski VDFs in C++.

4.2 Multiexponentiation performances

We have run the various optimizations proposed for multiexponentiation and compared them with the naive separate exponentiations using NTL. As expected, we can see that the strings method is the slowest, followed by the power of 4 one. However, using chained lists is slower than chunks contrary to our expectations. This can be explained by the use of memory allocations for each new element of the chained list. We could improve this by allocating multiple elements at the same time to reduce the overhead. However, despite all our efforts, we were unable to compete with the built-in separate exponentiations. This can be explained by the thin theoretical improvement of Dimitrov's multiexponentiation which is only $\frac{8}{7}$ times as fast. Furthermore, the NTL library is optimized to the bones so competing with it requires more advanced programming skills.

4.3 Exponentiation with constant radix in Pietrzak VDF

In this construction, almost every exponentiation which is computed in actually an exponentiation in radix x, the input value. Pietrzak [21] provides an example of how to leverage this property to improve performances but we extend this idea. This optimization is merely identical in the evaluation and the verification parts. We use a logarithmic notation meaning that for a number z, we define $\overline{x^z} = z$. Then we have the recursive definition for the following values

$$\overline{\mu_i} = 2^{\frac{T_i}{2}} \cdot \overline{x_i}$$

$$\overline{x_{i+1}} = r_i \cdot \overline{x_i} + \overline{\mu_i}$$

$$\overline{y_{i+1}} = r_i \cdot \overline{\mu_i} + \overline{y_i}$$

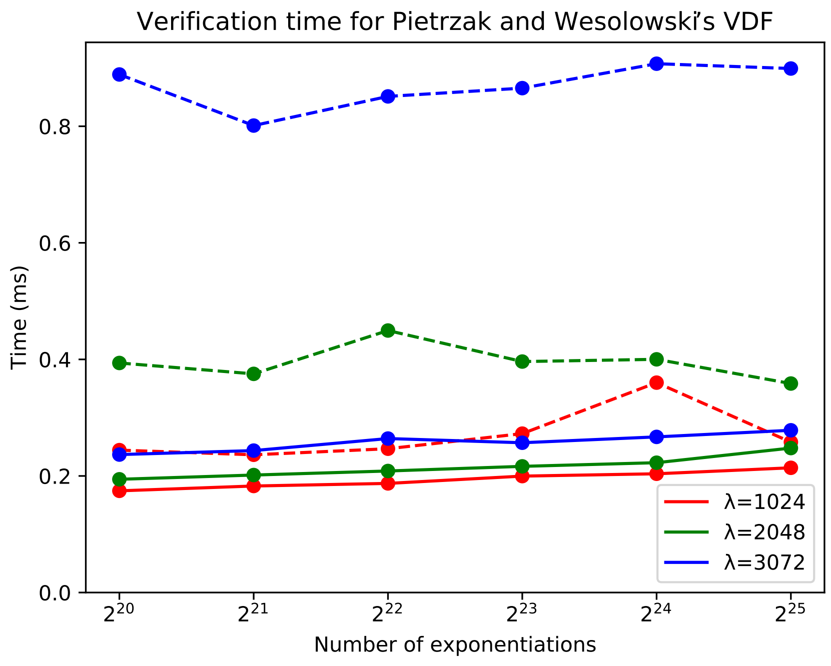


Figure 3 Verification time for Pietrzak and Wesolowski VDFs in C++.

It is then straightforward to show that the values μ_i', μ_i and x_i are exponentiations of x. Then keeping a trace of these values allows quick exponentiation when using precomputations however one should be aware that without precomputations it does not bring any improvement. Besides, this technique has a low memory consumption as we do not need to store the different values x_i and y_i because of the iterative aspect of the calculation. We have not implemented this technique yet and it would be interesting to see if we have a sensible performance improvement.

5 Discussion

5.1 Comparison between Pietrzak and Wesolowski VDFs

The simulations of Pietrzak and Wesolowski's constructions give a clear advantage for Pietrzak's one in terms of the performance of the verification step. However, we have to take into account the overhead induced in the network. As Pietrzak's proof consists of $\log(\tau)$ elements of the RSA group, the verifier should transmit $\log(t) + 1$ elements of $2k$ bits which can represent an overhead of 40KB only for the rate control protocol. This is not viable for DLT protocols, where transaction size must be small to optimize network throughput. Conversely, the Wesolowski's proof is a single element of the RSA group and a small prime number thus the overhead represents only a few KB.

5.2 The RSA modulus

The RSA modulus is a fundamental parameter in the Pietrzak and Wesolowski's constructions, and keeping its factorization is the heart of their security. Indeed, someone knowing its factors p and q can easily compute the Euler totient function $\phi(N) = (p-1)\cdot(q-1)$ and then compute the evaluation in a quasi-instant time because $x^{2^\tau} \bmod N = x^{2^\tau \bmod \phi(N)} \bmod N$.

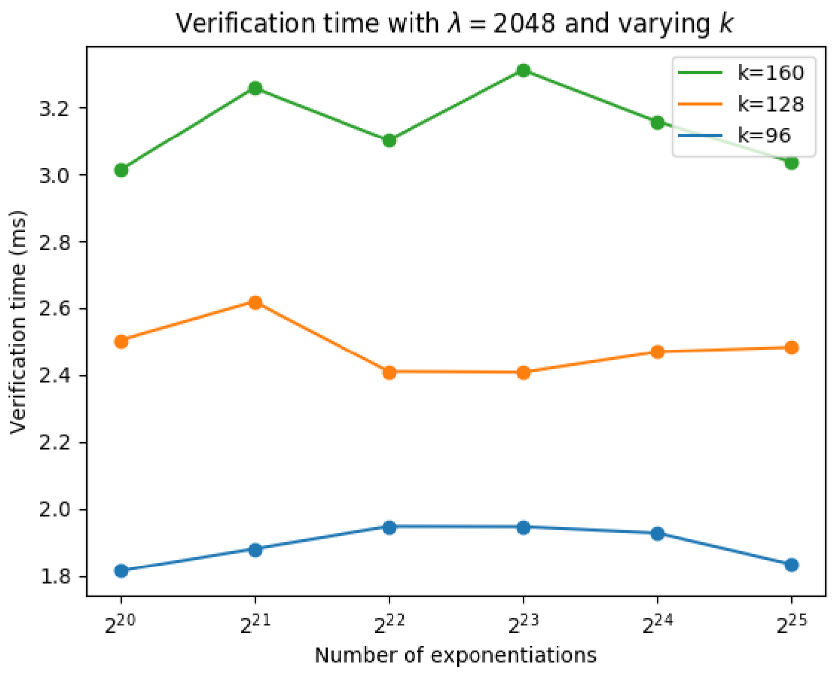


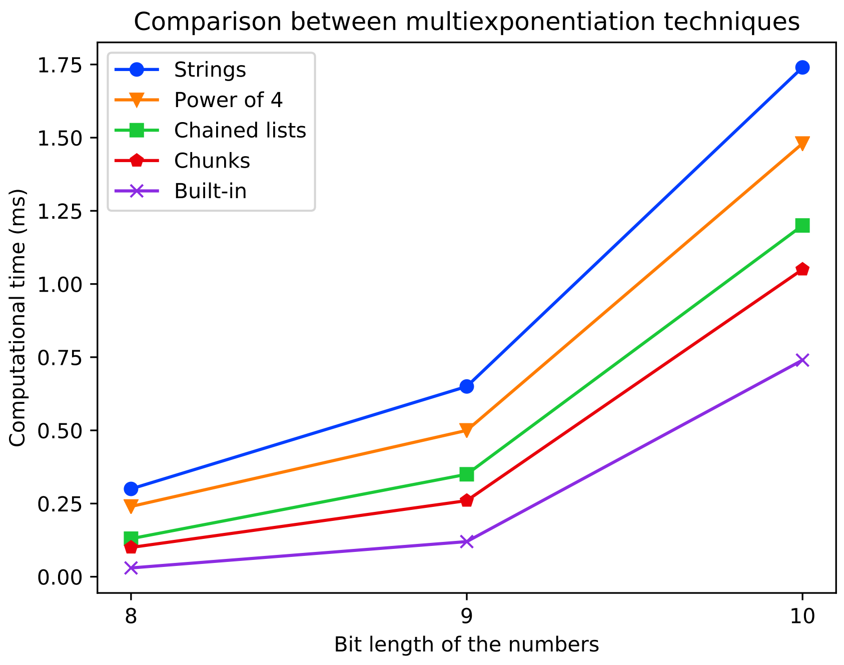
Figure 4 Verification time Wesolowski VDF when varying k with $\lambda = 2048$.

The factorization hardness of an RSA modulus is directly related to its bit length. In Table 1, we present the equivalency between RSA key length and its bit-level security. A k bit-level security means that it takes around 2^k operations to break it. It is estimated that 112-bits security is sufficient until 2030 [10] but for further uses a 128 bit-level security should be chosen. We then suggest using a 2048 bit long RSA modulus for VDF use.

Table 1 Equivalency between RSA keys length and bit-level security [19].

RSA key length	Bit-level security
1024	80
2048	112
3072	128
7,168	192
15,360	256

Finally, a crucial point of using VDFs for DLT applications, especially for permissionless technologies, is how to generate such a modulus in a decentralized way and guarantee that no one can retrieve the factorization, even the nodes having participated in its generation. The distributed generation of RSA keys has first been studied by Boneh and Franklin [8], and its security and performances have been then improved by [1, 11, 23, 9, 6]. The most recent algorithm designed by Frederiksen et al. guarantees a $(n-1)$ security (i.e., at least one of the participants is honest and follows the rules) with very fast performances [16].



0

Figure 5 Multiexponentiation improvements comparison.

Table 2 Performance comparison between PoW and VDF.

Hardware	**PoW** Hash/s (speedup factor)		**VDF** Squaring/s (speedup factor)	
CPU	10^4	$(\times 1)$	10^6	$(\times 1)$
FPGA	10^{10}	$(\times 10^6)$	$3 \cdot 10^7$	$(\times 30)$
ASIC	10^{12}	$(\times 10^8)$	n/a	

5.3 Comparison with Proof of Work

As mentioned in Section 1, VDFs can be used as a non-parallelizable PoW. Therefore, it is interesting to see the VDF performance when using highly specialized hardware such as FPGAs or ASICs which normally speed the PoW computation up to several orders of magnitude. In Table 2, we have collected the potential performance increase between PoW and VDF for different hardware [5]. In Table 3, we have estimated the potential bandwidth occupation when hardware generates IOTA transactions [22] which are encoded in 1.6KB. The difference between PoW and VDF is clear: the intrinsically not parallelizable nature of VDFs allows us to keep a very low throughput for a node even with an FPGA hardware. Currently, no ASICs are available for the computation of VDFs.

6 Conclusion

In this paper, we have presented an analysis of two VDFs, Wesolowski and Pietrzak's constructions, both based on exponentiations in RSA groups. Our work focused on simulating the computation of such VDFs to study the viability of their use in rate control for DLTs. We

Table 3 Spamming potential comparison between PoW and VDF.

Hardware	PoW	VDF
CPU	1.4 kbps	1.9 kbps
FPGA	1.3 Gbps	58 kbps
ASIC	120 Gbps	*n/a*

have seen that, although the Pietrzak's construction has better performances, its overhead on the network makes it not viable in certain contexts. Hence, the Wesolowski's one makes a better candidate, although its verification times is larger. Besides, we have suggested using multiexponentiation algorithms to compute faster Wesolowski's verification.

References

1 Joy Algesheimer, Jan Camenisch, and Victor Shoup. Efficient computation modulo a shared secret with application to the generation of shared safe-prime products. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 2442, pages 417–432, 2002. URL: https://eprint.iacr.org/2002/029.pdf.

2 Adam Back. Hashcash - A Denial of Service Counter-Measure. Technical Report August, Hashcash, 2002.

3 Juan Benet, David Dalrymple, and Nicola Greco. Proof of Replication. Technical report, Stanford University, 2017. doi:10.1007/s00221-007-1153-3.

4 Nir Bitansky, Shafi Goldwasser, Abhishek Jain, Omer Paneth, and Vinod Vaikuntanathan. Time-lock puzzles from randomized encodings. In *ITCS 2016 - Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 345–356, 2016. doi:10.1145/2840728.2840745.

5 Bitcoin Wiki. Mining hardware comparison - Bitcoin Wiki. URL: https://en.bitcoin.it/wiki/Mining_hardware_comparison.

6 Simon R. Blackburn, Mike Burmester, StevenD. Galbraith, and Simon Blake-Wilson. Weaknesses in Shared RSA Key Generation Protocols. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 1746, pages 300–306. Springer, Berlin, Heidelberg, December 1999. URL: http://link.springer.com/10.1007/3-540-46665-7_34.

7 Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10991 LNCS, pages 757–788, 2018. URL: https://eprint.iacr.org/2018/601.pdf.

8 Dan Boneh and Matthew Franklin. Efficient generation of shared RSA Keys. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1294(4):425–439, July 1997. doi:10.1007/BFb0052253.

9 Clifford Cocks. Split knowledge generation of RSA parameters. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 1355, pages 89–95, 1997. doi:10.1007/bfb0024452.

10 Cybernetica. Cryptographic Algorithms Lifecycle Report 2016. Technical report, Cybernetica, 2016. URL: https://www.ria.ee/sites/default/files/content-editors/publikatsioonid/cryptographic_algorithms_lifecycle_report_2016.pdf.

11 Ivan Damgård and Gert Læssøe Mikkelsen. Efficient, robust and constant-round distributed RSA key generation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5978 LNCS:183–200, 2010.

12 Luca De Feo, Simon Masson, Christophe Petit, and Antonio Sanso. Verifiable delay functions from supersingular isogenies and pairings. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11921 LNCS, pages 248–277, 2019. URL: `https://defeo.lu/`.

13 Vassil S. Dimitrov, Graham A. Jullien, and William C. Miller. Complexity and fast algorithms for multiexponentiations. *IEEE Transactions on Computers*, 49(2):141–147, 2000. `doi:10.1109/12.833110`.

14 Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9216, pages 585–605, 2015. URL: `https://eprint.iacr.org/2013/796.pdf`.

15 Ben Fisch, Joseph Bonneau, Nicola Greco, and Juan Benet. Scaling Proof-of-Replication for Filecoin Mining. Technical report, Stanford University, 2018. URL: `https://web.stanford.edu/{~}bfisch/porep_short.pdf`.

16 Tore Kasper Frederiksen, Yehuda Lindell, Valery Osheter, and Benny Pinkas. Fast distributed rsa key generation for semi-honest and malicious adversaries. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10992 LNCS, pages 331–361. Springer Verlag, 2018.

17 C. K. Koc. High-speed RSA implementation. Technical report, RSA Laboratories, 1994. URL: `ftp://ftp.rsasecurity.com/pub/pdfs/tr201.pdf`.

18 Arjen K. Lenstra and Benjamin Wesolowski. Trustworthy public randomness with sloth, unicorn, and trx. *International Journal of Applied Cryptography*, 3(4):330–343, January 2017. `doi:10.1504/IJACT.2017.089354`.

19 NIST. Recommendation for Key Management - Part 1: General. *NIST Special Publication 800-57*, Revision 3(July):1–147, January 2012. `doi:10.6028/NIST.SP.800-57pt1r4`.

20 K. Z. Pekmestzi. Complex number multipliers. *IEE Proceedings E: Computers and Digital Techniques*, 136(1):70–75, 1989. `doi:10.1049/ip-e.1989.0010`.

21 Krzysztof Pietrzak. Simple verifiable delay functions. In *Leibniz International Proceedings in Informatics, LIPIcs*, volume 124, 2019. `doi:10.4230/LIPIcs.ITCS.2019.60`.

22 Serguei Popov. The Tangle. Technical report, IOTA Foundation, 2018. URL: `https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf`.

23 Michael O. Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12(1):128–138, 1980. `doi:10.1016/0022-314X(80)90084-0`.

24 R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. `doi:10.1145/359340.359342`.

25 Victor Shoup. NTL - A Library for doing Number Theory, 2019. URL: `https://www.shoup.net/ntl/`.

26 Benjamin Wesolowski. Efficient verifiable delay functions. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11478 LNCS, pages 379–407, 2019. URL: `https://eprint.iacr.org/2018/623.pdf`.

Revisiting the Liquidity/Risk Trade-Off with Smart Contracts

Vincent Danos
CNRS, École Normale Supérieure, PSL, INRIA, Paris, France
vincent.danos@ens.fr

Jean Krivine
CNRS, Université de Paris, France
jean.krivine@irif.fr

Julien Prat
CNRS, École Polytechnique, Paris, France
Julien.Prat@ensae.fr

Abstract

Real-time financial settlements constrain traders to have the cash on hand before they can enter a trade [3]. This prevents short-selling and ultimately impedes liquidity. We propose a novel trading protocol which relaxes the cash constraint, and manages chains of deferred payments. Traders can buy without paying first, and can re-sell while still withholding payments. Trades naturally arrange in chains which contract when deals are closed and extend when new ones open. Default risk is handled by reversing trades.

In this short note we propose a class of novel financial instruments for zero-risk and zero-collateral intermediation. The central idea is that bilateral trades can be chained into *trade lines*. The ownership of an underlying asset becomes distributed among traders with positions in the trade line. The trading protocol determines who ends up owning that asset and the overall payoffs of the participants. Counterparty risk is avoided because the asset itself serves as a collateral for the entire chain of trades. The protocol can be readily implemented as a smart contract on a blockchain.

Additional examples, proofs, protocol variants, and game-theoretic properties related to the order-sensitivity of the games defined by trade lines can be found in the extended version of this note [1]. Therein, one can also find the definition and game-theoretic analysis of standard trade-lines with applications to trust-less zero-collateral intermediation.

2012 ACM Subject Classification Information systems → Online banking

Keywords and phrases Electronic trading, Smart contracts, Static analysis

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2020.10

Category Short Paper

1 Bilateral contracts

A bilateral contract between traders u, v is a set of clauses describing their time-dependent payment and delivery obligations.

An *atomic clause* for x in $\{u, v\}$ is a triple (a, b) in $\mathbb{R}_+ \times \mathbb{R}$, where $a \geq 0$ is the *activation payment*, b the *passive effect*, and x is the *active trader*. The first component a specifies the amount x needs to pay to trigger the clause. The second component b is the payment which ensues, with the convention that $b \geq 0$ if x receives the payment.

A *constant clause* is a disjunction of atomic ones ie an element of $\mathcal{P}_{\text{fin}}(\mathbb{R}_+ \times \mathbb{R})$.

A *clause* is a finite piecewise-constant function from $\mathbb{Z}$ to $\mathcal{P}_{\text{fin}}(\mathbb{R}_+ \times \mathbb{R})$.

Time corresponds to block number in a blockchain implementation.

The *domain* of a clause is $|\Theta| = \{t \mid |\Theta(t)| > 0\}$.

We write $\mathcal{C} \subseteq \mathbb{Z} \to \mathcal{P}_{\text{fin}}(\mathbb{R}^2_+)$ for the set of clauses.

We say a clause Θ is: *deterministic*, if $|\Theta(t)| \leq 1$; *eventually defined* if $[M, +\infty) \subseteq |\Theta|$ for some M. We write $\mathcal{C}_0$, $\mathcal{C}_e \subset \mathcal{C}$ for the associated sets.

Clauses can be merged using two operations: $(\Theta + \Theta')(t) = \Theta(t) + \Theta'(t)$ where the sum on the right is taken pairwise $\{\theta_i\} + \{\theta'_j\} = \{\theta_i + \theta'_j\}$; and $(\Theta \vee \Theta')(t) = \Theta(t) \cup \Theta'(t)$.

The triple $(\mathcal{C}, \vee, +)$ is a commutative idempotent semi-ring, with sub-semi-ring the set of eventually defined clauses (but not the deterministic ones $\mathcal{C}_0$). The domain map is a semi-ring morphism from $(\mathcal{C}, \vee, +)$ to $(\mathcal{P}_{\text{fin}}(\mathbb{Z}), \cup, \cap)$.

A *bilateral contract* between u and v consists of a *backward clause* β for u, and a *forward clause* ϕ for v. One can depict a contract as a game (Fig. 1). The forward/backward distinction only makes sense when contracts are composed (see next Section).

A contract ϕ, β between u and v is said to be: *exclusive* if $|\phi| \cap |\beta| = \emptyset$; in an *$F$-state* (*$B$-state*) if active payments requested by ϕ (β) have been made by v (u); *idle* if neither in a B-state or an F-state; *eventually-F* (B) if ϕ (β) is eventually defined.

Notice that the game tree is not strictly speaking that of a sequential game [2]: the availability of moves to either player is context-dependent and so are the payoffs; and moves may (depending on the context) be available simultaneously to both players.

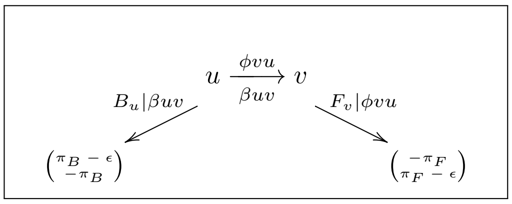

$u \xrightarrow[\beta uv]{\phi vu} v$
$B_u|\beta uv$
$F_v|\phi vu$
$\binom{\pi_B - \epsilon}{-\pi_B}$
$\binom{-\pi_F}{\pi_F - \epsilon}$

Figure 1 A contract as a game: ϵ is the constant cost of a move (gas); π_B (π_F) is the aggregated payoff to u (v); move B_u (F_v) is available in contexts where β (ϕ) holds.

A basic example is the *standard bilateral contract* with forward and backward clauses $\phi(t) = t < \Sigma \mapsto (a, 0)$, and $\beta(t) = t \geq \Delta \mapsto (0, p)$. This means that v may buy u's asset at price a at any time $t < \Sigma$; while u may cancel the deal at any time $t \geq \Delta$ and be paid a penalty p by v. One has $|\phi| = (-\infty, \Sigma)$, $|\beta| = [\Delta, +\infty)$. Hence this contract is: eventually-B; idle during the $[\Sigma, \Delta)$ interval if $\Sigma < \Delta$; and exclusive iff $\Sigma \leq \Delta$.

Absent the property of exclusivity, traders may move simultaneously. As a consequence, in a blockchain implementation, outcomes may depend on the order in which moves are ordered by the block-makers. A form of order-insensitivity can be achieved (see Supp. Inf. [1]).

2 Composite contracts

A *tradeline* is a non-empty list of composable contracts:

$$u_1 \xrightarrow[\beta_1]{\phi_1} u_2 \xrightarrow[\beta_2]{\phi_2} \cdots\ u_{n-1} \xrightarrow[\beta_{n-1}]{\phi_{n-1}} u_n$$

u_1 is called the *origin* and u_n the *end* of the trade line. If $u_1 = u_n$, one says the trade line is *resolved*. If, in a given context, no contract is in an F- or a B-state, one says the trade line is *irreducible*; if every contract is in an F- or a B-state, one says the trade line is *connected*.

A trade line represents a game being played between the owners of its positions. We distinguish two types of moves.

Contractions are moves whereby the owner of an active position acquires an adjacent position. Below the active (acquired) position is in red (blue). Boundary cases where v is the end of the trade line (indicated by the symbol $v\,\cdot$) are shown on the right. Payments consequent to clauses being triggered are left implicit.

$$\begin{tikzcd}
u \arrow[r, "\phi", "\beta"'] & v \arrow[r, "\phi'", "\beta'"'] & w \arrow[r, Rightarrow, "F_v", "\phi"'] & v \arrow[r, "\phi+\phi'", "\beta\vee\beta'"'] & w & u \arrow[r, "\phi", "\beta"'] & v \cdot \arrow[r, Rightarrow, "F_v", "\phi"'] & v \cdot \\
u \arrow[r, "\phi", "\beta"'] & v \arrow[r, "\phi'", "\beta'"'] & w \arrow[r, Rightarrow, "B_u", "\beta"'] & u \arrow[r, "\phi+\phi'", "\beta\vee\beta'"'] & w & u \arrow[r, "\phi", "\beta"'] & v \cdot \arrow[r, Rightarrow, "B_u", "\beta"'] & u \cdot
\end{tikzcd}$$

If the trade line fully resolves under contraction, the owner of the last remaining position is now in full possession of the underlying asset.

The idea adding forward clauses is that the payment just made (or once promised) by v is transferred onto w's forward clause. The idea of taking the union of backward clauses is that u is carrying over his original cancellation condition in the new contract with w. Specifically, the new backward clause $\beta \vee \beta'$ which ties in u and w is implied by β. This means that the player triggering the B-move does not have to stop after the first contraction and *can sweep the entire trade line* if she wishes to, for as long as β holds.

Let γ, γ' be trade lines, and suppose γ contracts to γ' under the rules above. The following holds: (i) if γ is connected, so is γ'; (ii) if γ is exclusive, so is γ'; (iii) if all arcs in γ are eventually-B or -F, so are the ones in γ'.

We can return to the standard contracts and compute their reductions. With simplified and self-evident notations we get the following contraction rules:

$$\begin{tikzcd}
u \arrow[r, "{\Sigma_1, a_1}", "{\Delta_1, p_1}"'] & v \arrow[r, "{\Sigma_2, a_2}", "{\Delta_2, p_2}"'] & w \arrow[r, Rightarrow, "B_u", "v \to^{p_1} u"'] & u \arrow[r, "{\min(\Sigma_1,\Sigma_2),\, a_1+a_2}", "{\Delta_1,\, p_1 \vee \Delta_2,\, p_2}"'] & w & \textit{for } t \geq \Delta_1 \\
u \arrow[r, "{\Sigma_1, a_1}", "{\Delta_1, p_1}"'] & v \arrow[r, "{\Sigma_2, a_2}", "{\Delta_2, p_2}"'] & w \arrow[r, Rightarrow, "F_v", "v \to^{a_1} u"'] & v \arrow[r, "{\min(\Sigma_1,\Sigma_2),\, a_1+a_2}", "{\Delta_1,\, p_1 \vee \Delta_2,\, p_2}"'] & w & \textit{for } t < \Sigma_1
\end{tikzcd}$$

For $t \geq \max(\Delta_1, \Delta_2)$ the trader with the backward clause will pick the highest of p_1 or p_2.

2.1 Extension

We also need a move to grow a trade line. Any trader at the end u of the trade line can extend it using the sell rule $S_{uv}(\phi, \beta)$ and append a new contract with clauses ϕ, β, thereby adding a new position v to the game:

$$\begin{tikzcd}
u \cdot \arrow[r, Rightarrow, "{S_{uv}(\phi,\beta)}"] & u \arrow[r, "\phi", "\beta"'] & v \cdot
\end{tikzcd}$$

If extensions were not constrained to happen the end of the trade line, the owner of the origin could introduce a new position u_0 left of it:

$$\begin{tikzcd}
u_0 \arrow[r, "-", "{-\infty < t \mapsto (0,a)}"'] & u_1 \arrow[r] & \cdots \arrow[r] & u_n
\end{tikzcd}$$

and sweep through the entire line by iterating B_{u_0} and collect an arbitrary fee a from everyone. We show below that our design choices prevent such catastrophic events.

3 Soundness of the trade game

We establish now an upper bound on the expenses incurred by playing the game.

The simplify the derivation we assume deterministic clauses. Let γ be a trade line, and let t be a time. We denote by $<$ the positional ordering in γ. For any position $i \in \gamma$ not at the end of γ, we denote by $\phi_i(t)$, $\beta_i(t)$ the clauses where i is in backward position.

Payoffs at fixed time are specified by pairs of real numbers: $\phi_i^1(t) \geq 0$ is the active payment i's Buyer makes to i to complete the deal; and $\phi_i^2(t)$ is the (possibly negative) passive payment from i to his Buyer which follows. When i's Buyer plays that forward move, i's payoff is therefore: $\phi_i^1(t) - \phi_i^2(t)$. We suppose henceforth that $\phi_i^1(t) \geq \phi_i^2(t)$, so that a forward move is always profitable to the passive player. This constraint is stable under contractions.

Likewise: $\beta_i^1(t) \geq 0$ is the active payment i needs to make to i's Buyer to cancel the deal; and $\beta_i^2(t)$ is the (possibly negative) passive payment from i's Buyer to i which follows. When i plays that backward move, i's payoff is therefore: $\beta_i^2(t) - \beta_i^1(t)$.

Fig. 2 summarises the situation. For F_v, B_v, v is the active trader; for B_u, F_w, v is passive

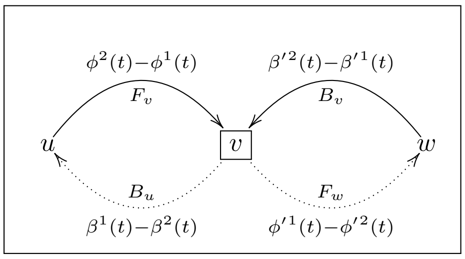


Figure 2 Contractions which change the balance of v together with the subsequent payoffs.

and evicted by the move. Accordingly, from v's viewpoint $\phi^2(t) - \phi^1(t)$, and $\beta'^2(t) - \beta'^1(t)$ are the active payoffs, and $\beta^1(t) - \beta^2(t)$, $\phi'^1(t) - \phi'^2(t)$ the passive ones.

We define the *leftward bounds* (leaving out positions where payoffs are undefined):

$$\begin{array}{lcll}
\beta(v,t) & = & \max_{i<v}(\sup_{s\geq t}(\beta_i^2(s) - \beta_i^1(s))) & \textit{passive expense on a } B\textit{-move} \\
\phi(v,t) & = & \sum_{i<v} \sup_{s\geq t}(\phi_i^1(s) - \phi_i^2(s)) & \textit{active expense on an } F\textit{-move}
\end{array}$$

The idea is that $\beta(v,t)$ is an upper bound for the expenses v may incur upon eviction by a B-move, whichever is the trace followed. Likewise, $\phi(v,t)$ is an upper bound for the price v will ever have to pay to acquire the underlying (by buying all left positions using a series of F-moves). Both quantities depend only on contracts left of v in γ.

We define also the *rightward bounds*:

$$\eta_v^B(t) \quad = \quad \sup_{s\geq t}(\beta_v^1(s) - \beta_v^2(s)) \qquad \textit{active expense on a } B\textit{-move}$$

The idea is that $\eta_v^B(t)$ upper bounds the active payment made by v on a B-move. This bound assumes no trader is fool enough to pick an option worse than his original backward clause. Here the control is local to v, because v's β clause self-propagates under reduction. Note that $\eta_v^B(t) \leq 0$ if v's backward clause always specifies a profit for v. There is no need to define a symmetric $\eta_v^F(t)$ to control for passive expenses on an F-move, as we have assumed above that F-moves are always profitable to the Seller.

▶ **Proposition 1** (max expenses). *Let γ be a trade line, and v a position in γ. Along any trace starting from γ where v plays no extension, v's expenses are upper bounded by:*

$$\phi(v,t) + \beta(v,t) + N\eta_v^B(t)$$

with N the number of B_v moves played by v. If $\eta_v^B(t) \leq 0$, v's expenses are upper bounded by $\beta(v,t) + \phi(v,t)$.

To cope with traces where v extends the trade line, ie plays with moves of type $S_{vw}(\beta_k, \phi_k)$, one can readily adapt $\eta_v^B(t)$ to also maximise over such moves k:

$$\hat{\eta}_v^B(t) \quad = \quad \max_k \sup_{s \geq t}(\beta_k^1(s) - \beta_k^2(s))$$

If all these payments are Seller-positive, ie $\hat{\eta}_v^B(t) \leq 0$, we can forget this additional term.

One can also show that the evolution of a trade line cannot lead to a solution where a Seller receives less than the originally asked price for a forward as well as a backward move.

▶ **Proposition 2** (Monotonicity). *Let γ be a trade line, and v a position in γ: v's forward payoff (as Seller) is non-decreasing, and v's backward payment is invariant.*

Note that even if there are Buyers waiting to join, v's forward payoff may never happen. Suppose v faces a w who extends the tradeline with a *forward-dead* contract $S_{wx}(\mathbf{0}, \beta)$, and w never plays F_w. The only way out for v is to B-sweep the trade line entirely.

4 Aside on implementation

Trade lines and their evolution rules can be interpreted by a dedicated smart contract connected to an external custodial contract to define the ownership of the assets traded in the protocol. When the game starts, the owner of the asset transfers its property to an account of the custodial contract which is controlled by the interpreter contract.

To implement passive payments which are essential to the protocol, one could forward payment obligations to an external system managing the players' debts. Prop. 1 gives a solution for a trust-less implementation. Using the associated upper bounds, the contract can statically compute the amount of cash a trader needs to stake in, upon joining. By asking traders to fully *provision* potential expenses, one does not have to trust them to honour their debts. Depending on specific time-dependencies, provisions can be partially returned as time advances and provisions are re-evaluated. There is no such concern for active payments, as these are payments which players have to make to change the state of the game.

When the trade line finally resolves, it remains for the interpreter contract to ask the custodial contract to transfer the ownership of the asset to the owner of the one remaining position in the game (which implies that the owner is known to the custodial contract).

5 Conclusion

We have defined a consistent trade protocol to manage chains of reversible bilateral contracts. Its design derives entirely from a simple premise: the need for a theory of deferred payments which allows one to postpone payments, and resell an asset one has not paid for yet. To do this one has to keep somehow a memory of past transactions, and introduce mechanics to revert some, as the need may occur. This leads to a protocol where chains of transactions define the state of an open game; the evolution of which relies on the reversibility of the component games.

References

1 Vincent Danos, Jean Krivine, and Julien Prat. Reversible and composable financial contracts, 2019. URL: `http://www.di.ens.fr/~danos/tls.pdf`.

2 Drew Fudenberg and Jean Tirole. Game theory, 1991. *Cambridge, Massachusetts*, 393(12):80, 1991.

3 Mariana Khapko and Marius Zoican. How fast should trades settle? *Society for Financial Studies (SFS) Cavalcade*, 2017.

Proof of Behavior

Paul-Marie Grollemund
Université Clermont Auvergne, LMBP UMR 6620, Aubière, France
paul_marie.grollemund@uca.fr

Pascal Lafourcade[1]
Université Clermont Auvergne, LIMOS UMR 6158, Aubière, France
https://sancy.iut-clermont.uca.fr/~lafourcade/
pascal.lafourcade@uca.fr

Kevin Thiry-Atighehchi
Université Clermont Auvergne, LIMOS UMR 6158, Aubière, France
kevin.atighehchi@uca.fr

Ariane Tichit
Université Clermont Auvergne, Cerdi UMR 6587, Clermont-Ferrand, France
ariane.tichit@uca.fr

Abstract

Our aim is to change the *Proof of Work* paradigm. Instead of wasting energy in dummy computations with hash computations, we propose a new approach based on the behavior of the users. Our idea is to design a mechanism that replaces the Proof of Work and that has a positive impact on the world and a social impact on the behaviors of the citizens. For this, we introduce the notion of *Proof of Behavior*. Based on this notion, we present a new cryptocurrency, called *EcoMobiCoin*, that encourages the ecological behavior in the mobility of the citizens.

2012 ACM Subject Classification Security and privacy

Keywords and phrases Proof of behavior, Blockchain, Security

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2020.11

Category Short Paper

1 Introduction

Bitcoin [12] was the beginning of a digital revolution and it is also the birth of the blockchain technology (see [3] for an overview). The security of this technology relies on the concept of Proof of Work (PoW). In order to validate a transaction, a miner needs to produce a PoW. In Bitcoin, a PoW is the computation of an objective of hash, which is finding a number that satisfies an inequation. Finding this number requires to compute thousands of hash functions. PoW is one of the main negative aspect of this technology since it is highly energy consuming [13]. Moreover in the case of Bitcoin, the performed hash computations are really useless. Our goal is to design an alternative to PoW, for this purpose we introduce the notion of *Proof of Behavior* (PoB).

Contributions. We present the notion of PoB, the idea is to incentivize citizens to have responsible behaviors instead of doing useless computations as in PoW. Our aim is to replace PoW by PoB. We propose a first application to design a new cryptocurrency for the mobility, called *EcoMobiCoin* for Ecological and Collaborative Mobility Coin. If you can prove that you are biking or walking or using public transportation to go somewhere instead of using

[1] Corresponding author

your car, or if you can prove that you are using your car with some passengers to go to somewhere, you are generating a Proof of Behavior for eco-responsible mobility and then creating new *EcoMobiCoins*. This approach aims at facilitating the energy transition that is a key point of the next years.

Related Work. Many works aim at improving existing blockchains or cryptocurrencies as for instance [2, 8, 9, 11, 5]. There are many works that use blockchain to develop new applications as for instance online secure e-voting [7] or online secure e-auction [4] or even proof of identity [10]. Moreover many cryptocurrencies have been designed after Bitcoin, as for example Ethereum, PeerCoin, PrimeCoin etc. In [1], the authors proposed a classification in 4 categories of the existing cryptocurrencies:

1. Scam: These are cryptocurrencies that are designed quickly, not secure and their only goal is to convince people to invest money in these coins in order that the designers earn some money. They are usual quickly identified by the community as *scams* and they disappear.
2. Clone: These cryptocurrencies are just some clones of Bitcoin to particular purpose as for instance PokerCoin for poker players.
3. New goal: Here the aim is to change the goal of the cryptocurrencies, for instance PrimeCoin aims at discovering new Cunningham chains that are mathematical advances in prime numbers. Two other examples are CureCoin or FoldingCoin that aim at using the computation to solve medicine problems.
4. New consensus: The goals of such cryptocurrencies is to propose different consensus. The first initiative was PeerCoin that introduces the notion of *Proof of Stake*. Some other initiatives exist like SpaceMint [14] that introduces the Proof of Space or PermaCoin that introduces the notion of *Proof of retrievability*.

Our concept of Proof of Behavior is clearly at the intersection of the two last categories. We are proposing a new goal and at the same time a new paradigm. The closest existing cryptocurrency to a PoB is SolarCoin[2]. The goal of SolarCoin is to "*incentivize solar electricity by rewarding the generators of solar electricity*". They reward solar energy producers with blockchain-based digital tokens at the rate of 1 SolarCoin (SLR) per 1 MWh of solar energy produced. More precisely, users produce solar energy and provide a proof of this production to the SolarCoin Foundation that approves its behavior. Then users receive SolarCoins and can use them. The experience of SolarCoin started in 2014 clearly shows that it is an economic model that works.

Concerning our application to mobility, the closed project is MobiCoin presented in 2018 at the Mobile World Congress in Barcelona, Spain by Mercedes-Benz to reward conductors that have an ecological drive[3]. They aim at collecting users data and rewarding some of them with Mobicoins. Unfortunately in 2020, this project is not yet used and it is difficult to obtain any information on its status. However our aim is different, since we reward collaborative mobility and zero emission mobility like walking and biking.

Outline. We first explain the concept of Proof of Behavior in the next section. Then we apply PoB to design EcoMobiCoin, before concluding.

[2] `https://solarcoin.org/`
[3] Visited the 18 January 2019,
`https://www.ellulschranz.com/mercedes-benz-invested-blockchain-technology`

2 Proof of Behavior

We first present the idea of PoB, then the differences with PoW and finally the necessary conditions for such system to work.

The idea of Proof of Behavior

The main idea behind PoB is that if users are doing some concrete actions in the real world and they can provide a proof of their actions then these PoB are used to generate new coin.

This is clearly a comeback to the essence of the revolution launched by Bitcoin: a system based on a decentralized, collaborative, distributed consensus to validate transactions and create new coins. Moreover, the main innovation in PoB is that it is not consuming time and energy to useless things.

Comparison

Comparing to the PoW the actions of the users in the real world allow everyone to participate to the coin generation. It is not necessary to spend money in specialized material for mining, as in Bitcoin, since it is user's behavior that gives the power to mine coins. With this change of paradigm everyone can decide to select which behavior he wants to have in order to contribute to a global improvement of the society.

The main difference is that valid proofs of behavior are used to generate new coins. It means that nothing is wasted, because PoB are positive actions for the society, so it does not matter if they are realized but not used to generate coins. It is not necessary that the behavior is more and more difficult according to the number of persons, as in Bitcoin where the system adapts itself in order that only few transactions are validated every ten minutes. This implies in Bitcoin that the cost of the transactions is more and more expensive, because everyone wants to win the race to find the nonce to solve the objective of hash and because the difficulty is increasing. In a proof of behavior any action can contribute to the generation of new coins.

Conditions

In Bitcoin, the revolution comes with a main innovation: decentralization. It means that central entities are not needed anymore to create currencies. It implies that everyone can mine and not only the financial institutions can generate money. A necessary condition in this system is that everyone can also verify the results of the computations of the miners since everything is publicly distributed. The same mechanism is present in PoB: everything is publicly verifiable and written in the blockchain.

The key point is to determine who has the right to write in the blockchain and how? In PoB, this right is not given to miners that have a lot of computational resources as in Bitcoin but it is, in some sense, shared between the three following actors:

User: Person who does some transactions by sending coins to someone.

ProofMaker: Person who realizes a PoB.

Verifier: Person who verifies the validity of PoB and the validity of transactions. Then he writes in the blockchain the verified PoB and transactions.

To summarize, everyone can be a ProofMaker and generate PoB. Everyone can verify the validity of some PoB and then uses these valid PoB to register on the blockchain some valid transactions. To compare with Bitcoin where the miners perform the verification of the validity of the transactions and also the proof of work, we have the verifiers that only verify the validity of the transactions and of the PoB. Moreover, the proof of work are done by the ProofMaker by having positive behaviors.

Moreover we add the fact that a PoB has a validity period. We use the fact that a behavior is something that is done at some precise time, then a PoB has a validity period of few hours (for instance 24h) to be used by a verifier to generate new coins. We can also add such constraints on the transactions, if a transaction is not written in the blockchain after few hours (it can also be for instance 24h) then the transaction is removed from the pool of transactions. Indeed this is implicitly done in Bitcoin. In this setting, in order to validate a transaction, a verifier needs to have verified a proof of behavior and the validity of some transactions.

Concerning the blockchain, we can imagine at least three possibilities:

Private: A consortium of partners like public transportation, cities, government or industrial can just vote or validate the verified associations PoB and transactions.

Public: Every verifier can write in the blockchain, the longest chain having the highest behavior score[4] is the main chain. We also add the fact that all blocks written after 24h in the main chain cannot be changed. This point limits the possibilities of fork.

Hybrid: A mix between public and private blockchain is also possible.

Each time a verifier writes a block, he creates one coin. It is important that this reward remains a constant and depends neither on the transactions nor on the PoB. At the same time, the owners of the PoB used by the verifier also receive one coin that is fairly split between all PoB owners used by the verifier.

The concept of Proof of Behavior is clearly an important innovation toward a new economical system where everyone is responsible of its acts.

3 Application: EcoMobiCoin

One of the first application of PoW is the design of a cryptocurrency to incentivize less emission in the transportation. For this, the first task is to define what are the behaviors that we want to promote. We identify four main behaviors: walking, biking, using public transportation and carpooling.

For each situation a proof of behavior is a real GPS trace that can be collected using a simple smartphone. For this we need a signature of the device that is unique. This is necessary in order that a device can be identified and not be used in several traces at the same time. The trace should also prove that the user was walking or biking or driving. For this some statistical algorithms [6] are used to determine if a user's GPS trace is a valid trace of the following behaviors: walking, biking or driving. These algorithms are public and used by verifiers to determine the trace validity. The verification is part of the work of the verifier and then he can write to the blockchain.

Concerning the public transportation, the proof contains two GPS traces: one for the user and for instance one for the tram line. Here other algorithms are used to prove that the two traces are similar. Finally for the carpooling, a PoB also include several GPS traces. Of course each proof of behavior is awarded by some EcoMobiCoins, so a PKI infrastructure is used to ensure all the cryptographic mechanisms as in any blockchain.

[4] This score depends of which behaviors the cryptocurrency wants to emphasize.

In comparison to other economic systems based on a cryptocurrency, PoB allows to define a range of ways to generate coins. Cryptocurrencies as SolarCoin are focusing on only one behavior or only one small subset of the society. On the opposite, a PoB-based cryptocurrency is affordable to a large part of the population. As a consequence, an economic system based on EcoMobiCoin is more robust and is likely to include a wider public embracing.

4 Conclusion

We change the paradigm of Proof of Work and we introduce the concept of Proof of Behavior. This allows us to incentivize behaviors of users. We propose one first application for transportation with the design of EcoMobiCoin. Many applications can be envisaged based on the notion of Proof of Behavior. We can imagine several other applications in order to reward good usages as soon as it is possible to construct a verifiable proof of behavior. In each application it is important to design adapted cryptographic primitives in order to have a sufficient security level in how the proofs of behavior are produced.

References

1 A. Tichit, P. Lafourcade, and V. Mazenod. Les monnaies virtuelles décentralisées sont-elles des dispositifs d'avenir ? *revue Interventions Economiques*, 2017.

2 E. Anceaume, R. Ludinard, M. Potop-Butucaru, and F. Tronel. Bitcoin a Distributed Shared Register. In *19th Intl. Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, 2017.

3 J-G. Dumas, P. Lafourcade, A. Tichit, and S. Varette. *Les blockchains en 50 Questions, comprendre le fonctionnement et les enjeux de cette technologie innovante.* Dunod, 2018.

4 P. Lafourcade, M. Nopere, J. Picot, D. Pizzuti, and E. Roudeix. Security analysis of auctionity: a blockchain based e-auction. In *12th International Symposium on Foundations and Practice of Security - Revised Selected Papers*, FPS, 2019.

5 I. Abraham, G. G. Gueta, D. Malkhi, M. K. Reiter, and M. Yin. Hot-stuff the linear, optimal-resilience, one-message BFT devil. *CoRR*, abs/1803.05069, 2018. arXiv:1803.05069.

6 P. C. Besse, B. Guillouet, J. Loubes, and F. Royer. Review and perspective for distance-based clustering of vehicle trajectories. *IEEE Transactions on Intelligent Transportation Systems*, 17(11):3306–3317, November 2016. doi:10.1109/TITS.2016.2547641.

7 Marwa Chaieb, Mirko Koscina, Souheib Yousfi, P. Lafourcade, and Riadh Robbana. Dabsters: a privacy preserving e-voting protocol for permissioned blockchain. In *16th International Colloquium on Theoretical Aspects of Computing*, ICTAC, 2019.

8 A. Durand, E. Anceaume, and R. Ludinard. STAKECUBE: Combining Sharding and Proof-of-Stake to build Fork-free Secure Permissionless Distributed Ledgers. In *7th International Conference, (NETYS)*, 2019. URL: https://hal.archives-ouvertes.fr/hal-02078072.

9 A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In *37th Annual International Cryptology Conference*, CRYPTO, 2017.

10 Marius Lombard-Platet and P. Lafourcade. Get-your-id: Decentralized proof of identity. In *12th International Symposium on Foundations and Practice of Security - Revised Selected Papers*, FPS, 2019.

11 S. Micali. ALGORAND: the efficient and democratic ledger. *CoRR*, abs/1607.01341, 2016. arXiv:1607.01341.

12 S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009. URL: http://www.bitcoin.org/bitcoin.pdf.

13 K.J. O'Dwyer and D. Malone. Bitcoin mining and its energy footprint. *IET Conference Proceedings*, pages 280–285(5), 2014. URL: `https://digital-library.theiet.org/content/conferences/10.1049/cp.2014.0699`.

14 Sunoo Park, Albert Kwon, Georg Fuchsbauer, Peter Gazi, Joël Alwen, and Krzysztof Pietrzak. Spacemint: A cryptocurrency based on proofs of space. In *Financial Cryptography and Data Security - 22nd International Conference, FC 2018*, volume 10957, pages 480–499. Springer, 2018.

Blockguard: Adaptive Blockchain Security

Shishir Rai
Kent State University, OH, USA
srai@kent.edu

Kendric Hood
Kent State University, OH, USA
khood5@kent.edu

Mikhail Nesterenko
Kent State University, OH, USA
`http://vega.cs.kent.edu/~mikhail/`
mikhail@cs.kent.edu

Gokarna Sharma[1]
Kent State University, OH, USA
`http://www.cs.kent.edu/~sharma/`
sharma@cs.kent.edu

Abstract

We change the security of blockchain transactions by varying the size of consensus committees. To improve performance, such committees operate concurrently. We present two algorithms that allow adaptive security by forming concurrent variable size consensus committees on demand. One is based on a single joint blockchain, the other is based on separate sharded blockchains. For in-committee consensus, we implement synchronous Byzantine fault tolerance algorithm (BFT), asynchronous BFT and proof-of-work consensus. We evaluate the performance of our adaptive security algorithms.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Computing methodologies → Distributed algorithms; Networks → Security protocols; Security and privacy → Distributed systems security

Keywords and phrases Blockchain, Consensus, Security, Distributed algorithms

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2020.12

Category Short Paper

Related Version A full version of the paper is on arXiv: `https://arxiv.org/abs/1907.13232`.

Supplementary Material The source code is available on GitHub at `https://github.com/khood5/distributed-consensus-abstract-simulator.git`.

1 Introduction

A secure distributed ledger, *blockchain* allows a decentralized network of peers to register a sequence of transactions despite potentially malicious actions of a minority of peers. Blockchain technology is poised to revolutionize a variety of fields: from currency and payment systems that are impervious to state and corporate manipulation, to automatically enforced contracts, to internet-of-things massive data recording.

In this paper, we state the problem of adaptive security and propose two efficient algorithms that solve it. We evaluate their performance with major consensus algorithms: PBFT, SBFT and proof-of-work. We measure their throughput, transaction waiting time and resistance to Byzantine peer corruption. Our results suggest that the adaptive security

[1] Corresponding author.

provides an effective trade-off between network performance and security without significant increase in network complexity or major architectural modifications. Thus, it should be adopted by current blockchain networks.

2 Definitions and Committee Consensus Algorithms

A set of n *peer processes* (or *peers*) forms a network to maintain the blockchain. The *blockchain* is a sequence of blocks or transactions. We use the terms interchangeably, i.e. we assume that a block contains a single transaction. A *transaction* is a unit of blockchain recording. Each subsequent transaction is cryptographically linked to the previous one. The first transaction in the blockchain is the *genesis* transaction. Peers communicate through broadcasts. Message delivery is FIFO. There is no message loss. Messages cannot be forged. Peers are either *honest* or *Byzantine.* A set of peers that cooperate to approve a transaction despite actions of Byzantine peers is a *consensus committee.*

Sharding. A *(recording) group* is a set of processes that maintain a single blockchain. There are as many groups as there are separate blockchains. In case of sharding, a peer in the consensus committee that approves a certain transaction in a blockchain does not necessarily belong to the group that records it. However, a peer may belong to only one recording group and only one consensus committee at a time.

PBFT and SBFT. In PBFT [2] The committee of peers elect the *leader.* The leader runs consensus on every transaction. It initiates several message exchanges with other committee peers. A non-leader Byzantine peer may delay messages or send incorrect messages. A Byzantine leader may temporarily block the consensus by sending different messages to different peers or not sending messages altogether. In either case, the honest peers discover the Byzantine leader and replace it by forcing a *view change.* PBFT is guaranteed to withstand up to $f < n/3$ Byzantine peers regardless of the message propagation delay. The operation of SBFT [1] is similar to PBFT. This algorithm relies on at least one honest peer confirming the transaction. However, it assumes that there is a bound on communication delay between honest peers. If a message is not received after a certain delay, it is guaranteed never to arrive. Thus, the algorithm has to delay to ascertain this lack of message receipt. In practice this may make SBFT slower. However, it has higher resilience threshold. It can tolerate up to $f < n/2$ Byzantine peers.

PoW. We implement proof-of-work consensus similar to Nakamoto [3]. To attach a new transaction to the blockchain, a peer *mines* the transaction by solving a computationally intensive task that links the new and previous transaction. Several peers may mine transactions concurrently. This is a *fork* in the blockchain. A branch of a fork may be extended by the addition of mined transactions on top of the current block. The shorter branch is discarded. PoW consensus operates correctly provided that the computational power of honest peers exceeds that of Byzantine peers. If peers have the same computational power, PoW consensus tolerates up to $f < n/2$ Byzantine peers.

3 The Adaptive Security Problem and Solutions

The problem. *The* Adaptive Security Problem *requires, as a solution, an adaptive security algorithm, to assign committees to the transactions such that each committee satisfies the transaction security level.* We consider an adaptive security algorithm that selects appropriate size committees and processes transactions with as much parallelism as possible. We present two such algorithms: *Composite Blockguard* and *Dynamic Blockguard*.

Composite Blockguard adaptive security algorithm. In this algorithm, peers are divided into storage groups maintaining independent blockchains. The algorithm maintains a list of idle groups and pending transactions. Once a new transaction arrives or a consensus committee is done, Composite Blockguard finds appropriate number of available groups, forms a consensus committee to process the next pending transaction and dispatches the transaction. If not enough idle groups are available, the pending transactions wait.

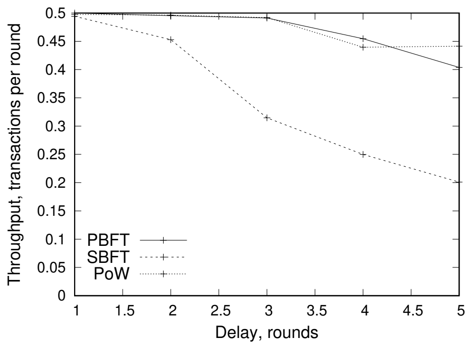
Dynamic Blockguard adaptive security algorithm. This algorithm has a single blockchain and thus a single recording group. A consensus committee is selected out of this group of peers. Multiple consensus committees may operate concurrently if their members do not intersect. This means that the committees have to concurrently write to the same blockchain. To ensure the integrity of the blockchain, the computation proceeds by alternating two stages: consensus stage and recording stage. In the *consensus* stage, committees agree on blocks to be written to the blockchain. Every committee must reach consensus before any committee may proceed to the next stage. In the *recording* stage, each committee broadcasts the transaction to the group maintaining the blockchain. That is, they broadcast it to the whole network. Each written transaction is cryptographically linked to all the written transaction in the previous recording stage. This way, the resultant blockchain is a series-parallel graph. *Committee selection window* is the set of unique peers that published in the blockchain most recently. Committee peers are picked at random from the committee selection window.

4 Performance Evaluation

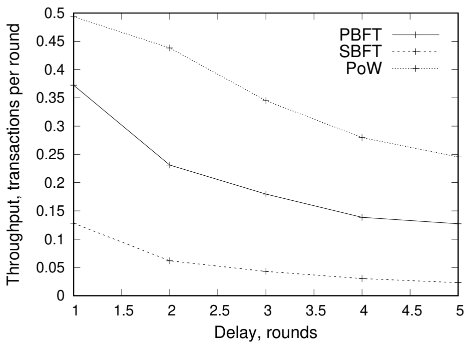
Setup. We evaluate the performance of Composite and Dynamic Blockguard using abstract simulation. The behavior of each algorithm is represented as a sequence of rounds. In every round, each peer may receive a single new message, do local computation and send messages to other peers.

Byzantine peers' goal is to successfully commit a fraudulent transaction to the blockchain, we model this as follows. A committee is *reliable* if the number of Byzantine peers in it does not exceed its tolerance threshold, *defeated* otherwise. The tolerance threshold is 1/3 for PBFT and 1/2 for SBFT and PoW. Defeated committees commit only fraudulent transactions to the blockchain, and reliable committees never commit fraudulent transactions. Byzantine leaders propose only fraudulent transactions. If a fraudulent transaction is proposed in a reliable committee then a view change occurs. This repeats until a non-byzantine leader is found. In PoW, if a Byzantine peer is the first to mine in a reliable committee then nothing is recorded and mining restarts.

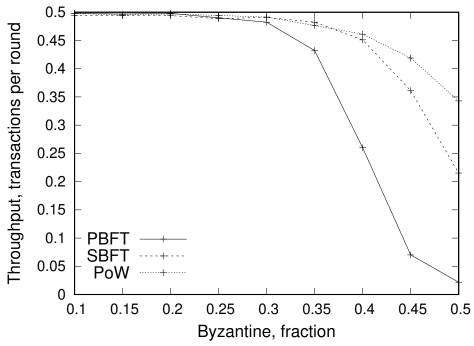
Experiment parameters and evaluation metrics. Unless stated otherwise, in the below experiments, the parameters are set as follows. The fraction of Byzantine faults is $n/10$. The number of peers in the network is 1024. There are 1000 rounds in a computation. Each data point is the average of 10 computations. A new transaction is generated every two



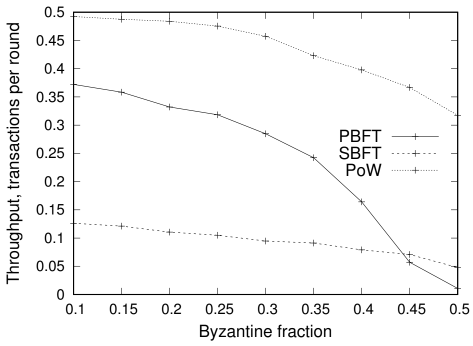
(a) Throughput, Composite Blockguard, varying delay



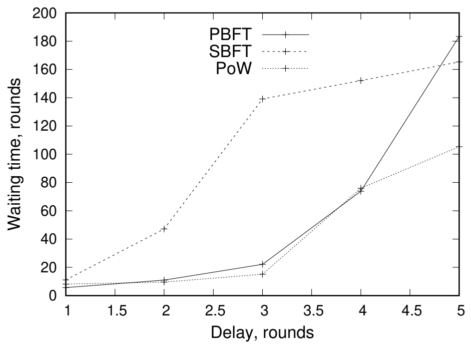
(b) Throughput, Dynamic Blockguard, varying delay



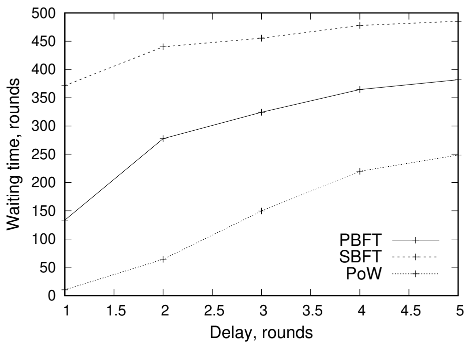
(c) Throughput, Composite Blockguard, varying Byzantine fraction



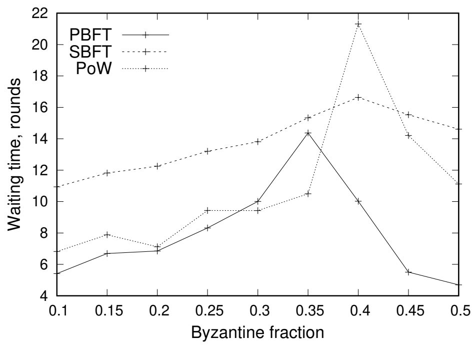
(d) Throughput, Dynamic Blockguard, varying Byzantine fraction



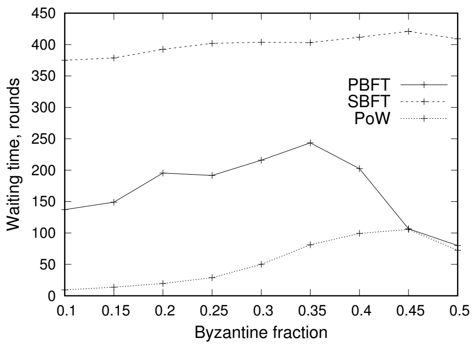
(e) Waiting time, Composite Blockguard, varying delay



(f) Waiting time, Dynamic Blockguard, varying delay



(g) Waiting time, Composite Blockguard, varying Byzantine fraction



(h) Waiting time, Dynamic Blockguard, varying Byzantine fraction

Figure 1 Performance of Blockguard adaptive security algorithms.

rounds. We have 5 security levels. The highest security level is the 5-th level which contains the whole network. Each lower level contains half of the peers of the higher level. We use geometric distribution to select the security level of newly generated transaction. In PoW, we use binomial distribution to determine the number of rounds it takes the peers to mine a transaction. The mode, i.e. most frequently occurring value, is 5 and variance 2.5. We vary maximum message delay and the fraction of Byzantine peers in the network. We consider a transaction approval as a consensus. We compute the following metrics. *Throughput* is the number of consensuses per round. Consensuses of defeated committees are not counted. *(Transaction) waiting time* is computed as follows. For coordinated consensus algorithms, i.e. PBFT and SBFT, it is the number of rounds from the moment the transaction is generated till the first peer determines that the transaction is committed. For PoW, it is the time for this transaction to be mined. The waiting time for transactions of defeated committees is counted.

Algorithm performance experiments. The results of the performance evaluation of the adaptive security algorithms are shown in Figure 1. Figures 1a and 1b demonstrate how throughput depends on the network delay for Composite and Dynamic Blockguard respectively. As network delay increases, the throughput declines. However, different consensus committees react to this increase differently. PBFT has the best performance and lowest decline since the committees just wait for the actual messages to arrive. SBFT exhibits the most sensitivity to the network delay. The reason is that SBFT has to wait for the maximum delay to determine that the message is not coming. Let us discuss Figures 1c and 1d. It shows that the performance of Composite and Dynamic Blockguard decreases as the fraction of Byzantine peers in the network increase. This is due to Byzantine peers slowing down the consensus algorithms. PBFT suffers the most since its tolerance threshold is only a third of the peers.

Figures 1e and 1f show the dependency of transaction waiting time on network delay. As expected, the waiting time increases with delay. SBFT is the most vulnerable to this increase since it has to wait for maximum delay time. Figures 1g and 1h show how waiting time varies with the fraction of Byzantine peers. Let us explain the trends in the data. As the consensus committee approaches its resiliency threshold, the number of view changes or repeated transaction mining increases which increases the transaction waiting time. If the fraction is away from this threshold, the committees are either reliable or defeated. In either case the waiting time is relatively low. Thus, there is a peak near $n/3$ for PBFT and near $n/5$ for SBFT and PoW. This trend is less pronounced in Dynamic Blockguard since it is masked by synchronization across consensus committees in the same stage.

The results of our experiments indicate that both Composite and Dynamic blackguard algorithm provide adaptive security with a trade-off between performance and security parameters.

References

1 Ittai Abraham, Srinivas Devadas, Kartik Nayak, and Ling Ren. Brief announcement: Practical synchronous byzantine consensus. In *DISC*, pages 41:1–41:4, 2017. `doi:10.4230/LIPIcs.DISC.2017.41`.

2 Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461, November 2002. `doi:10.1145/571637.571640`.

3 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system," http://bitcoin.org/bitcoin.pdf, 2008.

Welcome to the Jungle: A Reference Model for Blockchain, DLT and Smart-Contracts

Julien Hatin
Orange Labs, 42 rue des Coutures, Caen, France

Emmanuel Bertin
Orange Labs, 42 rue des Coutures, Caen, France

Baptiste Hemery
Orange Labs, 42 rue des Coutures, Caen, France

Nour El Madhoun
ISEP, 28 rue Notre Dame des Champs, Paris, France

Abstract

Blockchain technology has gained increasing attention from research and industry over the recent years. This interest is mainly due to its core property that allows users to perform transactions without a Trusted Third Party (TTP), while offering a transparent and fully protected tracking of these transactions. However, there is a lack of reference models to describe and compare various Blockchain technologies, leading to some confusion between different kinds of solutions. We propose in this paper a reference model aiming to assess and compare different kind of Blockchain-based ecosystems, including Decentralized Applications (DApp).

2012 ACM Subject Classification General and reference → Surveys and overviews; Computer systems organization → Distributed architectures

Keywords and phrases Blockchain, DLT, Smart-Contracts

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2020.13

Category Short Paper

1 Introduction

Starting from the Bitcoin application ten years ago, Blockchain and Distributed Ledger Technologies (DLT) have since considerably expanded in both industry and academic communities, leading to a very fragmented and somehow puzzling landscape. In this context, the aim of this paper is to define a reference model for Blockchain and DLT stakeholders, to properly characterize various ecosystems and use-cases, especially in the case of distributed applications. While existing modeling works have focused either on the engineering of DLT solutions, or on the business relationships between the stakeholders, we intend to propose a model addressing the interactions between these both levels. This model may then be used as a kind of "traveler's guide" for a Blockchain journey, enabling to better model needs and possible solutions.

2 State of the Art and Methodology

In the state of the art, few papers address the question of modelling DLT. Most research works in this field survey the Blockchain technology and its application use cases while introducing some high-level modelling on how transactions are performed [3, 12, 13, 14]. [12] is a typical example of this category of articles, providing a synthesis on Blockchain properties and a typology of application domains. However, some papers address more explicitly the question of modeling Blockchain and DLT. [5] introduces for example a comprehensive UMLbased Blockchain ontology, based on a study on how the Blockchain technology operates (transactions, blocks, etc.). However, such work remains focused on the theoretical Blockchain operations, and not on existing Blockchain or DLT solutions.

To fill this research gap, we have chosen to apply an empirical approach rather than starting from a theoretical study on the way Blockchain technology operates. To reach this target, we selected nine typical DLT solutions and studied their architecture, focusing on both similarities and differences. This choice of these nine DLT solutions is based on the reputation and usage, combined with a will to consider solutions with different architectures. We also focused not only on fundamentals of Blockchain (i.e. mining blocks to build a distributed ledger), but also on their usages with dApps (distributed applications) and smart contracts. However, we acknowledge that this choice of nine solutions might include some bias, due to our knowledge and practice of these technologies. This constitutes a limit of our work that we intend to address in our future work by extending the scope of the surveyed solutions.

3 Reference Model

DLT are not only bringing technical changes, but also changes in the actor model and in the value chain. This is precisely what we intend to capture in this article. Figure 1 details our reference model of the actors building a Blockchain ecosystem.

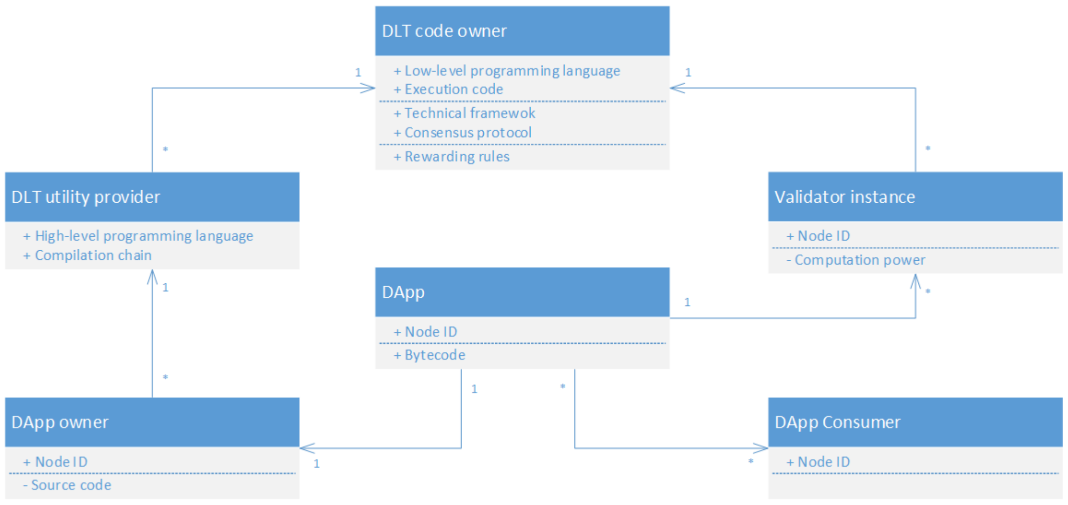


Figure 1 Actor model of DLT.

More precisely, we identified 6 different roles. The *DLT Code Owner* role is assigned to the company or organization that develops and maintains the original source code. Trust in the Core Code is essential, as the whole security of the blockchain network depends on it. Three elements are produced by the actor playing this role:

- The protocol that enable to issue, exchange and validates blocks and executes the various decentralized application.
- The low-level assembly-like language of dApp.
- The virtual machine that is able to execute it.

To run a decentralized application, a network of peers is needed. The members of this network are the blockchain nodes. A node has access to the ledger, can execute transaction and DApp on the blockchain.

To build concrete application, a set of tools is mandatory. They are provided by the *DLT Utility Provider*. Those tools consist of a human usable high-level language (i.e. C++ for EOS or solidity for Ethereum) and of the tools to compile it. Those mandatory tools are needed to build concrete dApp.

Using tools from the *DLT Utility Provider*, a *DApp Owner* can develop a new decentralized application. The compiled code is then published on the DLT by the *DApp Owner*. In blockchain architecture, the code is indeed not hosted by a central server. Instead the code is published in a network by the *DApp Owner*, and the storage is shared by every nodes of this network. The publication and the deployment of a decentralized application in this network can be done freely (i.e. EOS) or with fees (i.e. Ethereum or NEO).

A *DApp Consumer* is the end user of the decentralized application. It can execute function and read the result. Moreover, a *DApp consumer* is not necessarily a person or an organization but can also be another DApp. The function execution can be free, or with fees paid either by the *DApp Consumer* or the *DApp Owner*.

A *Validator Instance* is a key role in the blockchain systems: it validates the transaction. Depending on the platform this could be using a proof of stakes or a proof of work or even another consensus algorithm. The *DLT Code Owner* defines the consensus protocol and technical framework used by the validators, as well as potential incentives and rewarding rules. A *Validator Instance* is characterized by its computing power, in the case of Proof of Work mechanisms.

Finally, the *DApp* is also a part of the blockchain, holding the bytecode of the application. It is depending on the existence of *Validator Instances* to be published in the blockchain and to able to perform transactions.

At the end of this article, table 1 provides an extensive comparison between the surveyed DLT solutions according to the proposed role model.

4 Discussion and Application to existing DLT

The *DLT code owner* of any DApps-enabling blockchain is always endorsed by a single company for permissioned blockchains, or by a foundation for permissionless blockchains (cf Table 1). However, this entity is usually working with agile management process to include new features in the core code, e.g. the Lisk Improvement Proposition and the Ethereum Improvement Proposition. In addition, the core code is usually open source. Trusting the protocol and the low-level features is indeed required to enable trust between actors. An exception is Libra, which is permissioned, but managed by an association.

Another difference between the various *DLT Code Owners* lies in their level of control over their technical assets. We can identify here a first organizational strategy that aims at controlling the network protocol and leaving the application development and deployment to the *DApp owners* in association with their partners. In this category we find Libra, Ethereum, EOS, Lisk, and Hyperledger. Alternatively, other companies rely on existing open-source DLT (that they do not control) to build their solution on top of it. We can cite here Quorum, Monax, Counterparty. This model is usually associated with a tighter control on *DApps owners*.

We can also distinguish solutions by looking at their openness strategy. Two approaches exist here: the consortium approach and the free access approach. The main public blockchain that can run *DApps* as overviewed in this paper are Ethereum, Eos, Counterparty and Lisk. Those blockchains are open to any *DApp owner*. Oppositely, some actors have chosen to be selective on who can participate as they address a specific vertical such as Monax or Quorum.

Hyperledger have chosen a different approach by offering to its customers a solution to deploy their own consortium with their own partners. This can be a well-suited solution to construct quickly a blockchain environment. But this does not enable to open this network to anybody or to switch to a free access model. Facebook have chosen here to develop its own solution, which is very specific, because it starts as a permissioned network but aims to become a free access network.

5 Conclusion and perspectives

The proposed reference model is designed as a tool for helping practitioners (e.g., business managers and architects) to assess their choices in terms of roles and business models for designing DApps. It is also designed as a tool for researchers to ground studies on the value-chain of DLT and DApps, e.g. by simulating the behavior of the various actors according to different incentives

Our perspective is to use a multi-agent model, relying on the roles described in the model, to have a better comprehension of these actors' behaviors – in a context where *DApps owners*, *validator instances* and *DApps consumers* are all evolving in the jungle of competitive and often incompatible DLT solutions.

References

1 Block.one. Eos white paper, 2018. URL: `https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md`.

2 Vitalik Buterin et al. Ethereum white paper, 2013. URL: `https://ethereum.org/whitepaper/`.

3 Konstantinos Christidis and Michael Devetsikiotis. Blockchains and smart contracts for the internet of things. *Ieee Access*, 4:2292–2303, 2016. `doi:10.1109/ACCESS.2016.2566339`.

4 CounterpatyXCP. Counterpary documentation. URL: `https://counterparty.io/docs/`.

5 Joost de Kruijff and Hans Weigand. Understanding the blockchain using enterprise ontology. In *International Conference on Advanced Information Systems Engineering*, pages 29–43. Springer, 2017. `doi:10.1007/978-3-319-59536-8_3`.

6 Lisk Foundation. Lisk documentation. URL: `https://lisk.io/documentation/lisk-sdk/index.html`.

7 The Hyperledger White Paper Working Group. Hyperledger white papers. URL: `https://www.hyperledger.org/learn/white-papers`.

8 Libra Association Members. Libra white paper v2.0, 2020. URL: `https://libra.org/en-US/white-paper/`.

9 Monax.io. Monax documentation. URL: `https://docs.monax.io/`.

10 JP Morgan. Quorum wiki. URL: `https://github.com/jpmorganchase/quorum/wiki`.

11 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, Manubot, 2019.

12 Deepak Puthal, Nisha Malik, Saraju P Mohanty, Elias Kougianos, and Chi Yang. The blockchain as a decentralized security framework [future directions]. *IEEE Consumer Electronics Magazine*, 7(2):18–21, 2018. `doi:10.1109/MCE.2017.2776459`.

13 Horst Treiblmaier and Roman Beck. *Business Transformation through Blockchain*. Springer, 2019.

14 Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, 14(4):352–375, 2018. `doi:10.1504/IJWGS.2018.10016848`.

Table 1 Comparison of studied DLT.

Blockchain	DLT code owner	DLT utility provider	Developing language	DApp owner	DApp consumer	Validator Instance
Bitcoin[11]	Bitcoin Foundation	Bitcoin Foundation	Script	anyone	anyone	Bitcoin miners
Ehtereum[2]	Ethereum Foundation	anyone	solidity	ether holder	ether holder	Ethereum miners
Quorum[10]	Ethereum Foundation, JP Morgan	anyone	solidity	anyone with access to a permissioned node	anyone with access to a permissioned node	permissioned node
Counterparty[4]	Bitcoin Foundation	Counterpary	solidity	anyone	anyone	Bitcoin miners
EOS[1]	Block One	anyone	C++	token holder	anyone	approved block producer, elected by token holder
Libra[8]	Libra association	Libra association	Move	Association member	Libra holder	Association member
Monax[9]	The Linux Foundation	The Linux Foundation	solidity, Graphical flow	Any actor within the Monax system	Any actor within the Monax system	Monax affiliates
Lisk[6]	Lisk Foundation	Lisk Foundation	Javascript	anyone	anyone	101 active delegates
Hyperledger[7]	The Linux Foundation	The Linux Foundation	C++, solidity	Any actor within the system	Anyone within the system	Any actor